Scripting and Programming for Modeling, Simulation, and Control
## Contents

1 Introduction -- Technical computing in the 21st century .............................................................................................................................. ix
1.1 Chapter synopsis ...................................................................................................................................................................................... 1

2 Getting started with Maple's Document Mode: doing technical work with a clickable interface ................................................................. 7
2.1 Chapter synopsis ...................................................................................................................................................................................... 7
2.2 Starting up Maple, getting a fresh start ............................................................................................................................................. 7
2.3 Evaluating an expression involving exact arithmetic ......................................................................................................................... 8
   Grade school arithmetic ........................................................................................................................................................................... 8
   Diagnosing typographical mistakes ...................................................................................................................................................... 12
   Correcting typographical mistakes ...................................................................................................................................................... 15
   Exponentiation (powers). Numbers with lots of digits ....................................................................................................................... 16
   Detecting and fixing semantic and "logic" mistakes ......................................................................................................................... 17
2.4 Saving and retrieving your work ..................................................................................................................................................................... 18
2.5 Retrieving backups .................................................................................................................................................................................. 19
2.6 Algebra, plotting and mouse-clickable operations .............................................................................................................................. 20
   Algebraic expressions and equations. Solving equations. Working with pieces of expressions. .................................................................... 20
   For those with previous experience on other systems: some things are different, for a reason ................................................................. 20
   Plotting, approximate numerical solutions through cursor position .................................................................................................. 21
   The Expressions Palette and the Common Symbols Palette: entering Trigs, logs, roots, exponentials ......................................................... 25
   Approximate numerical (calculator - type) arithmetic in Maple ............................................................................................................. 26
   Evaluation, and selection of pieces .......................................................................................................................................................... 27
2.7 A quick-reference summary to this chapter ........................................................................................................................................... 28

3 Chapter 3 Technical word processing .................................................................................................................................................. 35
3.1 Chapter Overview ................................................................................................................................................................................... 35
3.2 Maple as a word processor ................................................................................................................................................................... 35
3.3 Shortcuts to entering math symbols ................................................................................................................................................. 38
3.4 Other word processor features ............................................................................................................................................................ 38
3.5 Troubleshooting word processing ........................................................................................................................................................ 38
3.6 Summary of Chapter 3 material ............................................................................................................................................................ 39

4 Chapter 4 Assignment ................................................................................................................................................................................... 41
4.1 Chapter Overview ................................................................................................................................................................................... 41
4.2 Assignment: remembering results for future use ..................................................................................................................................... 41
4.3 := is a keyboard shortcut for assignment ........................................................................................................................................... 42
4.4 The Variables Panel: a way to know the current value of your variables ............................................................................................ 43
4.5 Differences between the appearance of the worksheet and the state of a Maple session ........................................................................ 45
4.6 restart causes all variables to be reassigned ........................................................................................................................................ 47
4.7 Evaluation and chains of assignments..................................................................................................................................................... 49
4.8 Troubleshooting assignments .................................................................................................................................................................. 50
4.9 Summary of Chapter 4 material ............................................................................................................................................................ 54

5 Chapter 5 Building scripts ............................................................................................................................................................................ 57
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.9</td>
<td>Designing a function to use in animate</td>
<td>144</td>
</tr>
<tr>
<td>9.10</td>
<td>Designing a more elaborate animation</td>
<td>150</td>
</tr>
<tr>
<td>9.11</td>
<td>Solving a problem with the help of animation</td>
<td>155</td>
</tr>
<tr>
<td>9.12</td>
<td>Exporting animations and non-animated plots</td>
<td>162</td>
</tr>
<tr>
<td>9.13</td>
<td>Summary of Chapter 9</td>
<td>163</td>
</tr>
<tr>
<td>Index</td>
<td></td>
<td>165</td>
</tr>
</tbody>
</table>
**List of Tables**

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>An early computer</td>
<td>2</td>
</tr>
<tr>
<td>1.2</td>
<td>A high-end calculator in 2009</td>
<td>3</td>
</tr>
<tr>
<td>2.1</td>
<td>Maple started up with new document in Windows XP</td>
<td>7</td>
</tr>
<tr>
<td>2.2</td>
<td>Maple document with first entry area</td>
<td>8</td>
</tr>
<tr>
<td>2.3</td>
<td>Maple input using 2d math</td>
<td>8</td>
</tr>
<tr>
<td>2.4</td>
<td>Maple input with labeled result</td>
<td>9</td>
</tr>
<tr>
<td>2.5</td>
<td>Arithmetic Operations in Maple</td>
<td>9</td>
</tr>
<tr>
<td>2.6</td>
<td>Examples of Maple error messages</td>
<td>12</td>
</tr>
<tr>
<td>2.7</td>
<td>Create Document Block to force a Math input area wherever the cursor is placed</td>
<td>16</td>
</tr>
<tr>
<td>2.8</td>
<td>Example of a vocabulary mistake</td>
<td>17</td>
</tr>
<tr>
<td>2.9</td>
<td>Maple save menu operation</td>
<td>18</td>
</tr>
<tr>
<td>2.10</td>
<td>Maple save dialog box</td>
<td>19</td>
</tr>
<tr>
<td>2.11</td>
<td>The Maple state display</td>
<td>19</td>
</tr>
<tr>
<td>2.12</td>
<td>Example of Plotting</td>
<td>22</td>
</tr>
<tr>
<td>2.13</td>
<td>Plot created by right-click -&gt; Plot -&gt; 2DPlot (with cursor positioning)</td>
<td>23</td>
</tr>
<tr>
<td>2.14</td>
<td>User-configured plot using PlotBuilder instead of 2DPlot</td>
<td>23</td>
</tr>
<tr>
<td>2.15</td>
<td>PlotBuilder Dialog box</td>
<td>24</td>
</tr>
<tr>
<td>2.16</td>
<td>The Expression palette</td>
<td>25</td>
</tr>
<tr>
<td>2.17</td>
<td>Examples of palette-driven computation</td>
<td>26</td>
</tr>
<tr>
<td>2.18</td>
<td>Examples of computing with approximate solving</td>
<td>26</td>
</tr>
<tr>
<td>2.19</td>
<td>Evaluate at a point</td>
<td>27</td>
</tr>
<tr>
<td>2.20</td>
<td>Operations on equations, multi-part expressions</td>
<td>27</td>
</tr>
<tr>
<td>3.1</td>
<td>Maple in math entry mode</td>
<td>35</td>
</tr>
<tr>
<td>3.2</td>
<td>Document after control-T (or Insert-&gt;Text)</td>
<td>36</td>
</tr>
<tr>
<td>3.3</td>
<td>Document with a mixture of text and math</td>
<td>36</td>
</tr>
<tr>
<td>3.4</td>
<td>control-= puts the results of a calculation in the midst of text</td>
<td>37</td>
</tr>
<tr>
<td>3.5</td>
<td>Keyboard shortcuts in math mode through the escape key</td>
<td>38</td>
</tr>
<tr>
<td>4.1</td>
<td>Assignment</td>
<td>41</td>
</tr>
<tr>
<td>4.2</td>
<td>:= is another way to do assignment of a name</td>
<td>42</td>
</tr>
<tr>
<td>4.3</td>
<td>Assignment</td>
<td>42</td>
</tr>
<tr>
<td>4.4</td>
<td>The variables panel shows currently assigned variables</td>
<td>44</td>
</tr>
<tr>
<td>4.5</td>
<td>The state of a Maple Session</td>
<td>45</td>
</tr>
<tr>
<td>4.6</td>
<td>Evaluation of expressions involving assigned variables</td>
<td>48</td>
</tr>
<tr>
<td>4.7</td>
<td>BROKEN - MISSING TITLE!</td>
<td>49</td>
</tr>
<tr>
<td>4.8</td>
<td>Assignment := is not symmetric</td>
<td>51</td>
</tr>
<tr>
<td>4.9</td>
<td>Some Names can't be assigned to</td>
<td>52</td>
</tr>
<tr>
<td>4.10</td>
<td>Can't solve for a variable with an assigned value</td>
<td>52</td>
</tr>
<tr>
<td>5.1</td>
<td>Selecting parts of a solution</td>
<td>57</td>
</tr>
<tr>
<td>5.2</td>
<td>Basic data structures in Maple and operations to extract parts of them</td>
<td>57</td>
</tr>
<tr>
<td>5.3</td>
<td>Solving simultaneous equations</td>
<td>58</td>
</tr>
<tr>
<td>5.4</td>
<td>Executing a clone of a script through copy-paste</td>
<td>61</td>
</tr>
<tr>
<td>6.1</td>
<td>The textual form of equation solver solve</td>
<td>72</td>
</tr>
<tr>
<td>6.2</td>
<td>The textual version of plotting</td>
<td>73</td>
</tr>
<tr>
<td>6.3</td>
<td>Plotting of multiple expressions</td>
<td>75</td>
</tr>
<tr>
<td>6.4</td>
<td>Plotting points with lists of numbers</td>
<td>76</td>
</tr>
<tr>
<td>6.5</td>
<td>Plot options and labels</td>
<td>78</td>
</tr>
<tr>
<td>6.6</td>
<td>What happens when you forget to use the &quot; delimiter in strong, or use the wrong character for the delimiter</td>
<td>80</td>
</tr>
<tr>
<td>6.7</td>
<td>Plot command help</td>
<td>81</td>
</tr>
<tr>
<td>6.8</td>
<td>Examples of plot</td>
<td>82</td>
</tr>
<tr>
<td>6.9</td>
<td>Copying the example</td>
<td>83</td>
</tr>
<tr>
<td>6.10</td>
<td>Modifying the example</td>
<td>83</td>
</tr>
</tbody>
</table>
Table 9.18: Exporting animations through the pop-up menu .......................................................... 163
Table 9.17: Second try: shoot upwards at 22 feet per second .......................................................... 161
Table 9.16: A first try at shooting the ball up 30 feet: 15 miles per hour .......................................... 160
Table 9.14: Animating the ballFrame function to produce a movie of the ball in flight. .................... 159
Table 9.13: Trying out a function that draws the ball at a particular time. ........................................ 159
Table 9.12: A plot that draws a ball as a time t=0.5. ...................................................................... 158
Table 9.11: A plot that draws a ball as a circle at (0,6). ................................................................. 157
Table 9.10: A rough-draft plot of ball versus time. ...................................................................... 156
Table 9.9: Frames 1, 5, 10, 20, and 25 of the animation ............................................................... 143
Table 9.8: Animation pop-up menu .................................................................................................. 143
Table 9.6: First animation example .................................................................................................. 141
Table 9.5: Examples of plots where the x and y positions are parameterized by t .............................. 140
Table 9.4: Combining a line with other plots using display ............................................................ 138
Table 9.3: BROKEN - MISSING TITLE! .......................................................................................... 136
Table 9.2: Display combines plots .................................................................................................... 134
Table 9.1: Plot results, displayed and not displayed ...................................................................... 133
Table 8.9: several versions of a function with function definitions .................................................. 128
Table 8.8: A problem solved with a function that outputs a sequence of two numbers .................... 127
Table 8.6: A function that takes a list as its input ......................................................................... 124
Table 8.5: Unit Converter Assistant ............................................................................................... 124
Table 8.4: Solving multiple versions of a problem through functions ............................................. 122
Table 8.3: A problem solved with a script using function definition .............................................. 121
Table 8.2: Examples of Unit Conversion ....................................................................................... 119
Table 8.1: A script that uses functional (chaining), and its use ....................................................... 118
Table 7.10: Textual names of common operations in Maple ............................................................. 107
Table 7.9: with, the package loading operation ............................................................................. 106
Table 7.8: Troubleshooting function definitions ............................................................................. 105
Table 7.7: Comparing evaluation using eval and functional notation ............................................. 103
Table 7.6: User-defined functions through the Expression Palette .................................................. 101
Table 7.5: A function definition in a math textbook ....................................................................... 100
Table 7.4: Command Completion ................................................................................................... 98
Table 7.3: exp is the name of the exponential function ................................................................... 97
Table 7.2: Textual names for common math functions ................................................................... 95
Table 7.1: Function results in Maple ............................................................................................... 90
Table 6.16: Evaluating an expression using a particular value of one of the variables in the expression, and then solving ............................................................. 89
Table 6.15: Evaluating an expression using a particular value of one of the variables in the expression, and then solving ............................................................. 88
Table 6.14: Evaluating an expression using a particular value of one of the variables in the expression, and then solving ............................................................. 87
Table 6.13: Solving systems of equations with solve ....................................................................... 86
Table 6.12: Operations on lists ....................................................................................................... 84
Table 6.11: Plot option help page .................................................................................................... 84
Table 6.10: Plot option help page .................................................................................................... 84
Table 6.9: Functions for solving equations and inequalities ............................................................. 83
Table 6.8: Using solve and subs to solve equations and inequalities .............................................. 82
Table 6.7: Examples of linear equations ............................................................................................ 81
Table 6.6: A line through the points (0, 0) and (0, 1) ..................................................................... 80
Table 6.5: A line through the point (0, 0) and the vertical line x=1 .................................................. 79
Table 6.4: The slope of a line .......................................................................................................... 78
Table 6.3: The point-slope form of a line ............................................................................................ 77
Table 6.2: Examples of Unit Conversion ............................................................................................ 76
Table 6.1: A script that uses functional (chaining), and its use .......................................................... 75
Table 5.10: Textual names of common operations in Maple ............................................................. 72
Table 5.9: with, the package loading operation ............................................................................. 71
Table 5.8: Troubleshooting function definitions ............................................................................. 70
Table 5.7: Comparing evaluation using eval and functional notation ............................................. 69
Table 5.6: User-defined functions through the Expression Palette .................................................. 68
Table 5.5: A function definition in a math textbook ....................................................................... 67
Table 5.4: Command Completion ................................................................................................... 65
Table 5.3: exp is the name of the exponential function ................................................................... 64
Table 5.2: Textual names for common math functions ................................................................... 62
Table 5.1: Function results in Maple ............................................................................................... 58
Table 4.16: Evaluating an expression using a particular value of one of the variables in the expression, and then solving ............................................................. 53
Table 4.15: Evaluating an expression using a particular value of one of the variables in the expression, and then solving ............................................................. 52
Table 4.14: Evaluating an expression using a particular value of one of the variables in the expression, and then solving ............................................................. 51
Table 4.13: Solving systems of equations with solve ....................................................................... 50
Table 4.12: Operations on lists ....................................................................................................... 49
Table 4.11: Plot option help page .................................................................................................... 49
Table 4.10: Plot option help page .................................................................................................... 49
Table 4.9: Functions for solving equations and inequalities ............................................................. 48
Table 4.8: Using solve and subs to solve equations and inequalities .............................................. 47
Table 4.7: Examples of linear equations ............................................................................................ 46
Table 4.6: A line through the points (0, 0) and (0, 1) ..................................................................... 45
Table 4.5: A line through the point (0, 0) and the vertical line x=1 .................................................. 44
Table 4.4: The slope of a line .......................................................................................................... 43
Table 4.3: The point-slope form of a line ............................................................................................ 42
Table 4.2: Examples of Unit Conversion ............................................................................................ 41
Table 4.1: A script that uses functional (chaining), and its use .......................................................... 40
Table 3.10: Textual names of common operations in Maple ............................................................. 37
Table 3.9: with, the package loading operation ............................................................................. 36
Table 3.8: Troubleshooting function definitions ............................................................................. 35
Table 3.7: Comparing evaluation using eval and functional notation ............................................. 34
Table 3.6: User-defined functions through the Expression Palette .................................................. 33
Table 3.5: A function definition in a math textbook ....................................................................... 32
Table 3.4: Command Completion ................................................................................................... 31
Table 3.3: exp is the name of the exponential function ................................................................... 30
Table 3.2: Textual names for common math functions ................................................................... 29
Table 3.1: Function results in Maple ............................................................................................... 25
Table 2.16: Evaluating an expression using a particular value of one of the variables in the expression, and then solving ............................................................. 20
Table 2.15: Evaluating an expression using a particular value of one of the variables in the expression, and then solving ............................................................. 19
Table 2.14: Evaluating an expression using a particular value of one of the variables in the expression, and then solving ............................................................. 18
Table 2.13: Solving systems of equations with solve ....................................................................... 17
Table 2.12: Operations on lists ....................................................................................................... 16
Table 2.11: Plot option help page .................................................................................................... 16
Table 2.10: Plot option help page .................................................................................................... 16
Table 2.9: Functions for solving equations and inequalities ............................................................. 15
Table 2.8: Using solve and subs to solve equations and inequalities .............................................. 14
Table 2.7: Examples of linear equations ............................................................................................ 13
Table 2.6: A line through the points (0, 0) and (0, 1) ..................................................................... 12
Table 2.5: A line through the point (0, 0) and the vertical line x=1 .................................................. 11
Table 2.4: The slope of a line .......................................................................................................... 10
Table 2.3: The point-slope form of a line ............................................................................................ 9
Table 2.2: Examples of Unit Conversion ............................................................................................ 8
Table 2.1: A script that uses functional (chaining), and its use .......................................................... 7
Table 1.10: Textual names of common operations in Maple ............................................................. 4
Table 1.9: with, the package loading operation ............................................................................. 3
Table 1.8: Troubleshooting function definitions ............................................................................. 2
Table 1.7: Comparing evaluation using eval and functional notation ............................................. 1
Table 1.6: User-defined functions through the Expression Palette .................................................. 1
Table 1.5: A function definition in a math textbook ....................................................................... 1
Table 1.4: Command Completion ................................................................................................... 1
Table 1.3: exp is the name of the exponential function ................................................................... 1
Table 1.2: Textual names for common math functions ................................................................... 1
Table 1.1: Function results in Maple ............................................................................................... 1

viii • List of Tables
Dedication

To our students, who learn how to work with the new and different.

To my family, whose support is unwavering.

Acknowledgements

Jeremy Johnson started the Computation Lab course. He chose the initial direction and approach of the course. Frederick W. Chapman worked with us in 2006-2008 in the development of the course.

We are grateful to our colleagues David Augenblick, Jeremy Johnson, Mark Boady, and Brian Dolhansky for their help in the development of this text. David was a careful proofreader and useful sounding board. Jeremy's shrewd counsel and feedback often led to improvements. Brian's expert and speedy work in tagging and repairing the manuscript kept us on schedule. We also appreciate the assistance of Maplesoft for technical advice and support, especially that of Karishma Punwani, Bill Laarakkers, and Christina Spirou.
1 Introduction -- Technical computing in the 21st century

1.1 Chapter synopsis

1. We explain what technical computing is about, why it's done with a machine and what kinds of devices are used to do it.

2. The advantages doing calculation with a computer are surveyed.

3. Maple, a system for technical computing is introduced and contrasted with some of the other choices available today.

1.2 What is technical computing? What kinds of technical computing are there? What are we studying in this course?

Well, obviously, it's "computing done for technical work" -- primarily science and engineering, but any other field where mathematical and scientific reasoning is used. Nowadays this can include finance and business, health care, or digital media as well as the traditional math/science/engineering domain. You are probably familiar with numerical computing since it's done even without computers, with paper and pencil or with calculators in high school or even earlier. You probably have also seen graphing calculators do simple kinds of graphical computing, also sometimes known as mathematical visualization. In addition to this, computers can and are used to do symbolic computing -- computing where the answer is a formula rather than a number, as well as deductive or logical computing -- using logic or deduction to find answers. In this course, we will concentrate on numerical, symbolic and graphical computing, although we will see some instances of deductive computing as well.

The success of such reasoning in these fields is well-known. Students should aim for proficiency in technical computing to be able to be able to obtain success. Because of the many applications, languages, and software tools used nowadays for successful technical computing, the learning task is not "learn one programming language that will serve all needs". They need to learn the terminology and conceptual approaches used so that they can use their first experiences with technical computing as leverage to continuing self-taught learning and proficiency. Being able to build on the initial experience, rather than being defined and limited by it, is what will allow access to the stream of many applications and software tools that will be in play and in vogue during a career.

1.3 What are the advantages of doing technical calculation with a computer?

One advantage is greater quantity of calculation-- computers can do calculations billions of times faster than humans and thousands or millions of times faster than calculator-driven computation.. Using them makes some things feasible that are not possible any other way:

As you've seen with your high school physics, chemistry, and math courses, scientists and engineers often work with mathematical models -- systems of equations, symbols, and mathematical relations that try to describe key aspects of a situation. Calculation with models helps answer questions about the situation being modeled: How much fuel will be used in this operating scenario? How long will it take to heat up the furnace to 265 degrees? Computer simulation loads a model with some initial conditions, and then through calculation produces predictive results. Simulation can be used to make forecasts such as: how fast will the vehicle be moving after 5 seconds? How far will the pollutants move underground after being buried five years? Computer simulations can sometimes generate predictions even when standard techniques of "mathematical solution" are not adequate to find an answer.

Computers make it possible to use models that require much more extensive calculations to reach conclusions. The payoff may be better predictions. Extensive calculation also makes it possible to generate and present more information. Computer visualization goes beyond that to highly detailed pictures or animations of a situation described through a mathematical model. This can lead to better insight and understanding.

Another advantage is that it is easier to make a lasting written record of the work. One of the big differences between professional technical work and homework is that the modeling and calculation has lasting value: it matters to more than one or two people, and it is being used in an on-going project. In a professional situation, a worker may need to go back and review the work a year later, long after the details are faded from easy recall. Other people may want to reuse the programming and will need to be given an explanation about how to use it. Since they have long-term value, it is beneficial to put them into a form suitable for easy future reference and reuse. Thus the work typically includes both programming and documentary explanation.
1.4 The spectrum of devices for technical computing

Back in the days of the original electronic computers developed during and immediately after World War II, the only kind of device you could use for technical computing (which was the only computing that could be justified in those days) filled up a whole room and needed an extensive support staff.

Table 1.1: An early computer

| ENIAC, one of the first electronic computers, being programmed by Herman Goldstine at the University of Pennsylvania circa 1946. (US Army photo) Originally intended to do artillery calculations, it was used for the design and development of the first thermonuclear weapons. |
| See http://www.seas.upenn.edu/~museum/.

Today's users have a choice of a wide variety of devices:

**Personal computers**

Typically a computer for individual use can be expected to have the following features:

1. Processing capability of a billion or more arithmetic or memory operations per second. Significant amounts of calculation are used in doing the graphics involved in processing digital media or supporting the artificial worlds presented in computer games. This same power can be used to do the simulation and exploration in many common kinds of scientific and engineering situations. This course is about using that calculation power.

2. Standard selection and pointing devices, keyboard and mice. Some computers support "tablet" operation which allow use of a pen writing on the display. The display area and input devices make it easy to enter information via text or by pointing/selection. While the World Wide Web has made much processing possible with just "pointing and clicking", the complexity and mathematical sophistication of the models used in current technical computing outstrip the capabilities of current "point and click" technology. Some (computer programming) language-based interaction is necessary to handle things deftly. This requires a more knowledgeable user, but many people find that the extra power is worth the learning effort.

3. A screen capable of displaying information equivalent to one or more 8 1/2 x 11 inch pieces of paper. This helps support the development of documentation, or of more complex visualizations.

4. Local storage capable of storing a significant fraction of the text of the books in the Library of Congress. Of course, it's easy for a personal media collection to fill up this kind of storage, but it should not be forgotten that the billions of characters that this storage represents can store years or decades of efforts of computer programmers.

5. Connection to the internet, which makes it easier to communicate and share work with others, and to download new programming. From the viewpoint of technical computing, that it's easy to import and run substantial amounts of programming.

**High performance computers, also known as "supercomputers"**

While hand held devices and personal computers are used widely by the general public for non-technical purposes, there are a class of computers that are used primarily for mathematically-based computing. Typically instead of billions of numerical operations per second ("gigaflops" -- billions of floating point operations), they employ multiple processors in parallel to do trillions of operations ("teraflops") or quadrillions ("petaflops"). The hardware, personnel and energy costs of such computers are significant, so typically they are available through centers used through the Internet from many locations. The kinds of technical problems such devices are used for would include: simulation of complicated physical situations, such as for climate prediction, astrophysics, or engineering design (aircraft, groundwater, large building, automobile).

Using such large devices incur significant expense, from hardware, software and support staff costs, as well as electrical power consumed. For example, the U.S. Department of Energy has an IBM computer at its Los Alamos National Laboratory in New
Mexico, USA capable of 1 quadrillion ($10^{15}$) numerical operations per second. It consumes 2345.50 kilowatts when it is running. (reference: http://blog.enterpriseproducton.com/green/blog/2008/06/green-petaflop-ibms-roadrunner-wins-supercomputer-top-spot.html) The National Center for Computational Sciences (NCCS) at Oak Ridge National Laboratory in Tennessee, USA which has several large supercomputers, has a 2008 annual budget between $80 million and $100 million. (reference: http://news.cnet.com/8301-13772_3-9985500-52.html). A typical personal computer might be capable of approximately 4 billion numerical operations per second, so the Blue Gene computer at Los Alamos provides roughly $25 \times 10^6 = 250,000$ times more computing power.

Typically the programming is developed on personal computers, then moved to the larger devices. Usually the expensive supercomputer time is spent primarily on numerical computation, rather than on providing a nice-to-use interface for users. However, the results of a supercomputer may be shipped over the Internet to a personal computer so that a scientist or engineer may mull over the results in a more contemplative way without incurring additional supercomputer costs.

Multiple computers may be linked together over the Internet to get all the pieces of a particular elaborate computation done. This may have the supercomputers performing the massive numerical computations, while other smaller or personal computers linked in may be displaying the results to a distributed team of investigators who are collectively digesting results and steering the ongoing work.

**Hand held mobile devices**

Calculators are useful for casual computation, where one wants to figure out the solution to a small problem once. It's easy to punch in a few numbers and operations and to read the answer on the display. As inexpensive and small mobile devices, these are typically more limited in:

1. memory (limits to computation size and to built-in features)
2. energy consumption (slower processor speed)
3. form factor (keyboard and display too small for more than casual technical use)

Typically calculators are not networked, making it harder to share results to transfer them elsewhere to continue the work.

### Table 1.2: A high-end calculator in 2009

<table>
<thead>
<tr>
<th>Description</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>The TI-Nspire with CAS</td>
<td>A recent generation calculator from Texas Instruments. It can do numerical and symbolic calculations, as well as graphing. According to ticalc.org (<a href="http://www.ticalc.org/basics/calculators/ti-nspire-cas.html">http://www.ticalc.org/basics/calculators/ti-nspire-cas.html</a>) it has 16Mb memory, 20Mb storage and has a 150MHz processor. This makes it have about 100 times less memory, 8000 times less storage, and makes it run twenty times slower than a low-end laptop with a dual core 1.5GHz processor, 2Gb memory, and 160Gb disk. Its screen is 240 x 320 pixels, giving it about twenty times less display area than a typical laptop. This is one of the first generation of calculators with limited wireless networking available as an add-on.</td>
</tr>
</tbody>
</table>


**Smartphones, personal digital assistants (PDAs), media players, and tablets** can have processing capabilities almost as powerful as small personal computers. One advantage that these kinds of devices hold over calculators, is that they are typically networked so that it's possible to use them as a front end to a more powerful computer somewhere else in the Internet "cloud" of computational resources. Mark Dean of IBM, part of the engineering team that developed the original PCs for that company in the '80s, has written that the wave of popularity for personal computers has crested, "they're going the way of the vacuum tube, typewriter, vinyl records, CRT and incandescent light bulbs." (http://asmarterplanet.com/blog/2011/08/ibm-leads-the-way-in-the-post-pc-era.html). However the limitations of the small display and keyboard with current mobile devices will continue to constrain the appeal of such devices for extensive technical work for the near term.
A high-end tablet in 2011

The iPad2 is a recent generation networked mobile device from Apple. According to http://www.apple.com/ipad/specs/ and http://www.engadget.com/2011/03/09/ipad-2-review/, it can be configured with 512Mb memory, 64Gb storage, and a 1GHz dual core processor. Its screen is 1024 x 768 pixels. This makes it slightly less capable than a low-end laptop with a dual core 1.5GHz processor, 2Gb memory, and 160Gb disk.

Photo from http://4.mshcdn.com/wp-content/uploads/2010/01/ipad-3g.jpg

Dedicated controllers

Even some kinds of toasters have microprocessors in them nowadays. "Smart homes" (see for example http://home.howstuff-works.com/smart-home.htm and http://www.drexelsmarthouse.com/) may network appliances and home features such as HVAC and lighting control. While these devices are merely configured rather than programmed by end users, students of technical computing should not forget that this is another place where programming is necessary. The economic reasons for switching to computer control of devices appear to be the greater flexibility and variety of control that can be developed at modest cost through programming for many kinds of devices. The kind of programming done for device control often has a mathematical basis. Although the processors in dedicated devices are typically a few orders of magnitude slower than those of personal computers or even smartphones, the programming languages used for them are often the same or similar to those used in personal computing.

1.5 Maple, a system for technical computing

In this course, you will learn how to do various kinds of technical tasks using Maple. The first version of Maple was developed in the early 1980s at the University of Waterloo in Ontario, Canada, but has since undergone many refinements and extension. (In Fall 2011, we will be using Maple 15.) Maple's original emphasis was on algebraic computing, so it was used by those who needed formulas as results for their work. In the early '90s it was used extensively in calculus classes as a way of supporting exploratory experimentation and more extensive experience with "applied" problems. More recently it has expanded its domain into the symbolic, numeric, and graphical calculations done more generally in scientific and engineering modeling and simulation.

Today, Maple supports numerical computing and graphical visualization about as well as it does symbolic computing. It has its own programming language but also has ways of doing calculations through the graphical user interface (GUI) that is a kind of augmented "point and click". The default way of interacting with Maple also allows you to mix documentation and computing instructions and results. Thus it's particularly easy in Maple to produce documents that produce a well-documented solution -- a description of the problem, the explanation for how to solve it, and the computations that produce the details of the answer and evidence to justify its correctness.

1.6 What about Systems X, Y, Z, ...?

There are a variety of systems and programming languages used for technical computing nowadays. In the TIOBE Programming Community Index for August 2011, Java, C, C++, PHP, C#, Objective C, Visual Basic, Python, Perl and JavaScript are, in that order, the ten most popular programming languages (see http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html). In a large technical establishment such as a university or research lab, one might find, in addition to the generally popular languages and Maple, systems such as Matlab, Mathematica, Octave, Macsyma, Sage, Axiom, REDUCE, Mupad, and Fortran. Maple and other "niche" languages are used despite the much greater popularity of the "top 20" languages because they specialize in addressing tasks common in technical fields.

The vast array of tasks computers can be used for, the multiplicity of philosophies of system design, and the rapid and unrelenting nature of progress in computing means that there is no "best system" for any situation. Each system has its technical strengths. There may also be historical, cultural, or economic reasons why certain systems are used in certain places which co-exist with the technical justification. As with all popular and demanding activities, there are many products in use.

One important ramification of this is that students in technical fields must expect to become familiar and proficient in more than one programming system. Studying more than one brings greater knowledge of diverse ways of achieving a goal. It also allows one to function effectively as a team member in larger cross-organizational efforts and to better to cope with the diversity in computing cultures that's out there.
1.7 Why pick Maple as the first system?

1. It is an interactive system, facilitating quick exploration of new ideas. Compared to languages such as C++ or Java, one can immediately start up an interactive system and calculate results through "point and click" and a little typing. It's also possible to enter computational scripts, which are sequences of steps that are less elaborate to set up than programs. The scripts can be easily re-run on variants of the original situation just by changing a line or two in the script. This allows convenient "what-if" exploration, where a number of different scenarios are explored through computation. The worksheet interface to Maple allows sophisticated mathematical typography and graphics as well as mouse- and palette-driven input.

2. It can handle calculations with formulas. Rather than figuring out the formula yourself, you can even get Maple to calculate the formula for you. While many mathematical calculations produce numerical results, formulas are needed to specify how those results are produced. Designers or developers often need to find and create the formulas rather than just copy them from a book. A system where formulas are easy to represent and easy to create supports this kind of work. A system that allows you to represent and calculate both numbers and formulas makes it easier to do and document technical calculation than a system that just works with numbers.

3. It supports a variety of data structures that support technical computation: formulas, equations, functions, sets, lists, tables, vectors, and matrices. Rather than having calculations done with a number of digits that's fixed forever, it is easy to change the precision if more (or less) precision is needed. Having higher-level interaction with the computer is usually more productive of human time. Having all of these entities as "first class entities" in the system requires less mental effort by the user, since they do not have to translate what they are thinking about (e.g. a table of formulas) into the terms that the computer is using. Rather, the computer language accommodates the human style of thought.

4. It supports documentation as well as calculation. From the instructor's point of view, it's easier to create documents that explain Maple because we can use Maple both to handle the calculation and the presentation of results. For students, having a file with both the directions that cause the computation and the results makes it easier to present the work cleanly for grading, and for future reference. For professionals doing technical work, having an integrated environment where text, programming and results can be combined together can be a convenience.

5. It has a "conventional programming language". An objective of this course is to make you become familiar enough with some of the standard elements of programming (e.g. assignments, conditionals, loops, procedure definitions) so that you can use them creatively to handle certain common situations that can't be handled through point-and-click or built-in operations. These elements will be found in highly similar form in the languages used by many other systems used for technical work.

6. The mathematics of modeling and simulation is an explicit feature of the language. While its programming language is conventional, Maple's language has an important added plus -- you can work over expressions, equations, functions, and other mathematical objects just as conveniently as if they were numbers or characters. Conventional languages (e.g. Java) can represent arithmetic operations -- multiplication, addition, subtraction, and division -- but after that there are special tricks and conversions that you must perform to bridge the gap between what is written in the program, and the mathematical ideas that you are trying to use in the model. It's more straightforward in Maple to represent the mathematical model and the computation based on it. This ease of expression and comprehension by programmers has a hard-headed dollars and sense payoff. Less programmer time can be spent developing a computation. There is also an efficiency advantage to explicit representation of formulae: it is possible to use systems such as Maple to automatically improve the quality and efficiency of simulation calculations. This again leads to lower costs of doing the work.

We think these things provide a software engineering advantage that will lead most technical computation systems to eventually have such functionality built-in into them.

1.8 Using more than one system

Any user of computers who expects to use them professionally for design and investigation must expect to eventually learn multiple systems. Using computer applications for work is like using tools in a workshop-- you would not expect to use one tool to do all tasks, even if the tool, like Maple, has "Swiss Army Knife" capabilities. Features that make it easier to do certain kinds of things may slow down doing other things. Even when systems overlap in features, they typically have differing philosophies and different technical strengths, which means that certain kinds of work may be significantly easier in one system than another. For example,
developing something in Mathematica or Maple may be fine and quick for a personal computer, but making the same programming work on a supercomputer may take a lot of effort in a different language. Yet a work environment with multiple languages need not be overwhelmingly complex. Most systems with major development effort behind them (such as Maple and those mentioned in the "section above) have many similarities.

What makes things work out is this: at the introductory level, the difference between casual computing and professional technical computing is the style of working (higher emphasis on documentation, justification and ease of reuse), and the use of language-based commands/programming needed to do the more sophisticated operations in technical work. "Crossing over" to the professional mode of operation means getting over the hurdles of learning the new style of work, and learning how to interact with computers in a typical computer language. Once this hurdle is passed, it should take only incremental effort to acquire expertise in the second, third, or \( n \)th technical system. Having formal instruction on the first system should provide an explicit introduction into the concepts and the work processes to facilitate this.

Most systems realize that they cannot be the sole provider of technical computing services. If Institution A uses Mathematica and Institution B uses Python for their work, then if they expect to use each other's efforts, there has to be a way of interconnecting programs written in one system with that of another. Thus most systems have interconnections. For example, the Matlab Symbolic Toolkit allows Matlab users to call Maple to do formula manipulation. Similarly, Maple users can link to Matlab and run a Matlab program they've gotten from a colleague as part of a problem-solving process written in Maple.

Knowledge of basic programming and the concepts of software development make it possible to switch between systems with only a modest amount of additional effort. Software interconnection allows one to reuse programming done in another system without having to translate into another language. Symbolic computation systems like Maple also have the additional bonus of being able to translate some of their programming between languages. There is a "convert-to-Matlab" feature for example, or a "convert-to-C" feature for computations involving just numbers or text.
2 Getting started with Maple's Document Mode: doing technical work with a clickable interface

2.1 Chapter synopsis

1. How to start up Maple and perform simple calculations, algebra, and plots within it.
2. Detecting and fixing typographical mistakes.
3. Detecting mistakes caused by misunderstaplondings and incorrect decision making by the user.
4. How to save Maple work so that you can refer to it or resume working on it later.
5. How to recover a Maple worksheet if it or your computer crashes.

2.2 Starting up Maple, getting a fresh start

Start up the Maple application (this varies on the type of computer system you have, typically it involves clicking or double-clicking on the Maple 15 icon, but if you can't figure it out yourself ask for a demo for someone who knows). Once the Maple application window appears, a new document will appear in the main working area of the Maple application. A flashing cursor will appear with the outline of a small rectangle with dashed lines. The entry mode will read "Math" and "2D Math". You can make the "quick help" black box disappear by clicking on the "close box" circle-X in the upper right hand corner.

Table 2.1: Maple started up with new document in Windows XP

Click on close box
After you close the quick help box, you will see the Maple cursor in a small rectangle with a dashed line outline.

**Table 2.2: Maple document with first entry area**

![Calculation entry area](image)

At this point, what you type will appear in the small rectangle and be regarded as a mathematical expression. In the next section, we describe what to type in order to get something useful to happen.

### 2.3 Evaluating an expression involving exact arithmetic

**Grade school arithmetic**

In the math area, type $2 + 3$. "2 + 3" is regarded as a mathematical expression by Maple when you type it into the dashed rectangle. As you are typing, the input indicators should say that you are entering Math mode, in "2D Math" input, using "Times New Roman" font.

**Table 2.3: Maple input using 2d math**

![Maple input using 2d math](image)
This expression should show up in the work area. When you hit the `enter` key, then Maple will evaluate the expression. After the expression is evaluated, you should see the result displayed below the input, as in the figure below:

![Maple input with labeled result](image)

Maple has automatically calculated the answer and given it a label (1). After Maple calculates the answer, the cursor should appear below the result in another dashed-line rectangle. This indicates that Maple is ready to do another calculation.

Maple supports all the basic arithmetic operations in a fashion similar to many other programming languages. One thing that takes some getting used to is that * (asterisk) is used to input multiplication rather than "x" or a centered dot. Another thing that takes getting used to is that what you type is formatted to look like math notation. Thus if you type a /, Maple understands that you are talking about division and immediately starts formatting your input as if it were a fraction. If you type an asterisk, Maple formats that into a dot (⋅). There is also formatting that occurs with caret (^) since that is the way you enter an exponent (power) in Maple.

### Table 2.5: Arithmetic Operations in Maple

<table>
<thead>
<tr>
<th>Operation</th>
<th>Character to type/ character name</th>
<th>Notes</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>addition</td>
<td>+ (&quot;plus&quot;)</td>
<td></td>
<td>2 + 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(2.1)</td>
</tr>
<tr>
<td>multiplication</td>
<td>* (&quot;asterisk&quot;)</td>
<td>Typing an asterisk makes a center dot(⋅) appear in the displayed expression.</td>
<td>2·3</td>
</tr>
<tr>
<td>division</td>
<td>/ (&quot;slash&quot;)</td>
<td>Typing a slash draws a baseline and then positions the cursor in the denominator. Subsequent typing appears in the denominator. To get out of the denominator and return to normal typing, use the right-arrow key (→).</td>
<td>2/6</td>
</tr>
<tr>
<td><strong>Multiple divisions are by default conducted left-to-right.</strong></td>
<td>$\frac{2}{6} = \frac{5}{5}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Multiple subtractions are conducted leftmost first.</strong></td>
<td>$3 - 5 - 2$</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Use parentheses to change the order of calculation. They are also good for removing any guesswork by the reader as the order of operations.</strong></td>
<td>$(2 + 3) \cdot 5$</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Put a dash in front of a number or parenthesized expression to negate it.</strong></td>
<td>$-(3 - 5 - 2)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Typing a caret moves the cursor to the exponent position. Subsequent typing appears in the exponent. To get out of the exponent and return to the baseline, use the right-arrow key (→).</strong></td>
<td>$2^3$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$2^3 - 5$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$2^{2^{-1}}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$2^{-2} + \frac{4}{5}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(2.4)  
(2.5)  
(2.6)  
(2.7)  
(2.8)  
(2.9)  
(2.10)  
(2.11)  
(2.12)  
(2.13)
Let's try out some of these operations. We can enter a sum of two fractions by using "+" and "/". If we type "2/3→+5/6 enter", We intWthis is what we see:

\[
\frac{2}{3} + \frac{5}{6} \quad \Rightarrow \quad \frac{3}{2}
\]  

(2.17)

The way to get a fraction in is to type a slash (/). As soon as you do so, Maple draws an underscore and positions the cursor underneath the fraction line. The next characters you type appear as the denominator. If you type the "+" right after the "/", the plus will appear in the denominator which is permitted by Maple but not what we want in this situation. To get the "+" to appear outside of the fraction, we type the right arrow key (the key with → on it). This moves the cursor out of the fraction back into the baseline of the expression. Then we can enter the + for addition, and another fraction. After we hit the enter key, Maple will simplify the result into a single fraction with any common factors removed from the numerator and denominator.

Now let's do a multiplication. The Maple programming language (like most) uses an asterisk * as the symbol for multiplication. However, Maple displays the expression with a centered dot. This may be disconcerting -- what you type is not what you see. When you are in "math mode" Maple will be using fancy typography to display whatever math you are entering. We'll see more of this shortly. See if you can reproduce this result:

\[
2 \cdot 3 \quad \Rightarrow \quad 6
\]  

(2.18)

We can mix operations. Try to enter and calculate the following:

\[
1 + \frac{2}{3+4} + 5\cdot6 + 7 \quad \Rightarrow \quad \frac{67}{14}
\]  

(2.19)

In order to get that last denominator, we had to select the expression we had entered for the numerator with the mouse, so that the entire contents of the entry rectangle were blue. Then we typed a slash and the denominator appeared beneath it all.
An alternative to using the mouse to enter expression (1.2.18) would be to use parentheses. If we type 
\[(1+2/3+4\rightarrow+5*6+7)/8 \text{ enter}^*\] we will see this:

\[
\left(1 + \frac{2}{3 + 4} + 5 \cdot 6 + 7\right) \quad \frac{67}{14} \quad (2.20)
\]

This allows you to enter complicated expressions without having to use the mouse. Of course, the mouse is still necessary if you want to go back and edit.

We observe in passing that a distinctive feature of Maple is that Maple does exact arithmetic with integers and fractions. It keeps fractions as the ratio of two integers. It will, however, automatically simplify such ratios to lowest terms:

\[
\frac{2}{3} \cdot \frac{6}{7} = \frac{18}{21} = \frac{2}{3} \quad (2.21)
\]

**Diagnosing typographical mistakes**

Making mistakes is a normal part of using any tool, be it a computer or otherwise. You'll probably make as many mistakes learning how to use Maple as you would make when learning a new sport, a musical instrument, or when learning how to write a good essay. Some of them will be obvious as soon as you make them, others will be subtle or harder to figure out how to fix.

When you make some kinds of mistakes, the computer may give you an *error message*. For example, if you make a typo and Maple doesn't recognize what you enter as being a valid command, it will complain. Here are some typical error messages. At this stage, some of the messages will make sense and you can react appropriately. Others will use vocabulary that is unfamiliar. For those, the best thing to do is to have a clear idea of what you want to enter (by closely imitating examples that are known to work) and checking carefully that what you have typed was accurate.

**Table 2.6: Examples of Maple error messages**

<table>
<thead>
<tr>
<th>Error Message</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 +</td>
<td>Error, invalid sum/difference</td>
</tr>
<tr>
<td>2 + 4</td>
<td>6</td>
</tr>
<tr>
<td>+ 2 4</td>
<td>Error, missing operation</td>
</tr>
</tbody>
</table>
This time we mistakenly enter the expression with the symbols in the wrong order. Maple complains that it expects some operation to be entered between the two numbers but there isn't one.

```
. + 4
```

Error, invalid matrix/vector product

```
[+ 4]
```

We intended to enter "2+4" but typed a period instead of a 2 by mistake. Even though we are trying to do the same thing, the error message is different because a different symbol (the period) in this context suggests to Maple that we are trying to do linear algebra. The appropriate thing to do here is to correct the expression and hit enter again.

```
2 + 4
```

6

(2.26)

```
\[
\frac{3}{5 + 3 \cdot (}
\]
```

Error, unable to match delimiters

```
\frac{3}{5 + 3 \cdot (}
```

We started a sub-expression with a parentheses but forgot to finish it. In Maple, a delimiter refers to a parenthesis -- ( or ) -- a bracket [ or ], or a brace { or }. Delimiters are symbols that mark the beginning and end of an expression. In many instances they are necessary to unambiguously indicate meaning. For example, \(5 \cdot (3 + 5)\) evaluates to 40, whereas the expression without parentheses, 5 \(\cdot 3 + 5\), means 20 because multiplications are always done before additions unless the parentheses indicate otherwise.

```
\left( 3 + \left( \frac{5}{7} \cdot 5 \right) \right) \cdot 2
```

Error, unable to match delimiters

```
\left( 3 + \left( \frac{5}{7} \cdot 5 \right) \right) \cdot 2
```

This is another instance of the same mistake. We wanted to enter \(\left( 3 + \left( \frac{5}{7} \cdot 5 \right) \right) \cdot 2\) but forgot a parentheses.
We intended to enter "1+3" but typed the extra comma in by mistake. We get an error message that talks about sequences, a concept in Maple that we haven't discussed yet (that's coming in the next few chapters). Maple thinks that the sequence is "invalid" because it usually expects commas to be between items, such as "1,2,x".

Maple's language has many other elements in it that we haven't gotten to discuss yet. If you use any of them by mistake then you will see messages with vocabulary we haven't discussed yet.

It's a fairly typical experience for new users to see some messages that you won't be able to gather much intelligence from other than the fact that you made a mistake that you should fix,. With knowledge-rich systems such as Maple, you may blunder into sections of the system that you haven't learned yet. The best strategy is to back out of the situation by editing the expression so that it is exactly like something that is known to work.

This one is fairly obvious. In order to fix it though, we need to know what numerator we intended to enter. We can fix this by editing the expression with the missing numerator and hitting enter again, as below:

\[
2 + \frac{9}{3}
\]

(2.29)

Maplesoft, the producer of Maple, provides an on-line list of all the errors in Maple 15 at http://www.maplesoft.com/support/help/category.aspx?CID=2544. The left column of the page lists all messages. Clicking on a particular message link will produce an explanation that is often helpful in figuring out what has caused the message.

Correcting typographical mistakes

The standard procedure for fixing a mistake is as you would in a word processor: *edit the mistaken input* and *re-execute the computation*. Here are ways of doing this:

1. Using the mouse, position the cursor where the mistake is. Then use the backspace key to erase the characters you want to get rid of. Type in more characters to replace it.
2. Use the left arrow key (←) to back up. Typing after backing up then inserts the new typing at the point where the cursor was positioned.
3. Use the mouse or other "pointing device" of your computer to select a section of what you typed. New typing then replaces the selection of what you typed.
4. Use the mouse to select a region, then "cut", which you can do through the Maple menu Edit -> Cut. Of course most people use the keyboard shortcut for cutting, which for Windows or Linux is control-X while on the Mac it's command-X.
5. Copying and pasting (control/command-C and control/command-V) also works in Maple.
You may find that sometimes you attempt to create a 2D Math input area but Maple does not compute a result for the input after you hit the enter key. To create a "clickable math" input area if this happens, place the cursor where you want the input area to be, and use the Format->Create Document Block Menu item of the Maple window:

Table 2.7: Create Document Block to force a Math input area wherever the cursor is placed

Exponentiation (powers). Numbers with lots of digits

Use a caret ^ to specify an exponent (a "power"). In math mode, Maple will position the cursor so that the next things you enter will become the exponent. As with fractions, you can get out of the exponent by using the right arrow key → when you want to go back to non-exponent numbers. Try entering these expressions

\[ 2^3 \]

\[ 2^{1000} - 2 \]

(2.30)

We note that Maple does integer and fraction operations exactly. It will not introduce any rounding error into a computation as a calculator would when the answer requires more than ten decimal digits to write down.

There are limits to the number of digits Maple will use for integers or fractions, but they have to do more with limits of the computer
hardware and memory rather than a "pre-ordained" decision about how many digits might be useful to keep. If you type *kernel-opts*(maxdigits) into a Math input area on the worksheet, Maple will print out a number which is the maximum number of digits it can handle in any integer or fraction. On the author's computer in 2011,

\[ \text{kernelopts(maxdigits)} = 268435448 \]

This says that the largest number that can be computed by Maple has over 268 million digits. Even if we wanted to use the answer, we wouldn't want to look at

For example, Maple can compute the result of \( \frac{1}{52!} + \frac{2^{100}}{3^{27}} \) exactly (try it!).

Exact computation is useful not only for doing algebra but also for things such as computing probabilities through counting, or in deriving mathematical formulas for use in simulation or prediction programs.

**Detecting and fixing semantic and "logic" mistakes**

There will be other kinds of mistakes where there is no error message, but the response is not what you want. Sometimes this happens because you say something that while grammatical, means something entirely different from what you meant. This kind of mistake is called a *semantic error*. This could be as simple as entering "2-3" where you meant to type "2+3". Sometimes it is less obvious, such as mistyping the 23-rd digit of a 55 digit number or the 12th term of a long sum. Or worse yet, it could be *because you are using the incorrect vocabulary* so what you think you are saying does not have that meaning to the computer.

**Table 2.8: Example of a vocabulary mistake**

<table>
<thead>
<tr>
<th>2\times3 + 5</th>
<th>2\times3 + 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>2\times3 + 5</td>
<td>11</td>
</tr>
</tbody>
</table>

Suppose we were under the (mistaken) impression could use "x" in Maple to stand for multiplication. We might use the above as a way to calculate "two times three, plus five". However, while there is no error message, what is calculated is not the number we were expecting. The first step towards proceeding to fix this is to realize that it's *not what we want*. To Maple, this is a legitimate calculation -- you want to create a formula that's "two times the variable x3, plus five".

Knowing that the proper way to enter multiplication is through a palette, or symbol "*" (asterisk) as explained in XXXX.

| (3 + 5)(4 − 1) | 8           |
| (3 + 5)x(4 − 1) | 8x(3)      |
| (3 + 5)⋅(4 − 1) | 24         |

Maple allows you to write any of these things, but they mean different things to it. Maple's grammatical rules requires you to specify multiplication by use of the explicit mark * for multiplication, or the use of the \( a-b \) Palette item as described in the section "The Expression Palette"
later in this chapter. The use of "x" or just juxtaposition doesn't mean multiplication in Maple -- these are not errors, they mean something else other than multiplication (functional composition, to be precise, although for our purposes it's easier to just think "doesn't mean multiplication").

Finally, there are mistakes made because you ask Maple to do the wrong calculation. Even though Maple does what you want, it turns out that what you wanted was wrong!

For example, you may read a word problem and decide to solve the equation \(3 \cdot x + 2 = 6\), whose solution is \(x = 4/3\). But when you plug the solution into the circumstances of the problem, you find that it doesn't solve the word problem. The problem may be because your decision that \(3 \cdot x + 2 = 6\) was the equation when it should have been \(2 \cdot x + 4 = 6\). This is known as an "error in logic" or just a "logic error".

Sometimes it's obvious that you made a mistake because the answer is obviously wrong. For example, seeing "-1" as the answer to "how many eggs did the chickens lay in three days?" is obviously wrong since there's no such thing as "negative eggs". Sometimes you can plug your solution into the original situation to see if it works as it should.

### 2.4 Saving and retrieving your work

You can save your work in a Maple worksheet file through the File -> Save (keyboard shortcut: control/command-S) menu item of the Maple application. A dialog box appears allowing you to type in the name of the worksheet (we've typed in "test" into the box in the .)

The file is saved with a ".mw" suffix (e.g. test1.mw).

To retrieve the file in a subsequent Maple session, start up the Maple application as before and then use the File->Open menu item to initiate the dialog that retrieves the file.

Table 2.9: Maple save menu operation
2.5 Retrieving backups

The "state of Maple" display appears on the bottom left hand side of the Maple window. Usually this display says "Ready", which means that Maple is awaiting your next command. Sometimes it reads "Evaluating..." which indicates that Maple is actively computing an answer. Sometimes it says "Autosaving worksheet", which means that it is saving a copy of the present state of your open worksheets into temporary storage on your computer. The amount of time Maple spends autosaving becomes noticeable in longer Maple sessions when the worksheet contains a lot of results.

Should your computer suddenly lose power or should Maple crash, you can retrieve the last autosaved worksheet by selecting the File -> Recent Documents -> Restore Backup menu item. This will fill your Maple with copies of all the autosaved worksheets. You may then delete them or save them to permanent file space as you wish.

Table 2.11: The Maple state display
Algebra, plotting and mouse-clickable operations

Algebraic expressions and equations. Solving equations. Working with pieces of expressions.

We don't have to limit ourselves to just numerical calculation in Maple (even if the ability to use as many digits as we wish and exact fractions allows us to do arithmetic more like the math books do). Another distinctive feature of Maple is that you can do algebra by entering expressions with symbols -- the \( x, y, z, i, \) and \( n \) that we see in algebra books. Maple will automatically collect terms and do some simplifications for us automatically

\[
x^2 + 2 \cdot x + 5 + 3 \cdot x
\]

\[
x^2 + 5x + 5
\]  

(2.37)

We can even enter equations:

\[
\frac{3}{5} \cdot x + 1 = 4 - x
\]

\[
\frac{3}{5}x + 1 = 4 - x
\]  

(2.38)

\[
3 \cdot x + 1 + 4 \cdot x = a \cdot x + b
\]

\[
7x + 1 = ax + b
\]  

(2.39)

Note that while Maple automatically collected the \( x \) terms on the left hand side of the equation, it does not try to do the more interventionist operation of moving all the \( x \) terms to the same side of the equation.

Now, enter the following expression, but rather than hitting the enter key after you've entered it, do a right-click. On the Mac, instead of right-clicking hold the control key down then click on the mouse button (this is referred to as "control-click"). A menu of algebraic operations will pop up. Select Factor and see how Maple can factor the polynomial:

\[
x^2 + 5 \cdot x - 50 \xrightarrow{\text{factor}} (x + 10) \cdot (x - 5)
\]

Note that this line does not have a (XX) label for it.

To further demonstrate the right-click (control-click) operations available, enter the following equation. Right click on the expression and a pop-up menu should appear that includes a "solve" item. Select the solve and a submenu will appear where you can specify that you want to solve for \( x \).

\[
\frac{3}{5} \cdot x + 1 = 4 - x \xrightarrow{\text{solve for } x} \left[ x = \frac{15}{8} \right]
\]

For those with previous experience on other systems: some things are different, for a reason

(This section is for people who already know a programming language and are noticing that Maple does some things differently. Other readers can skip this section.)

Most mainstream programming languages work in a similar way. That is why learning Maple has value beyond just being able to use the Maple system -- once you learn Maple, learning Matlab or Java or C is a matter of incremental adjustment. Noticing the differences is educational in that it makes you become aware of the arbitrariness of some of features of languages. If you have used another programming language such as Java or Visual Basic (VB), you will notice that some things do not work the same in Maple as they do there. Some of this can be explained if you know something about the history of programming languages. Some of it is explained by being aware that the designers of Maple had different goals than those who invented Java or VB.)
One thing that you have undoubtedly noted is that symbols do not have to have a value associated with them. For example, in Java if you said

\[ k = 5; \]

Then if you were to create another expression in Java such as `System.out.println(k^2 + k + k + 3);` then "5" would be used as the value of \( k \) in the expression and you would end up printing 38. In Maple, you do not have to associate \( k \) with a numerical value before you use \( k \) in an algebraic expression. If there is no prior association, Maple just treats the expression as a formula with symbols in it. It may do some algebraic simplification on what you entered, but it does not need to get a number as a result. Since there was no prior assignment for the expressions in section 2.5.1, the calculation done with them just keeps the formula..

Another thing that is different is that in Maple ",=" is used for equations, not assignment. The operator in Maple corresponding to ",=" in Java or VB is ":=" (a colon immediately followed by an equals, with no spaces inbetween). In Maple, if we wanted to associate "5" with the symbol \( k \), then we would do:

\[
\begin{align*}
k &:= 5 \\
k^2 + k + 3 + k &
\end{align*}
\]

People who know more than one programming language have a better understanding of which features are change a lot between languages, (such as whether = or := is used for assignment), and which ones are fairly uniform (+ being used for addition, or the use of parentheses in functions and expressions).

Maple does not use ",=" for assignment because, being a mathematically oriented language that can handle algebra, it wants to make the entry of equations a natural thing. Its use of ",=" for assignment is a feature borrowed from the Algol/Algol 68/Pascal family of programming languages, which picked this operator to make it clear that the assignment operation is different from algebraic equality.

Is ",=" better than ",="? That's a kind of question that is about as hard to answer as trying to decide whether "uno" is better than "one". If one had a language where you had to do ",=*=*=%=###+++###" instead of ",=" or ",=" , you could be critical of the choice because it takes much more effort to enter a 35 character operator than a one or two character one. But the Algol-family choice of ",=" has reasonable motivation -- studies of novice programmers have shown that beginners using languages where ",=" is the assignment make more mistakes because they confuse its use in mathematics with its use in programming. Novices have been observed to write things like "5=k" which does not work as an assignment, even though mathematically the equations "k=5" and "5=k" mean the same thing.

Just as with architectural design of a building, each feature of a programming language is typically carefully considered. Many features are borrowed or copied from predecessor languages, where they have already been subjected to the test of many people using the feature. If you are not familiar with Algol family languages, you can see examples of them at various educational web sites, such as http://www.engin.umd.umich.edu/CIS/course.des/cis400/algol/average.html and http://portal.acm.org/citation.cfm?id=154766.155365.

**Plotting, approximate numerical solutions through cursor position**

The right-clickable interface can also activate a plot of a formula you may have entered. Enter this formula, then right-click and select **Plot -> 2d plot**. The automatic defaults for plotting this produce this result.
Table 2.12: Example of Plotting

\[ y = x^2 - 10x + 4 \]

![Graph of the function](image-url)
The user has clicked on the plot and positioned the cursor at the coordinate (-4.12, 61.60). The cursor was not captured by the screenshot although it is visible under ordinary use.

The 2DPlot operation makes pre-set decisions about the plot, such as the range of $x$ (-10 to 10), the color of the line, axes labelling, etc. Users can inject their own preferences about these things about this by selecting right-click -> Plot -> PlotBuilder and filling in the dialog box with their choices.

Table 2.14: User-configured plot using PlotBuilder instead of 2DPlot
Table 2.15: PlotBuilder Dialog box
The Expressions Palette and the Common Symbols Palette: entering Trigs, logs, roots, exponentials

It's possible to get the common functions of high school algebra, pre-calculus, and calculus by using the Expression palette of the Maple Window:

Table 2.16: The Expression palette

For example, to enter the square root of 36, click on the palette entry for $\sqrt{a}$. That expression will appear in the document, with the "a" selected. If you then type 36 on the keyboard, that number will replace the selected text. If you then hit the enter key, Maple will evaluate the expression and produce the exact result "6".

\[ x + y + \frac{1}{2} + \frac{1}{4} + \sqrt{36} \]

\[ x + y + \frac{27}{4} \quad (2.42) \]

You can use the palette multiple times, to create more complicated expressions. Just continue to the use mouse or the arrow keys to move around in the expression. Selecting and typing, backspacing or deleting are all ways of replacing or correcting pieces of the expression.

The Common symbols palette, two panels below the Expression palette, can be used to enter $\pi$ and $e$, the base of the natural logarithm system.

The palette does not have the inverse trig functions, so you have to enter them through typing. Their names are: \texttt{arcsin, arccos, arctan}, etc.
Approximate numerical (calculator-type) arithmetic in Maple

If you enter expressions with integers, exact fractions, and symbols such as π and \( e \), then Maple will perform exact calculations rather than give approximate answers as a conventional calculator would. You can get approximations by selecting the "numerically solve" instead of the "solve" option from the right-click pop-up menu.

### Examples of numerical computation

1. Enter fraction, select approximate->20 from right-click pop-up menu.

\[
\frac{47}{52} + \frac{4}{3} \quad \text{at 20 digits} \quad 2.2371794871794871795
\]

2. Enter exact expression, select approximate->5 from right click pop-up menu

\[
\sin\left(\frac{\pi}{10}\right) \quad \text{at 5 digits} \quad 0.30902
\]
3. Enter equation. Then solve->solve, then select Element->1, then right hand side, then approximate->10

\[
\sin\left(\sqrt{e^x}\right) = \frac{1}{3} \quad \text{solve} \quad \left\{ x = 2 \ln\left( \arcsin\left( \frac{1}{3} \right) \right) \right\} \quad \text{select entry 1} \quad \left\{ x = 2 \ln\left( \arcsin\left( \frac{1}{3} \right) \right) \right\} \quad \text{right hand side} \quad \left\{ 2 \ln\left( \arcsin\left( \frac{1}{3} \right) \right) \right\} \quad \text{at 10 digits} \quad -2.158578910
\]

**Evaluation, and selection of pieces.**

Sometimes you wish to **evaluate an expression for a particular value of a variable**. There is a right-click operation that does this.

**Table 2.19: Evaluate at a point**

<table>
<thead>
<tr>
<th>Expression</th>
<th>Evaluate at point</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x^2 - 2 \cdot a \cdot x = 0 )</td>
<td>( \frac{1}{4} - a = 0 )</td>
<td>Evaluate at point</td>
</tr>
<tr>
<td>( x^2 - 2 \cdot a \cdot x = 0 )</td>
<td>( x^2 - 6y^2 = 0 )</td>
<td>Evaluate at point</td>
</tr>
<tr>
<td>( 3 \cdot y + 5 )</td>
<td>( 14 )</td>
<td>Evaluate at point</td>
</tr>
</tbody>
</table>

Using the right-click menu, it's possible to select or extract a portion of an expression for further work.

**Table 2.20: Operations on equations, multi-part expressions**

<table>
<thead>
<tr>
<th>Right hand side, left hand side</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x = \frac{\sin(a)}{r^2 - 1} )</td>
<td>One of the options in the right-click menu is &quot;right hand side&quot;. It only works for equations.</td>
</tr>
<tr>
<td>( x = \frac{\sin(a)}{r^2 - 1} )</td>
<td>( x )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operations on multi-part expressions</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select entry</td>
<td>Solving this quadratic equation reveals that there are two solutions. Right-clicking on the solution and then selecting entry 1 gives the first solution, enclosed in brackets [ ]. Right-clicking on that and again selecting entry 1 gives the first solution, without the brackets. Right-clicking on that and selecting approximation gives a calculator approximation to the root.</td>
</tr>
</tbody>
</table>

| \( x^2 - 4 \cdot x = 4 \) | Solve for \( x \) \[ \{ x = 2 + 2 \sqrt{2}, x = 2 - 2 \sqrt{2} \} \] \{ x = 2 + 2 \sqrt{2} \} | \{ x = 2 + 2 \sqrt{2} \} |
| Select entry 1 | \( x = 2 + 2 \sqrt{2} \) |
| At 5 digits | \( x = 4.8284 \) |
## 2.7 A quick-reference summary to this chapter

<table>
<thead>
<tr>
<th>Example</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Arithmetic</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[2 + \frac{3^2}{4} - \frac{1}{6}]</td>
<td>Use +, *, -, /, ^ for arithmetic. Hitting the Enter key produces a labelled result.</td>
<td>2 D Math input mode displays the textbook-like version of what you input. Maple's simplification automatically combined fractions and places things in lowest terms.</td>
</tr>
<tr>
<td>5!</td>
<td>Use ! for factorial</td>
<td>Do you know what 5!! (double factorial) means?</td>
</tr>
</tbody>
</table>

### Example

2 + \(\frac{3^2}{4} - \frac{1}{6}\) = \(\frac{49}{12}\) (2.46)

### Example

Use ! for factorial

5! = 120 (2.47)

### Making mistakes

\[2 + \left(\frac{3}{5}\right)\] Error message mistakes (from typos or mistakes in intensions) The easiest ones to detect. You have to figure out what you are doing wrong, though. The error message may not always be helpful in advising you on this, although it often is.

A farmer plants a fence post every foot, for 1250 feet. At that point, he switches to planting the fence posts every 4.7 feet for another 940 feet. How many fence posts does he need in all?

\[
\frac{1250}{1} + \frac{940}{4.7} = 1450.000000
\] (2.48)

"Logic errors" Often you can find these kinds of mistakes by looking at simpler versions of the problem where the answer can be figured out with paper and pencil. For example, sit down and draw a picture of the posts and their positions when the initial distance is 10 feet, and the second distance is 9.8 feet. Count the posts in your picture. Then you "scale up" the calculation to handle the large situation.

It's easy to compute the wrong answer in these kind of "off by one" situation.

### Editing (fixing mistakes)

backspace, delete erase starting from current cursor selection
### Arrow keys
- → ← move cursor within current selection
- Select with mouse/type replaces selected text
- Cut, copy and paste of a selection works as it does with a text processor

### File saves, opens
- Save files with **File -> Save** or **File -> Save As**.
  - Open a saved file with **File -> Open**. Other File operations are similar to that of standard word processors.

### Functions and math symbols

\[
\sqrt{\csc \left( \frac{\pi}{2} \right) + e} = (1 + e)^{1/3}
\]  

(2.49)

- Insert math into an expression by using the Expression Palette. You can enter \( \pi \) using the Common Symbols Palette. \( e \) (the natural logarithm base) can also be entered this way. Note: typing \( e \) from the keyboard does not enter this symbol.

- Chapter 2 demonstrated the following functions and symbols:
  - Square roots, \( n \)-th roots
  - Natural logarithms
  - Trig functions: \( \sin, \cos \) (trig functions all use radians, not degrees)
  - Base 10 logarithms
  - \( \text{arcsin}, \text{arccos}, \text{arctan} \)
  - \( \pi, e \)

- If you are entering a function by the keyboard rather than the palette, you must enclose the function's argument in parentheses.

### Algebra

\[
\begin{align*}
x^2 - 2x - 15 &= 0 & \text{left hand side} \\
x^2 - 2x - 15 &= \text{factor} (x + 3) (x - 5) \\
\sin(x) &= 1 & \text{solve} \left[ x = \frac{1}{2} \pi \right] \\
\sin(x) &= 1 & \text{solve} 1.570796327 \\
x^2 - \arcsin(x)
\end{align*}
\]

- Right-click (control-click on Mac) on an entered expression to get the pop-up menu.
- Chapter 2 demonstrated examples of the following operations:
  - Factor
  - Solve
  - Solve numerically
  - Right hand side (of an equation)
  - Left hand side (of an equation)
  - Select (n-th part) of an expression
  - Approximate numerically (to 5, 10, 20, etc. digits' accuracy)
  - Plot (two dimensional)
  - Many plot options to determine range and domain of plot, color, captions, etc.
  - Evaluate at a point
  - Choose values for variables in an expression
The expression must be something that involves a single variable and will result in a number when a value is used for that variable. Thus you can’t do a 2d plot of \(x^2 - a\) because you wouldn’t get a number if you picked a value just for \(x\) (or just for \(a\)).

Maple uses defaults for the plot range, and the plot color.

Trying to plot an equation produces an implicit plot (see next appendix).

A dialog box appears that allows you the select from many more options, such as plot color, the line style, vertical and horizontal ranges, captions, etc.

Exact numbers in Maple have no decimal points.

Symbolic constants such as \(\pi\) and \(e\) entered from the Common Symbols Palette are also exact.

Numbers with decimal points in Maple cause arithmetic calculations to be done approximately.

Use of limited precision numbers in algebra (e.g. factoring, differentiation, solving) may not produce good results.

Use them in Maple only when an approximate result is desired.

Numbers like .25 or .6015 are limited precision. If you want exact algebra done, use 1/4 or 6015/10000, etc.
In very large calculations, limited precision calculations may be noticeably faster than those with exact arithmetic. Most of the time there isn’t an appreciable difference.

### Evaluate at a point

<table>
<thead>
<tr>
<th>Expression</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x^2 - 2\cdot x = 0 )</td>
<td>( \text{evaluate at point} \ 1 - a = 0 )</td>
</tr>
<tr>
<td>( x^2 - 6y^2 = 0 )</td>
<td>( \text{evaluate at point} \ 3\cdot y + 5 \rightarrow 14 )</td>
</tr>
</tbody>
</table>

### Operations on equations

- **right hand side, left hand side**
  
<table>
<thead>
<tr>
<th>Expression</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x = \frac{\sin(a)}{r^2 - 1} )</td>
<td>( \text{right hand side} \ \frac{\sin(a)}{r^2 - 1} )</td>
</tr>
<tr>
<td>( x = \frac{\sin(a)}{r^2 - 1} )</td>
<td>( \text{left hand side} \ x )</td>
</tr>
</tbody>
</table>

### Operations on multi-part expressions

- **select entry**
  
<table>
<thead>
<tr>
<th>Expression</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x^2 - 4\cdot x = 4 )</td>
<td>( \text{solve for } x )</td>
</tr>
<tr>
<td>( [x = 2 + 2\sqrt{2}, x = 2 - 2\sqrt{2}] )</td>
<td>( \text{select entry 1} )</td>
</tr>
<tr>
<td>( x = 2 + 2\sqrt{2} )</td>
<td>( \text{select entry 1} )</td>
</tr>
<tr>
<td>( x = 4.8284 )</td>
<td>( \text{at 5 digits} )</td>
</tr>
</tbody>
</table>

### Commentary

- **Enter an expression in a document, then right-click (control-click on Mac) followed by:**

  - **Operations on symbolic expressions**
    
    | Expression | Example |
    |------------|---------|
    | \( x^2 - 1 \) | \( \text{solve} \ (x = 1), (x = -1) \) |
solve->solve for a variable

\[ x^2 - 2\cdot a \cdot x = 0 \quad \text{solve for } x \quad \Rightarrow \quad \{x = 0\}, \{x = 2\cdot a\} \]

solve->numerically solve

\[ x = \cos(x) \quad \text{solve} \quad 0.7390851332 \]

The thing to try when there is a numerical answer but the exact solution is too complicated to understand or Maple can't find an exact solution.

Factoring

\[ x^2 - 1 \quad \text{factor} \quad (x - 1)\ (x + 1) \]

Factoring can simplify an expression sometimes. Factoring doesn't know the trig simplification rules, though.

\[ \cos(x)^2 - \sin(x)^2 \quad \text{factor} \quad (\cos(x) - \sin(x))\ (\cos(x) + \sin(x)) \]

Plots->2d plot

\[ x^2 - 1 \rightarrow \]

The expression must be something that involves a single variable and will result in a number when a value is used for that variable. Thus you can't do a 2d plot of \( x^2 - a \) because you wouldn't get a number if you picked a value just for \( x \) (or just for \( a \)).

Maple uses defaults for the plot range, and the plot color.
A dialog box appears that allows you the select from many more options, such as plot color, the line style, vertical and horizontal ranges, captions, etc.

<table>
<thead>
<tr>
<th>Operations on equations</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>right hand side, left hand side</td>
<td>$x = \frac{\sin(a)}{\rho^2 - 1}$ right hand side $\xrightarrow{\text{right hand side}} \frac{\sin(a)}{\rho^2 - 1}$  [x = \frac{\sin(a)}{\rho^2 - 1}] left hand side $\xrightarrow{\text{left hand side}} \sin(a)$  [x = \frac{\sin(a)}{\rho^2 - 1}] move to right, move to left $x^2 + x + 1 = a \xrightarrow{\text{move to right}} 0 = a - x^2 - x - 1$ This moves the entire side of an equation to the other side.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operations on constant expressions</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>approximate $\rightarrow$ 5 (or 10, 20, 50)</td>
<td>$\tan\left(\frac{\pi}{10}\right) \sqrt{\frac{1}{e^{10}}} \xrightarrow{\text{at 20 digits}} 0.34157868529293212152$ Uses approximation methods to turn constants into an approximate (decimal point) number with 5 (10, 20, 50) digits.</td>
</tr>
<tr>
<td></td>
<td>$x = \ln(5000!) \xrightarrow{\text{at 20 digits}} x = 37591.143508876766569$</td>
</tr>
</tbody>
</table>
3 Chapter 3 Technical word processing

3.1 Chapter Overview

In this chapter, we learn how to use Maple as a word processor. This allows us to write reports that combine technical writing with math formulae, calculated results, pictures, tables, etc. Many of the features are highly similar to Microsoft Word or similar WYSIWYG (what you see is what you get) word processors. The strength of Maple's word processing is that it makes it easy to enter technical formulae, and that the word processing and calculation can be done in the same document.

3.2 Maple as a word processor

Maple documents allow a mixture of text and mathematics. By default, Maple expects that when you position the cursor by clicking somewhere in the document, you will be entering math and be wanting it do to a calculation. The document is in what is called math entry mode.

You can tell whether the document is in math entry mode because the Math button on the Maple toolbar will be gray, and the "C" menu item says 2D Math.

Another mode of operation for Maple documents is text mode. When in text mode, Maple has the behavior of a word processor. It just shows what you typed. Hitting enter while you are in text mode just causes text entry to move to the next line. It does not cause any calculation to be done with what you typed.
You can switch to entering text in the following way:

1. Position the cursor at the spot where you want to enter text.

2. Click on the Text button on the Maple toolbar. This places the Maple document in text entry mode. Alternatively you can switch to Text mode by typing control-T (on Macintosh, command-T) or by using the Maple menu bar Insert->Text. You can tell when you've switched to text entry mode because the Text button will be gray, and the "C" menu item says Text.

3. With the keyboard, enter your verbiage. When you are in text mode, you will also see that the menu bar will be enabled for boldface, italics, underlining, left-/center-/right- justified text, colored text, and colored backgrounds, and bulleted/numbered text.

4. To switch back to math, click on the Math button on the Maple tool bar. Alternatively you can type control-R (on Macintosh, command-R), or use the Maple menu Insert->2-D Math.

Table 3.2: Document after control-T (or Insert->Text)

![Text button on Maple toolbar](image)

A Maple worksheet in text mode in OS X. Although it is hard to see, the cursor is positioned at top left of screen.

You can do mathematical word processing without any computation by switching between text and math modes, using the Palettes to help you enter the math. As long as you don't hit the return (enter) key, the math will not cause any calculation.

Table 3.3: Document with a mixture of text and math
The user typed the text, then went into math mode by typing control-R (command-R). They then entered the "F = G..." formula in math mode. The user got the subscripted $m$ by typing an underscore _ after the $m$ to get Maple to descend to subscripts, then used the right-arrow key to ascend back up to the main level of the expression. The other symbols in the midst of the rest of the narrative are entered in a similar way.

Richard saw a description of Newton’s law of gravitation in his physics supplemental textbook, *Stephen Hawking for Dummies*:

$$F = \frac{G \cdot m_1 \cdot m_2}{R},$$ where it was expected that $m_1$, $m_2$, and $R \geq 0$. Although he didn’t consider himself a strong physics student, he was glad that they hadn’t dumbed down the material so much that it had lost all the mathematics.

It is possible to mix text and the results of calculations in a paragraph. Typing control-= (command-=) when the cursor is in a math expression will cause Maple to print an "=" and then the result of evaluating the expression on the same line. This is an alternative to hitting the enter key and allows those kinds of calculations to be mixed with text.

Table 3.4: control-= puts the results of a calculation in the midst of text

The user typed the text, then did a control-R (command-R), then entered the math expression and then typed control-=. After the calculation result appeared, they typed control-T and entered the remainder of the sentence.
3.3 Shortcuts to entering math symbols

Using the Palettes, we can enter a wide variety of mathematics -- expressions, math symbols, Greek letters (using the Greek Palette), arrows, etc. There are additional Palettes not shown by default, which you can get by View → Palettes → Show All Palettes. However, you can enter many symbols in math mode from the keyboard through "shortcuts". Most of the shortcuts consists of typing the textual name of the symbol or some abbreviation of it, and then hitting the escape key -- the key labelled Esc on many keyboards.

For example, to enter the symbol ∞ while in math mode, you can type `infin` and then hit the escape key. A pop-up menu of choices will appear to allow you to complete entry of the symbol. With practice, this can be a faster way of entering ∞ than using the Palettes.

Table 3.5: Keyboard shortcuts in math mode through the escape key

<table>
<thead>
<tr>
<th>In math mode, we type <code>infin</code>.</th>
<th>After hitting the escape key, a menu of completions appears.</th>
<th>We pick the first alternative (either by hitting the return key or by operating the mouse to select the first option) and what we typed is replaced by the selection.</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
</tr>
</tbody>
</table>

Greek symbols can be entered by typing the romanized name of the letter, followed by escape. For example, in math mode, typing `omega` followed by escape produces ω. Typing Omega followed by escape produces Ω (the upper case version of the Greek letter).

A shortcut to entering the symbolic constant e (the base of the natural logarithm) is to type `e`, then hit the escape key, then return.

You can see a summary to all of the conveniences Maple offers through Help → Quick Reference.

3.4 Other word processor features

Inspection of the worksheet toolbar reveals many more word processing features: line justification, bold face and italics, numbered items, colored letters or backgrounds, font sizes, and font types. The Insert operation on the Maple toolbar allows creation of Tables and Images (graphics files). Rudimentary drawings can be inserted through Insert → Canvas. We encourage you to explore and make use of the features on your own.

3.5 Troubleshooting word processing

A phenomenon that you may encounter is not being able to switch back to math mode from text mode, even after performing the operation that should do so (clicking on the Text button of the document toolbar, typing control-T, performing Insert → 2D Math, etc. This may be due to the worksheet losing track of where you are in the document. A "sure-fire" cure for switching modes is to position the cursor at the point where you want to enter math, then do Format → Create Document Block. A dashed box will appear at the location of the cursor, indicating that it is again in math mode.
Tools→Spellcheck (alternatively, the F7 key) will run a spelling check on the non-math part of your document.

### 3.6 Summary of Chapter 3 material

<table>
<thead>
<tr>
<th>Name</th>
<th>Menu operation</th>
<th>Key short cut</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Important word processing operations in a Maple worksheet</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Switch entry to 2D Math mode</td>
<td>Insert→2D Math Click on &quot;Math&quot; oval in menu bar just below names of worksheets.</td>
<td>control-R (command-R on Mac)</td>
</tr>
<tr>
<td>Switch entry to Text mode</td>
<td>Insert→Text Click on &quot;Text&quot; oval in menu bar just below names of worksheets.</td>
<td>control-T (command-T)</td>
</tr>
<tr>
<td>Use a keyboard shortcut in Math mode</td>
<td></td>
<td>Type the shortcut, then hit the escape key. For example, typing <code>omega</code> and then escape will turn the text into $\omega$. Typing e and then escape will allow you to turn the text into the symbolic constant $e$ without needing the Expression Palette.</td>
</tr>
</tbody>
</table>
## 4 Chapter 4 Assignment

### 4.1 Chapter Overview

We learn how to label results with symbolic names through the operation of "assign to a name", sometimes called *assignment*. This allows us to reuse the results in subsequent steps of a multi-step calculation without re-entering or copying it. Assignment causes a portion of computer memory to store the computed result, so that it can subsequently be referred to by name. Reassignment changes what's stored.

The keyboard operation := provides a keyboard shortcut for assignment. := will be used heavily in later work in programming as we shift from mouse/menu operation to textual specification of calculations.

`unassign` causes Maple to forget the association of value with name. All assignments can be forgotten through `restart`.

### 4.2 Assignment: remembering results for future use

We can compute a result and label it with name. This action is called *assignment*. We can do this with the clickable menu by the action right-click (control-click on Macintosh) → *assign to a name*. A pop-up menu will appear asking us to fill in the name that we want to use.

Once we have assigned a result to a name, we can use the name, and Maple will use the assigned value.

A name can be any sequence of upper- or lower-case letters, digits and the underscore character _. It must start with a letter. Maple distinguishes between upper and lower case letters, so `result` and `Result` are considered different names.

In programming, the term *variable* is used interchangeably with *name*. Both refer to an identifier which the action of assignment associates with a computed result. Computer books often talk about "assigning the result to a variable" which means the same thing as "assigning the result to a name".

<table>
<thead>
<tr>
<th>Table 4.1: Assignment</th>
</tr>
</thead>
</table>
| \[
\sin\left(\frac{\pi}{4}\right) + 1 \\
\frac{1}{2} \sqrt{2} + 1
\] |
| assign to a name \(\text{trigResult}\) |
| After computing a result, we assign it to the name \(\text{trigResult}\) by clicking on the result, selecting the menu item *assign to a name* and then typing in the name. (4.1) |

| \[
\frac{1}{2} \left( \frac{1}{2} \sqrt{2} + 2 \right) \sqrt{2}
\] |
| \(\text{at 5 digits}\) |
| 1.9142 |
| (4.2) |

| \[
(\text{trigResult} + 1) - (\text{trigResult} - 1)
\] |
| We can then use the name instead of repeatedly entering or copying expressions. (4.3) |

| \[
\frac{1}{2} \left( \frac{1}{2} \sqrt{2} + 2 \right) \sqrt{2}
\] |
| (4.4) |
4.3 \( := \) is a keyboard shortcut for assignment

The Maple operator \( := \) (colon immediately followed by an equals) also performs assignment.

Table 4.2: \( := \) is another way to do assignment of a name

<table>
<thead>
<tr>
<th>Form</th>
<th>Examples</th>
<th>Commentary</th>
</tr>
</thead>
<tbody>
<tr>
<td>name ( := ) expression whose value will be assigned</td>
<td>( x := 5 )</td>
<td>Assigns the name ( x ) the value 5.</td>
</tr>
<tr>
<td>( poly := z + \frac{5}{2} + \sin \left( \frac{\pi}{3} \right) )</td>
<td>Assigns the name ( poly ) the value consisting of ( \frac{5}{2} + \sin \left( \frac{\pi}{3} \right) ) plus whatever the assigned value of ( z ) is. If no value has been assigned the name ( z ), then the result is the algebraic formula: ( z + \frac{5}{2} + \sin \left( \frac{\pi}{3} \right) ).</td>
<td></td>
</tr>
</tbody>
</table>

When you enter an expression in Math mode (even if it's just a name, without any arithmetic), you ask Maple to evaluate what you have entered. Maple also (before it does anything else) figures out the values assigned to names that appear in the expression. Then it does arithmetic, function calculation, and any other operations you've described in the expression.

The only exception to the "Maple always evaluates before assignment" occurs when with the name that appears on the left hand side of \( := \). That name is not evaluated, it literally is used as the name assigned to.

Assignment really requires two steps. The first is figuring out the result. The second is assigning the result to the name. The "figuring out the result" step is called evaluation. If any of the symbols in the expression being evaluated have an assigned value, then that value is used. If those values involve other symbols, those are in turn checked for values, etc.

Symbols without an assigned value have their own names as their value. This allows you to enter an expression such as \( x^2 + 2 \cdot x + 5 \) in \( x \) and use the \( x \)'s as symbols in the normal mathematical style as long as you don't assign \( x \) a value.

If you use a name/variable in an expression, and it has no assigned value, then Maple uses the rule that the value of an name with no assigned value is just the name itself.

To deassign a variable use \texttt{unassign}. For example, \texttt{unassign('x')} causes any value of \( x \) have no assigned value, just as if it had never been assigned. This does not undo the effects of other computations done while the assignment was in effect, as the following examples below illustrate.

Table 4.3: Assignment

<table>
<thead>
<tr>
<th>Examples of assignment with ( := )</th>
<th>Commentary</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p := x^2 + x + a \cdot x + 5 )</td>
<td>We assign the name ( p ) the value of the expression. Note that since ( x ) and ( a ) have not been assigned values, the results of evaluation just leaves them as symbols.</td>
</tr>
<tr>
<td>( x^2 + x + ax + 5 )</td>
<td>(4.5)</td>
</tr>
<tr>
<td>( p + 1 )</td>
<td>If we enter an expression containing ( p ), its value is plugged in for the calculation of the result.</td>
</tr>
<tr>
<td>( x^2 + x + ax + 6 )</td>
<td>(4.6)</td>
</tr>
<tr>
<td>( x := 3 )</td>
<td>Here we assign the name ( x ) the value 3.</td>
</tr>
<tr>
<td>( 3 )</td>
<td>(4.7)</td>
</tr>
</tbody>
</table>
### Examples of assignment with :=

<table>
<thead>
<tr>
<th>Assignment</th>
<th>Commentary</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p := 17 + 3a$</td>
<td>(4.8) If we now do a calculation with $p$, the value of $x$ is used since $p$'s value mentions $x$. There may be a chain of assignments that Maple must look at to evaluate an expression.</td>
</tr>
<tr>
<td>$x := 4$</td>
<td>(4.9) We can solve the result $1.3.4$ for $a$ by right clicking that expression.</td>
</tr>
<tr>
<td>$p := 25 + 4a$</td>
<td>(4.10) We change the value of $x$ by assigning it a different value.</td>
</tr>
<tr>
<td>$y := x$</td>
<td>(4.11) When we do another calculation with $p$, the most recent assigned value of $x$ is used.</td>
</tr>
<tr>
<td>$unassign('x')$</td>
<td>(4.12) This a way of assigning 4 -- the current value of $x$ to the name $y$.</td>
</tr>
<tr>
<td>$p := x^2 + x + ax + 5$</td>
<td>(4.13) We can undo the connection between $x$ and any value by unassigning $x$. This operation produces no output, so no label. We can barely tell that it has happened. The quote marks surrounding the $x$ -- <code>'x'</code> are mandatory, otherwise $x$ would be replaced by its value and Maple would try to unassign the corresponding value rather than $x$ itself.</td>
</tr>
<tr>
<td>$y := 4$</td>
<td>(4.14) $p$ still has a value, but since $x$ no longer has a value, we are back to the original result.</td>
</tr>
</tbody>
</table>

### 4.4 The Variables Panel: a way to know the current value of your variables

The operation of assignment uses part of the computer's memory to remember an association of a name with a result. If you are doing several assignments, you can see the current state of the Maple session -- the associations of names with values currently in effect -- by using the Variables Panel. It will list all of the variables assigned, and their present value.

A useful way to think about assignment is to think of the computer creating a memory slot containing the result, labeled with the name being assigned to.
Table 4.4: The variables panel shows currently assigned variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>$x^2 + x + a$</td>
</tr>
</tbody>
</table>

After assignment, the Variables Panel shows that the variable $p$ is assigned an expression.

If we then assign $x$ a value, then there are two variables listed. Notice that the value of $p$ is unchanged.

Changing the value of $x$ doesn't change the value of $p$ either. However, in evaluating $p$ (which is what happens when you enter “p” into the worksheet, Maple will always use the current value of whatever symbols are used. If the symbol has no assigned value, then the name of the symbol is its value.

If we unassign a variable, then we can think...
4.5 Differences between the appearance of the worksheet and the state of a Maple session

The state of a Maple session consists of all the variables that are currently assigned, and what their values are. The state changes every time we do another assignment or unassignment. Only an assignment operation (or an unassignment) can change the state of the session. The Variables Panel displays the state.

Typically work in a Maple document progresses in a top-to-bottom fashion -- additions are placed at the end of whatever is already there. This is also true for an ordinary word processor such as Microsoft Word. However, a key feature of a word processor is that if you need to go back and fix something, you can do so. In a Maple worksheet, if you go back and change a line. However, while this will change the instructions, Maple will not automatically update the results due to the changes. You have to hit enter on the lines you want to be re-calculated.

When you save a Maple worksheet as described by Saving and retrieving your work (page 18) does not save the state of assigned variables. This means that when you reopen a saved worksheet, no variables are assigned. This can be confirmed by looking at the Variables Panel after reopening a worksheet, as illustrated by the "The state of a Maple Session" example, below. When you reopen a saved worksheet, it's not "what you see is what you get" -- at least, not right away.

To repeat: a saved worksheet does not save the state of the session (the variable assignments). Opening a saved worksheet file does not cause it to automatically execute the operations in the worksheet. This gives you a chance to edit the worksheet to change the calculations specified, before carrying out the instructions.

We know that we can jump back and execute a line in the worksheet a second time, just by positioning the cursor there and hitting enter (return). Thus the way that the document looks is not an accurate reflection of variable assignments that currently exist in the session.

The Variable panel of the Maple worksheet indicates which variables are assigned, and what their assigned values are. Use it to determine the current state if you are not sure what it is.

Table 4.5: The state of a Maple Session

| Example |
|-----------------
|                      |
We took the work of the previous section and saved it as a worksheet named Start1.mw. We then closed the worksheet. In a fresh Maple, we opened up the worksheet again. Note that while all the work we did is displayed, the Variable Panel indicates that no variables are assigned when the worksheet is read in.

When the Start1.mw, the cursor is positioned in the first line. If we hit enter, we re-execute the first line again. Note that the Variable Panel now has the result of the first line -- assigning $p$ a value.
Example

After hitting enter the first time, the cursor moves to the second line -- "x := 3". Hitting enter again, causes the second assignment to occur.

If we now go back and position the cursor on the first line and hit enter again, p is assigned a value that reflects that fact that x has an assigned value.

If we then execute the rest of the lines of the worksheet by hitting enter some more. But the results are not the same as in the previous example. This because the state of variables was different when we executed the first line the second time. In general, re-evaluation of an expression will yield different results if the variables it mentions have different values than the first time it was evaluated.

4.6 restart causes all variables to be deassigned

We've seen that it's possible to erase an assignment using unassign. If you want to forget all assignments we've made so far, then you can use restart. This can be useful in situations where you've done some work and made some assignments, but now want to switch to working on a different problem and would like Maple to forget about the assignments you made before. It is generally a good idea to restart at the beginning of unrelated sections of the worksheet just in case variables were previously assigned values that might not be related to their use in the new section.
Table 4.6: Evaluation of expressions involving assigned variables

Examples of assignment with :=

We saved a worksheet similar to that of the previous example, but we placed a `restart` instruction at the beginning of it. We executed all the instructions of the worksheet, in order, by hitting the "!!" icon on Maple toolbar. Note that only \( p \) is assigned a value at the conclusion because we unassigned \( x \).

We got the worksheet to look like this by executing the \( x := 3 \) line, then every line starting with \( p := x^2 + x + a \cdot x \) (but not the `restart`). The effect of executing the assignment to \( p \) once \( x \) had a value causes \( p \) to be assigned the result of evaluating the expression with \( x \)'s value as 3: \( 12 + 3 \cdot a \). Since this formula no longer has any \( x \) in it, reassigning \( x \) to have a value of 4 doesn't really change the result of evaluating \( p \), unlike the previous examples.
However, if we re-execute the entire worksheet, including the restart, we have results identical to what we had when things were fresh. restart removes any inadvertent side-effects that old assignments may have on evaluation.

### 4.7 Evaluation and chains of assignments

Evaluation always uses the current assigned value of a variable. Evaluation will follow the chain of assignments to find values that does not mention any assigned values.

Table 4.7: BROKEN - MISSING TITLE!

<table>
<thead>
<tr>
<th>Examples of assignment with :=</th>
<th>Commentary</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>We start fresh through restart..</td>
</tr>
<tr>
<td></td>
<td>We assign the name $p$ the value of the expression, then $x$. When we assign result2, it evaluates $p$, and finds it to have the value $x^2 + x + a \cdot x$. But $x$'s value is 3 and $a$'s value is $y$. It uses 3 and $y$ in the evaluation. This is why result2's assigned value is not the same as that of $p$.</td>
</tr>
</tbody>
</table>
Examples of assignment with :=

<table>
<thead>
<tr>
<th>Commentary</th>
</tr>
</thead>
<tbody>
<tr>
<td>If we then ( a := z+1 ), then Variable Panel shows that the value of ( a ) has changed, but the values of ( p ), result2, and ( x ) have not. However, if we ask Maple to evaluate ( p ), the evaluation uses the value of the symbols mentioned in the value of ( p ). This is why the evaluation of ( p ) is different -- ( 15 + 3 \cdot z ), even though the assigned value of ( p ) continues to be what it was originally.</td>
</tr>
</tbody>
</table>

4.8 Troubleshooting assignments

Equations are not the same as assignment

Assignment is an operation that many programming languages have. In some languages (e.g. Maple, Pascal, Eiffel, MuPad) \( := \) is used for the assignment operation. In others (C, Java, Matlab) \( = \) is used as the symbol for assignment. Maple uses \( := \) because it uses \( = \) for equations. It would be confusing to computers and to human readers to use the same symbol for two common but different operations in a single language.

\( = \) and \( := \) mean different things in Maple

Example

We assign \( a \) the value 3. However, for the next line we type "\( x = 4 \)". This is an equation and it doesn't assign \( x \) any value. The Variable Panel shows this because it lists \( a \) as the only variable that has an assigned value.
We edit the \( x=4 \) line into \( x:=4 \), and hit the enter key. This time the assignment goes through.

The name to be assigned always goes on the left hand side of the :=

Since \( 5 = x \) and \( x = 5 \) mean the same thing as mathematical equations, some people think that this should mean that \( x := 5 \) and \( 5 := x \) should both assign the value 5 to \( x \). However, only \( x := 5 \) does the assignment.

Table 4.8: Assignment := is not symmetric

<table>
<thead>
<tr>
<th>5 := x</th>
<th>Error, illegal use of an object as a name</th>
</tr>
</thead>
</table>

This doesn't mean anything to Maple. The name is supposed to be on the left hand side.

\[ x := 5 \]

This assigns \( x \) the value 5.

\[ z := y \]

This assigns \( z \) the (symbolic) value \( y \). It doesn't assign \( y \) any value.

To undo all assignments, use \texttt{restart}

Sometimes you want Maple to forget all the assignments you have made in a session. You can get this to happen either by using \texttt{unassign} on each assigned name, or by entering \texttt{restart} in Math mode and then hitting enter. This will unassign everything, undoing all the assignments.
restart does not erase the worksheet, however. The worksheet still looks the same, including the written record of the assignments you had previously done. What the restart does is to delete all the slots you have set up in your mental model.

**Some names are already used by Maple. You will get an error message if you try to assign to them yourself.**

When you first start up Maple, the names that you would ordinarily think of using to assign to are not assigned. However a few are, such as the symbolic constants \( \pi \) and \( i \). Maple will tell you that such names are reserved for system use. You need to pick another name.

<table>
<thead>
<tr>
<th>Table 4.9: Some Names can't be assigned to</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example</strong></td>
</tr>
<tr>
<td>( \text{Pi} := 47 )</td>
</tr>
<tr>
<td>Error, attempting to assign to 'Pi' which is protected</td>
</tr>
<tr>
<td>( \text{for} := 3.1 )</td>
</tr>
<tr>
<td>Error, controlling variable of for loop must be a name</td>
</tr>
<tr>
<td>( \text{solve} := x + 1 )</td>
</tr>
<tr>
<td>Error, attempting to assign to 'solve' which is protected</td>
</tr>
</tbody>
</table>

**Can't solve for a variable with an assigned value**

The solve operation takes an equation, and solves with respect to a variable. If that variable becomes assigned a numeric value, then solving again won't work because evaluation causes Maple to see only the number, not the symbol being solved for. Problems occur with other Maple operations that need a named variable, such as plotting.

<table>
<thead>
<tr>
<th>Table 4.10: Can't solve for a variable with an assigned value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>restart</strong></td>
</tr>
<tr>
<td>( eqn := x^2 - 5x + 6 = 0 )</td>
</tr>
<tr>
<td>( x^2 - 5x + 6 = 0 )</td>
</tr>
<tr>
<td>( (x = 3), (x = 2) )</td>
</tr>
<tr>
<td>( x := 3 )</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>( x^2 + 5x + 6 = 0 )</td>
</tr>
<tr>
<td>30 = 0</td>
</tr>
</tbody>
</table>
This is because the equation is evaluated before the right-click menu is activated, producing the equation $30=0$. Since $30=0$ has no variables, Solve isn't permitted since it makes no sense.

If we do a restart and then enter the equation, we can solve it because $x$'s assignment is gone.

\[\text{restart}\]

\[x^2 + 5x + 6 = 0\]

\[\text{solve}\]

\[\{x = -2\}, \{x = -3\}\]

if we assign $x$ a value, then plotting the expression produces a straight line $30$. This is because the expression is evaluated (producing the constant $30$) before it is handed to the plot operation. Plotting a constant is what produces the straight line.
### 4.9 Summary of Chapter 4 material

**Assignment**

<table>
<thead>
<tr>
<th>General form</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assignment is performed by using the assign to a name operation of the clickable menu.</td>
<td>$x := 5$</td>
</tr>
<tr>
<td>Assignment also be performed by typing in the name, followed by $\Rightarrow$, followed by the expression whose value will be the result to be assigned.</td>
<td>$y := z + \frac{x^2}{2}$</td>
</tr>
</tbody>
</table>

**Unassignment**

<table>
<thead>
<tr>
<th>General form</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>unassign('symbol name ')</td>
<td>$x + 1$</td>
</tr>
</tbody>
</table>

(4.29)  
(4.30)  
(4.31)  
(4.32)  
(4.33)
Undoes all assignments made by the user in the session so far.

### The state of the Maple session

The Variable Panel indicates the variables that are currently assigned in the Maple session, and their assigned values.
5 Chapter 5 Building scripts

5.1 Chapter Overview

We briefly discuss a few extra concepts useful with solve: how to use a combination of relations rather than just a single equation, and how to take apart or combine by the various forms of solve.

We then explore the concept of script: a sequence of operations useful for solving a problem. Having a script makes it easy to solve different versions of a problem by editing the parameter values and re-executing the script. Often, the justification of the effort to program a computation is driven by its reuse -- repeating the same computation with slight alterations each time. A frequently recurring scenario is a parameterized computation: 1) use variables to assign values to the parameters and 2) have subsequent steps of the computation refer to the parametric variables. Maple is well-equipped for reuse of parameterized scripts, since it has an !!! button on the Maple menu that will execute in sequence all the operations in a worksheet. Re-execution can also be specified by Edit → Execute → Selection or Worksheet.

5.2 Getting information from solve, data structures

The result of the solve operation can have multiple parts. When there are multiple solutions to the equation, the result of solve is a sequence, list, or set of solutions. We can select a part by giving an index (1 or 2, typically).

<table>
<thead>
<tr>
<th>Table 5.1: Selecting parts of a solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>We enter the equation, then right-click on it and select the solve item from the menu. This produces a sequence of two solutions. We right-click on that and pick the Select Element menu item to get the submenu that allows us to pick the first entry. Selecting the first element of the set gives us the equation. Selecting the right hand side of the equation gives us the first root.</td>
</tr>
<tr>
<td>3·x = x^2 - 28</td>
</tr>
<tr>
<td>select entry 1</td>
</tr>
<tr>
<td>3·x = 28</td>
</tr>
<tr>
<td>If we give solve a linear equation, it has only one solution. We can still select the first entry from the set, and then the right hand side of that to get the root.</td>
</tr>
<tr>
<td>3·x = x^2 - 28</td>
</tr>
<tr>
<td>[x = -4]</td>
</tr>
</tbody>
</table>

The right-click menu also offers the solve for variable item, which allows us to solve for x explicitly. The answer comes back in a slightly different form, but it still has parts that we can select from.

Maple (as well as many other programming languages) can compute with objects that have structure. Here are four different kinds of structures that Maple can handle:

<table>
<thead>
<tr>
<th>Table 5.2: Basic data structures in Maple and operations to extract parts of them</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type of structure</td>
</tr>
<tr>
<td>Equations, inequalities</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
### Type of structure

<table>
<thead>
<tr>
<th>What they look like</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Values or expressions separated by a comma</td>
<td>[ x ] (5.4)</td>
</tr>
<tr>
<td>A sequence surrounded by curly braces { }</td>
<td>{3, 7, 10} (5.8)</td>
</tr>
<tr>
<td>A sequence surrounded by square brackets [ ]</td>
<td>[ x + y = 3, y - 2x = 37 ] (5.10)</td>
</tr>
</tbody>
</table>

### Sequences

Values or expressions separated by a comma

- \[ 19, 47, 92 \] (5.5)
- \[ x^2 - 2 = 0, 0 < x \] (5.6)
- \[ x^2 - 2 = 0, 0 < x \] (5.6)
- \[ \text{select entry 1} \]
- \[ x^2 - 2 = 0 \] (5.7)

### Sets

A sequence surrounded by curly braces \{ \}

- \{3, 7, 10\} (5.8)
- \{3, 7, 10\} (5.8)
- \[ \text{select entry 3} \]
- \[ 10 \] (5.9)

### Lists

A sequence surrounded by square brackets [ ]

- \[ x + y = 3, y - 2x = 37 \] (5.10)
- \[ x + y = 3, y - 2x = 37 \] (5.10)
- \[ \text{solve} \]
- \[ \left\{ x = \frac{34}{3}, y = \frac{43}{3} \right\} \] (5.11)

Note that the result of this `solve` is a set of solutions.

For the time being, we just want you to recognize the different kinds of structures that are output by `solve` and other functions and be able to select parts from them. Later on we will get a lot of work done by performing more sophisticated operations with them.

### 5.3 Finding simultaneous solutions, constraining solutions.

Suppose we want to solve the system of equations \[ x + y = 5 \] and \[ -3y + 7 = x \]. This means finding values of \( x \) and \( y \) that simultaneously satisfy both equations. We can do this in Maple by typing in the first equation and then the second, separated by a comma. This is called entering a sequence of equations. Right-clicking (control-click on Macintosh) on the sequence will allow you to solve the system.

In Lab 1, you discovered that `solve` could also handle inequalities as well as equalities. You can enter a sequence of equations and inequalities to `solve`. This can be used to limit solutions to a particular range of values.

#### Table 5.3: Solving simultaneous equations

| \( x + y = 5, -3y + 7 = x \) | The result of this `solve` is a set of solutions. |
| \( x + y = 5, -3y + 7 = x \) | (5.12) |

The result of this `solve` is a set of solutions.
We assign an equation to \( p \), and solve it. Solving this equation produces 4 roots. Two of them are complex numbers (since they have \( i \) in them). The others are real.

\[
p := x^4 + 3x^2 - 57.5 = 0
\]

We enter a sequence consisting of the equation named \( p \), and the inequality \( x \geq 0 \). When ask Maple to find values of \( x \) that satisfy both the equation and the inequality simultaneously -- roots of the equation \( p \) that are non-negative. We might be interested in only non-negative real solutions, if, for example, we were using the equation to model a physical situation and \( x \) was measuring weight. Negative values for weight wouldn't be a solution we'd be interested in.

5.4 Scripting: making computational work easy to reuse

Consider the problem you did in Lab 1, along with a solution:

**Version 1 and solution**


A breeding group of 20 bighorn sheep is released in a protected area in Colorado. It is expected that with careful management the number of sheep, \( N \), after \( t \) years will be given by the formula:

\[
N = \frac{220}{1 + 10 \cdot (0.83)^t}
\]

and that the sheep population will be able to maintain itself without further supervision once the population reaches a size of 80.

(a) Graph \( N \) versus \( t \).

(b) How many years must the state of Colorado maintain a program to care for the sheep? Show your work.

(c) How many bighorn sheep can the environment in the protected area support? (hint: examine the graph for large values of \( t \).)

**Solution to (a)**

We can enter the equation from the problem, but we really need the right hand side of it because the *plot* operation can't plot equations, it can only plot things that evaluate to a number (rather than an equation).
Once we have the expression for the sheep population, we need to play with the plotting ranges to see when the leveling off occurs. We’d have to think about it and experiment a bit -- but the computer makes the replotting easy to do once we make our decisions about what to try.

\[ N = \frac{220}{1 + 10 \cdot (0.83)^t} \]

\[ \text{right hand side} \quad \frac{220}{1 + 10 \cdot 0.83^t} \quad \text{assign to a name} \quad \text{sheepPopExpr} \]

Solution to (b)

\[ 80 = \text{sheepPopExpr} \]

\[ 80 = \frac{220}{1 + 10 \cdot 0.83^t} \]

\[ \text{solve} \]

\[ \{ t = 9.354227718 \} \]

We can read the leveling off point from the plot, assuming that we have figured out the appropriate time range in (b). Alternatively, we can do a little calculus and take the limit of the expression as \( t \) goes to infinity.

\[ \lim_{t \to \infty} \text{sheepPopExpr} \]

\[ 220. \]
We can imagine ourselves working as an environmental engineer for the National Forest Service and being very pleased with ourselves for solving the problem with Maple. But now we are handed two more wildlife management problems to do, from other regions in our territory:

**Version 2**

A breeding group of 33 bighorn sheep is released in a protected area in Colorado. It is expected that with careful management the number of sheep, \( N \), after \( t \) years will be given by the formula:

\[
N = \frac{330}{1 + 10 \cdot (0.79)^t}
\]

and that the sheep population will be able to maintain itself without further supervision once the population reaches a size of 85.

(a) Graph \( N \) versus \( t \).
(b) How many years must the state of Colorado maintain a program to care for the sheep? Show your work.
(c) How many bighorn sheep can the environment in the protected area support? (hint: examine the graph for large values of \( t \)).

**Version 3**

A breeding group of 45 bighorn sheep is released in a protected area in Colorado. It is expected that with careful management the number of sheep, \( N \), after \( t \) years will be given by the formula:

\[
N = \frac{450}{1 + 10 \cdot (0.63)^t}
\]

and that the sheep population will be able to maintain itself without further supervision once the population reaches a size of 90.

(a) Graph \( N \) versus \( t \).
(b) How many years must the state of Montana maintain a program to care for the sheep? Show your work.
(c) How many bighorn sheep can the environment in the protected area support? (hint: examine the graph for large values of \( t \)).

We have the feeling that we will shortly be handed problems for a number of other locations as well. How can we reuse our original work with minimal effort?

If we had the first solution, we could produce the second solution through *copy-paste-edit-re-execute*:

Table 5.4: Executing a clone of a script through copy-paste

1. Copy and paste the original solution into a new document, or to a spot later in the same document. You do this as with a word processor -- select a region of the worksheet with the mouse, type control-C (command-C on a Mac). Then position the document cursor in the destination, and type control-V (command-V on a Mac). A duplicate of the solution should appear there.
2. Using the mouse, edit the formulae as needed. You only need to edit the black (input) formulae, not the blue ones.

3. Position the mouse at the first computation and hit enter. Continue to work your way through the sequence of the commands.

4. Alternatively, select the entire region containing the edited version of the solution and hit Edit→Execute→Selection.

5. If the region to be executed is the entire worksheet, then you can hit the !!! button of the Maple toolbar, or you can do Edit→Execute→Worksheet.

The results of executing the edited script are not totally correct. We will have to change a few things by hand: the clickable operation "evaluate at a point" in step (b) will use N=80 (which is what the copy says) instead of N=85, so we will have to redo it. Also, the last plot will not show an appropriate vertical range unless we do that. We'd like to do a little less of this hand-tuning in re-execution. Also, we'd like to make it easier to remember what to change between multiple versions.

To see the edited version of the script look at A breeding group (page 64).

5.5 Rewriting the script by assigning parameters at the start of the script

While copying and editing is probably a little faster than typing in the whole script again, we would also spend time hunt around for assignments to edit each time we wanted to re-run the script. We can reduce the "hunting around" by adopting the following working rule: identify the parameters of the problem, and change the script so that it assigns values to all parameters at the beginning of the problem.

This does not mean "put all assignments at the beginning of your script". Only the assignments involving parameters -- values that are specified by the problem description -- need to be relocated to the start of the script.

Finding and naming parameters

First, solve at least one version of the problem. Then, imagine what would need to be changed if you were trying to solve alternative versions of the problem. You can find parameters if you have several versions of a problem by looking at what changes in the worksheet from version to version.

For example, in the sheep problem, we note the following things changing in different versions of the problem. We pick names for these.

1. the numerator of the "sheep equation" (\( P \))
2. the coefficient in the denominator of the equation (\( c \))
3. the value of the stable population (\( s \))

The other number in the script is the original population of the sheep. We realize that it can be derived by evaluating the "sheep equation" at \( t=0 \). Thus this value does not need to be a parameter as we can derive it from the other information. It might be a good check though.

We then write the script to assign values to the parameters at the start of the script, and then write the other operations and expressions in terms of the symbols.

This allows us to redo the script just by changing the values at the beginning of the script. We use the word processing features to add extra directions to make the script easier to use, basically saying "here, change these things", and "this is the end of where you should stop copying".

To see the edited version of the script look at Attachment: Version 2 of Sheep Script, with parameters (page 65).
Having created this script, we can handle the third version of the problem by editing the values of the parameters and executing the script again by selecting the entire script with the mouse, and then doing Edit->Execute->Selection. We get this result:

### 5.6 Summary of script writing

Script writing is appropriate when you expect to handle several different versions of the same problem. In professional work, this is often the case -- if it's worth doing at all, it's probably going to occur more than once.

*Figure out how to solve the problem first. Then write the script.* There's really not much point in writing the script if you don't have some idea of the sequence of operations in it.

Once you have a worksheet of instructions for solving one version of the problem, look at it and the other versions and find the parameters. Set up a new worksheet where the first thing you do is to assign values to the parameters. Then work through the rest of the instructions and rewrite them to use the parameters instead of the fixed values from one version of the problem.

### 5.7 Troubleshooting scripts

Programming books have the tendency to show things that work. They say less about what to do about the things that don't work. However, it is usually the case that computer users often spend more time "getting things to work" than "working". Learning how to get out of jams is at least as valuable as knowing how to enter operations and what they mean. We offer this advice, which is "commonsense about work" as applied to interactive software development:

1. Solve one version of the problem before you try to start scripting. You can use Maple to experiment -- enter and edit snippets of operations that try out the solution technique for part of the problem. Eventually edit them together so that they solve the whole problem. If you have only fuzzy notions about the math or the operations you want to proceed, your computer work will just amplify that. Having a worksheet that solves one version of the problem can remove a lot of the fuzziness.

   Where does the inspiration for solving the problem come from? If you are lucky, the solution may be told to you. Or you may find a description of a similar problem as a starter. But the big bucks, as they say, go to those who can devise the solution plan themselves.

2. Limit each step so that it is a small step. If you get into trouble, you will be able to nip it at the bud. This becomes a more prominent tactic in later work when we are tempted to construct long-winded one-line expressions that do everything at once.

3. Test pieces individually, then put them together. For example, if you don't really understand how to make a dotted line plot in teal with a title, you should try to do that with a simple plot (such as $x^2$) rather than an expression whose shape you aren't that familiar with it. Then take what works and substitute the real expression you want to plot in a copy of the $x^2$ plot operation. The complexity of troubleshooting is reduced if you have fewer lines of untrustworthy code to look through for problems. This can be called divide and conquer troubleshooting.

4. If what you have doesn't work, find something similar which does work, and then incrementally edit it. For example, if you can't get $a := x^2 + 3 \cdot x + 1$ in, then first see whether you can get $a := 1$ to work. Once you succeed with that, edit the expression to $a := 3 \cdot x + 1$ and so forth.

5. If strange things continue to happen despite your best efforts to troubleshoot, it may be that previous settings in the Maple session are interfering with your current work. Recall that some people in the ENGR101 lab couldn't get their oscilloscopes to work because of settings changed in the oscilloscopes by groups earlier in the day. This can be particularly true if you are developing scripts and are assigning parameter values, then switch to development of another script in the same document. The values you assigned will not magically unassign themselves when you start working on something new in the same worksheet.

The remedy for this is to put a restart in as the first operation in your script, then re-execute the worksheet.
5.8 Attachments

Attachment: Version 2 of sheep script without parameters

Version 2 of sheep problem, with edited script

A breeding group of 30 bighorn sheep is released in a protected area in Colorado. It is expected that with careful management the number of sheep, $N$, after $t$ years will be given by the formula:

$$N = \frac{330}{1 + 10 \cdot (0.79)^t}$$

and that the sheep population will be able to maintain itself without further supervision once the population reaches a size of 85.

(a) Graph $N$ versus $t$.

(b) How many years must the state of Colorado maintain a program to care for the sheep? Show your work.

(c) How many bighorn sheep can the environment in the protected area support? (hint: examine the graph for large values of $t$.)

Solution to (a)

$$N = \frac{330}{1 + 10 \cdot (0.79)^t} \quad N = \frac{330}{1 + 10 \cdot 0.79^t} \quad \text{right hand side} \quad N = \frac{330}{1 + 10 \cdot 0.79^t} \quad \text{assign to a name} \quad \text{sheepPopExpr}$$

$$\text{sheepPopExpr}$$

$$\frac{330}{1 + 10 \cdot 0.79^t}$$

$$\rightarrow$$

![Graph of # of sheep versus time](image)
Solution to (b)

\[ 85 = \frac{330}{1 + 10 \cdot 0.79^t} \]  
\[ \text{slove} \]
\[ \{t = 5.277302835\} \]

We can read the leveling off point from the plot, assuming that we have figured out the appropriate time range in (b). Alternatively, we can do a little calculus and take the limit of the expression as \( t \) goes to infinity.

\[ \lim_{t \to \infty} \frac{330}{1 + 10 \cdot 0.79^t} = 330. \]

**Attachment: Version 2 of Sheep Script, with parameters**

*Version 2, with use of parameters*

Start of parameters -- change these for each version of the problem

\[ P := 330 \]
\[ 330 \]
\[ c := 0.79 \]
\[ 0.79 \]

We call the size of the stable population \( s \).

\[ s := 85 \]
\[ 85 \]

**End of parameters**

(a) Graph \( N \) versus \( t \).

(b) How many years must the state of Colorado maintain a program to care for the sheep? Show your work.

(c) How many bighorn sheep can the environment in the protected area support? (hint: examine the graph for large values of \( t \).)

(a)

\[ \text{sheepPopExpr} := \frac{P}{1 + 10 \cdot (c)^t} \]

\[ \frac{450}{1 + 10 \cdot 0.83^t} \]
Note that $sheepPopExpr$ is not a parameter since assignment is always the same for all versions of the problem.

To make the graphing work all the time, we set the vertical axis to "P+30" rather than a fixed value. This will set the vertical axis so that it will be 30 more than the value of P which from the first problem we have realized is the top of the graph.

$s = sheepPopExpr$

\[
\frac{330}{1 + 10^{0.79^t}}
\]

(5.32)

This is the time in years that model predicts it will take for the sheep population to reach self-sustaining status, allowing the wildlife managers to move onto another job.

(c) A little thinking reveals that the leveling off value (as indicated from the graph), is the numerator of $sheepPopExpr$, since the denominator goes to 1 as t goes to infinity.

\[
\lim_{t \to \infty} sheepPopExpr = 330.
\]

(5.34)
Attachment: Version 3 of Sheep Script, with parameters

Start of parameters -- change these for each version of the problem

\[ P := 450 \]  \hspace{1cm} (5.35)

\[ c := 0.83 \]  \hspace{1cm} (5.36)

\[ \text{sheepEquation} := N = \frac{P}{1 + 10 \cdot (c)^t} \]

\[ N = \frac{450}{1 + 10 \cdot 0.83^t} \]  \hspace{1cm} (5.37)

We call the size of the stable population \( s \).

\[ s := 100 \]  \hspace{1cm} (5.38)

End of parameters

(a) Graph \( N \) versus \( t \).

(b) How many years must the state of Colorado maintain a program to care for the sheep? Show your work.

(c) How many bighorn sheep can the environment in the protected area support? (hint: examine the graph for large values of \( t \).)

(a)

\[ \text{sheepPopExpr} := \frac{P}{1 + 10 \cdot (c)^t} \]  \hspace{1cm} (5.39)

Note that \( \text{sheepPopExpr} \) is not a parameter since assignment is always the same for all versions of the problem.

To make the graphing work all the time, we set the vertical axis to "P+30" rather than a fixed value. This will set the vertical axis so that it will be 30 more than the value of \( P \) which from the first problem we have realized is the top of the graph.

\[ \text{sheepPopExpr} \]

\[ \frac{450}{1 + 10 \cdot 0.83^t} \]  \hspace{1cm} (5.40)

→
This is the time in years that model predicts it will take for the sheep population to reach self-sustaining status, allowing the wildlife managers to move onto another job.

(c) A little thinking reveals that the leveling off value (as indicated from the graph), is the numerator of $sheepPopExpr$, since the denominator goes to 1 as $t$ goes to infinity.

\[
\lim_{t \to \infty} sheepPopExpr = 450.
\]

**End of script**

### 5.9 Summary of Chapter 5 material

<table>
<thead>
<tr>
<th>Type of structure</th>
<th>What they look like</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic data structures in Maple</td>
<td>Expression related by $=$, $&gt;$, $&lt;$, $\geq$, $\leq$, or $\neq$.</td>
<td>$x + y = 0$&lt;br&gt;$x + y = 0$&lt;br&gt;$x^2 - 3x &gt; 4$&lt;br&gt;$4 &lt; x^2 - 3x$&lt;br&gt;(5.43) (5.44)</td>
</tr>
<tr>
<td>Type of structure</td>
<td>What they look like</td>
<td>Examples</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------</td>
<td>----------</td>
</tr>
</tbody>
</table>
| Sequences         | Values separated by a comma | 19, 47, 92  
|                   |                     | 19, 47, 92 (5.45) |
| Lists             | A sequence surrounded by square brackets [ ] | MyData := [1.0, x, 3/4, a]  
|                   |                     | [1.0, x, 3/4, a] (5.46) |
| Sets              | A sequence surrounded by curly braces { } | Scores := {3, 7, 3, 10}  
|                   |                     | {3, 7, 10} (5.47) |

**Solving simultaneous equations**

\[ x + y = 5, \; -3y + 7 = x \]

\[ x + y = 5, \; -3y + 7 = x \]

\[ \text{solve} \]

\[ \{x = 4, y = 1\} \]

The result of this *solve* is a set of solutions.

\[ p := x^4 + 3x^2 - 57.5 = 0 \]

\[ x^4 + 3x^2 - 57.5 = 0 \]

\[ \text{solve} \]

\[ \{x = 3.0380606341\}, \; \{x = -3.0380606341\}, \; \{x = 2.495959218\}, \; \{x = -2.495959218\} \]

Solving this equation produces 4 roots. Two of them are complex numbers (since they have \( i \) in them) the others are real.

\[ p, x \geq 0 \]

\[ x^4 + 3x^2 - 57.5 = 0, \; 0 \leq x \]

\[ \text{solve} \]

\[ \{x = 2.495959218\} \]

This gets the solver to list only the positive real solutions. We might be interested in only non-negative real solutions, if, for example, we were using the equation to model a physical situation and \( x \) was measuring weight. In this case the other values of \( x \) wouldn't be relevant to our situation.

**Scripts**

Creating a script for a problem | In a Maple worksheet, take a version of a problem and create a sequence of operations in the worksheet that solve it.
Note similarities and differences between different versions of the problem. Envision what you'd have to change in the worksheet in order to solve a different version of the problem, and what would stay the same. You may have to rewrite some of the expressions to refer to the parameter rather than the value.

Assign the parameters at the beginning of the script. Rework the rest of the script so that the formulas refer to the parameters by name, rather than the values used in the original version of the problem.

For example, if the value 42 appears in several places in your script, define a parameter \( p := 42 \) at the start of the script and edit the other occurrences of 42 to be \( p \) instead. When you have a different version of the problem, you can edit just the single line \( p := 42 \) into say \( p := 47 \) and won't need to edit any other lines of the script.

<table>
<thead>
<tr>
<th>Using a script</th>
<th>Copy and paste the script to a new location.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Edit the assignments to reflect the new version of the problem.</td>
</tr>
<tr>
<td></td>
<td>Click on the !!! button of the Maple toolbar to execute the entire worksheet. Or select a portion of it with a mouse and hit Edit-&gt;Execute-&gt;Selection. As a last resort, just hit enter (return on Macintosh) multiple times to perform the operations in the new version of the script.</td>
</tr>
</tbody>
</table>

| Rationale for using scripts | More work to do than clickable interface the first time. Saves time if you expect to want to reuse the operations on multiple versions of the same problem. Also it is less error prone. |
| Divide and conquer troubleshooting | Test each piece of the script as soon as soon as you build it, rather than waiting to test it only after you have entered it all. It's easier to test incrementally, a piece at a time, because if there are fewer lines of code to suspect of causing problems. |
6 Chapter 6 More sophisticated scripting: textual entry, multiplots, point plots, character strings, lists, solving equations, and evaluation

6.1 Chapter Overview

We introduce textual entry of `solve` and `plot` operations, where the operation is specified by keyboard alone, without the use of the mouse or palettes. Textual entry is often preferred by programmers because it is easier to edit scripts written with text. We will retain use of the clickable interface for doing quick one-time calculations, or for developing ideas the first time before we start script-writing.

We begin to introduce additional concepts in Maple, to enhance what we can solve and plot:

1. In Maple, a character string is a collection of characters delimited by "s: "This is a string." We see how strings are used in the textual entry of labels and colors in plots.

2. We discuss lists e.g. [1, 2, x, 3.5]. We first saw them in Getting information from `solve`, data structures (page 57) provide a way of organizing multiple results in a single "data container", making it to operate on the whole collection of results while retaining the ability to getting at individual results from within the collection. Lists are used in both `solve` and `plot`.

3. `solve` uses two other types of data containers: sequences and sets. We describe how to recognize them, and how to extract information from them.

4. Programmers rely on the on-line documentation to manage the complexity of remembering the details of a knowledge-intensive system such as Maple. They learn/remember how to use a feature by looking up the description, finding an example close to what is desired, and then actively experiment with the example in a fresh worksheet. Reading without experimentation is usually not very productive.

5. In Chapter 4 Assignment (page 41), we explained assignment and how Maple uses assigned values whenever it sees a name in an expression. We introduce the `eval` operation, which allows variables to take on values during evaluation without assignment. Another way to view `eval` is that it is a way for assignments to be done temporarily and immediately forgotten. This can be an attractive alternative if you are concerned about situations where you'd be making many assignments and then undoing them through `unassign`. `eval` allows you to evaluate an expression for a particular value of a variable in one line, rather than having to type in the assignment and unassignment as well.

6.2 Textual entry of operations

The textual form of an operation in Maple has the general form:

```
operationName( sequence of values )
```

The `operationName` can be something like `solve` or `plot`. By `sequence of values`, we mean one or more items, each item separated from the next by a comma. "Sequence" here is the same kind of sequence that was first seen in the previous section on `solve` (page 58).

Maple will evaluate what you enter in the same way that was described for mathematical expressions `possible` (page 37). If there are assigned variables mentioned in the sequence, then their values will be used. If Maple knows the `operationName` (e.g. `sin`, `solve`, `plot`), then it will perform the calculation specified by the built-in programming. Otherwise, the result will be more or less what you typed in.

The technical term for the "values" in this situation is actual parameter or argument. Note that the parentheses around the sequence of values are mandatory -- you will either get an error or a result that's far from what you want if you omit the parentheses.
This style of writing things is sometimes called functional notation. In mathematics examples of functional notation are $f(x)$ or $g(3,5)$. In these examples, the name of the operation is the function name $f$ or $g$, while the actual parameters or arguments would be $x$ or the sequence $3,5$. Evaluating the function is what causes Maple to perform the action.

**Table 6.1: The textual form of equation solver solve**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{solve}(x = 3x^2 - 2, x)$</td>
<td>The form of the answer returned by the textual version is slightly different from invoking solve through the clickable interface. The former is typically a form that is easier to work with in scripts.</td>
</tr>
<tr>
<td>$x = 3x^2 - 2 \quad \rightarrow \quad \text{solve} \quad {x = \frac{-2}{3}, {x = 1}$</td>
<td>(6.1)</td>
</tr>
<tr>
<td>$\text{solve} \quad x = 3x^2 - 2, x$</td>
<td>In this, the operation Name (or function name) is solve. There are two arguments. The first argument is the equation $x = 3x^2 - 2$. The second argument is the symbol $x$. In this situation the result of performing the solve operation is a sequence of two numbers, the two roots of the equation.</td>
</tr>
<tr>
<td>$\text{solve} \quad (x = 3x^2 - 2, x)$</td>
<td>What you get if you leave out the parentheses -- not much! Because you left out the parentheses, Maple does not think you are asking for any function to be evaluated, which is how the work gets done.</td>
</tr>
<tr>
<td>$\text{error, unable to match delimiters}$</td>
<td>You get another one of those &quot;unable to match delimiters&quot; messages if you forget one of the parentheses.</td>
</tr>
<tr>
<td>$\text{solve} \quad (x = 3x^2 - 2, x)$</td>
<td>This is what happens if we forget to use the right arrow key to descend from the exponent of $x^2$. Maple thinks that we are talking about $x$ to a power called $2-2,x$, which understandably it doesn't make sense to it as a power.</td>
</tr>
</tbody>
</table>
### Table 6.2: The textual version of plotting

<table>
<thead>
<tr>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>plot(x - 3*x^2 - 2, x = -3 .. 3)</code></td>
<td>This is a textual form of plot. The first argument is an expression, the second argument is an equation naming a variable and a range of a plot. Note that if we wanted to change the range from -3..3 to -5..2 then we would just edit that line of the worksheet and hit enter again. If we wanted to redo the plot in the clickable interface, we would have to right-click and enter all the information all over again.</td>
</tr>
<tr>
<td><code>plot(x - 3*x^2 - 2, x)</code></td>
<td>If the second argument is just the variable, then <code>plot</code> uses default values for the range.</td>
</tr>
<tr>
<td><code>plot(x - 3*x^2 - 2, x = -3 .. 3)</code></td>
<td>No plotting happens if we forget the mandatory parentheses.</td>
</tr>
</tbody>
</table>

(6.5)
Suppose we entered this instead of `plot(x - 3 \cdot x^2 - 2, x = -3.3)` . There is no error message, but the picture is not like the first plot. Why did we get something different?

The only way that we would discover that a mistake is if we already had an idea of what the graph should look like, and noticed that the result differed significantly from what we expected. If you haven't formed a basis for expectations, you won't discover such mistakes.

Once you realize that the picture is incorrect, you would be spurred to search for the cause. Since there are no error messages and a plot (albeit a weird one) was produced, the most likely cause is that the arguments to the plot function are wrong. The obvious place to look for correct examples is the on-line documentation. If you look at the on-line documentation for help, you will see that giving plot three arguments means something different -- the third argument can be taken as the value of the vertical range of a plot. Evidently what is happening is that you are seeing only the tiniest top slice of the plot produced above because of the inadvertent specification of the vertical range.

This example illustrates the fact that just because there is no error message, it does not mean that all is correct. The computer needs to be told what to do in a way that it can follow. The rules are typically more restrictive than those of ordinary human language.

An attachment at the end of the chapter shows the textual form of common functions, subscripts. These textual forms can be entered from the keyboard wherever the palette entry would work.

### 6.3 Why are there two different styles for entering operations?

The clickable interface is a good way to get a calculation done quickly, but the actions specified in this way are hard to edit when building scripts. Maple, like most languages, has a textual version of all operations it performs. The editing involved in scripting development can often be easier to do on the textual version. In other words, the clickable interface is good for a one-time calculation, but not so good for the editing and re-execution involved in script reuse.

Another advantage of the textual mode of operation is that the number and variety of operations available in textual form is far greater than what's available in the clickable interface. Maple has several thousand operations. Building a clickable interface to all of them would result in tedious navigation through menus that would either be huge or involve many sublevels.

The downside of using the textual entry is that the developer must spell the text correctly, with the right number and placement of parentheses. Experienced users find that the textual interface is faster to deal with for scripting, while the clickable interface is faster for short, more casual use. Fortunately, in either case one can edit failed attempts and retry, so perfect entry is not necessary to be productive.

Becoming proficient with textual entry of operations is part of the transition technical users make in going from reuse of other's work to routinely creating their own programming. Without such proficiency, it is hard to realize the full power of the computer in modeling and simulation situations.
6.4 Plotting a list of expressions (multi-plots), plotting lists of numbers (point plots)

Recall that lists in Maple are a way of collecting expressions together into a single object, as discussed in the previous table describing lists and other data containers (page 58). You specify a list by listing the items in the list, enclosed in square brackets [].

If the first argument to plot is a list of expressions, then plot will on a single graph display the plots of all the expressions in the collection. By default, each one will be displayed in a different color.

Table 6.3: Plotting of multiple expressions

<table>
<thead>
<tr>
<th>Example</th>
<th>Plot two expressions on a range where both have comparable-sized results. We use the list ([x^2, \sin(x^2)]) to indicate the two expressions that should be plotted.</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\text{plot}([x^2, \sin(x^2)], x = 0..4))</td>
<td></td>
</tr>
</tbody>
</table>

![Plot of two expressions](image-url)
Example

Problem: approximately where is the expression $-3x^2 + x - 2$ equal to 5? While we could use solve to tell us exactly, it's often worthwhile to draw a picture and process the situation visually.

If we give plot the two expressions "$-5$" and $-3x^2 + x - 2$ then plot will plot not only the parabola (the second expression, it will also plot the expression that is always -5 for any value of x. This corresponds to the horizontal line drawn on the plot.

Visually we can see that the parabola is -5 at roughly -.8 and 1.2. We can even get a little more precise by clicking on the plot and using the coordinate window as was discussed in *Plotting, approximate numerical solutions through cursor position* (page 21).

Plotting multiple expressions simultaneously can be useful when you want to compare them. Assuming that the scales are comparable, one can get a sense of similarity or dissimilarity "at a glance".

We can plot data points rather than smooth curves, if we give the textual form of plot separate lists of x and y coordinates. can be used with the textual version of plot. If we give the textual version of the plot operation two lists of numbers that have the same length, then plot will regard the first list as a list of x-coordinates, and the second list as corresponding y-coordinates. If you provide plot with the third argument style=point, then it will produce a point plot. Otherwise, it will draw lines connecting each point.

Table 6.4: Plotting points with lists of numbers

<table>
<thead>
<tr>
<th>xList := [1, 2, 3, 4]</th>
<th>yList := [5, 6, 7, -1]</th>
<th>We use the textual form of plot to plot the points (1,5), (2,6), (3,7) and (4,-1).</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1, 2, 3, 4]</td>
<td>[5, 6, 7, -1]</td>
<td>(6.6)</td>
</tr>
<tr>
<td>(6.6)</td>
<td></td>
<td>(6.7)</td>
</tr>
</tbody>
</table>
Without the third argument, `plot` will try to connect the points with a curve.

Maple cares about whether things are capitalized or not. `Style` is not the same as `style`. 
6.5 Character Strings and putting titles and labels into plots

A string in Maple is something enclosed in double-quotes: "red", "this is a string?", "Gregory+Brothers+++++4++++++" are all strings. The double-quote symbol is mandatory for a string. Single-quotes ‘ (also known as apostrophes), backquotes ` (also known as acute accent marks) are not substitutes for double-quotes in writing Maple strings. Characters enclosed by apostrophes or backquotes mean something different to Maple. Use of the wrong punctuation marks will lead to undesired results or error messages.

In addition to being used to input data points in plot, lists also can be used in specifying colors and axis labels.

**Table 6.5: Plot options and labels**

<table>
<thead>
<tr>
<th>Plot</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>plot([sin(x), sin(x/2), sin(2*x)], x=-4..4, color=[&quot;Red&quot;, &quot;Green&quot;, &quot;Blue&quot;]</code></td>
<td>If one of the arguments to the plot operation is of the form <code>color = list of color names</code> then those colors will be used. Most reasonable names will work, but the full list can be seen in on line help (search for <code>colornames</code>). Note that we are using the textual version of the symbolic constant π. If you can remember how to spell it, it can be easier than selecting it from the Common Symbols Palette.</td>
</tr>
<tr>
<td><code>plot([sin(x), sin(x/2), sin(2*x)], x=-4..4, color=&quot;Red&quot;, &quot;Green&quot;, &quot;Blue&quot;)</code></td>
<td>Note that because the expressions being plotted are given in a set, the color assignment is not in the same order that they were typed in. If we wanted the same order, we should give the plot expressions in a list.</td>
</tr>
<tr>
<td><code>plot([sin(x), sin(x/2), sin(2*x)], x=-4..4, color=['Red', 'Green', 'Blue'])</code></td>
<td>Forgetting brackets for the list of colors.</td>
</tr>
<tr>
<td><code>plot([sin(x), sin(x/2), sin(2*x)], x=-4..4, color=[&quot;Red&quot;, &quot;Green&quot;, &quot;Blue&quot;]</code></td>
<td>One of the proficiency issues with textual input is that you have to remember all the ( ) parentheses and [ ] brackets. Can you find the the missing delimiter(s)?</td>
</tr>
</tbody>
</table>
If you leave enough delimiters out, you get error messages that don't complain about missing delimiters. You have to figure out what the problem is, which might involve a missing parentheses even if the message doesn't say so.

The "Error, (in sin)" is a cue that you should look at the places where you included \( \sin \) in your text and inspect it for problems. It doesn't take too much effort for you to notice that there's no finishing parentheses in the first \( \sin(x) \).

One reason why there was no error message about delimiters is that there are multiple missing parentheses. Because there are equal numbers of missing left and right parentheses, there was no alarm for missing delimiters.

The problem with this plot is that \( \pi \) doesn't mean the same thing to Maple as \( \Pi \). Maple is case-sensitive. Only \( \Pi \) means the symbolic math constant having to do with the circumference of a circle.

If one of the arguments to the \( \text{plot} \) operation is of the form \( \text{labels} = \text{list of axes titles} \) then those will be used. Each title needs to be a string.

In subsequent work, we will see strings used in other situations within Maple other than for plot titles.

### 6.6 Troubleshooting with strings

The most common mistakes with strings is to leave out the delimiting "s, or to use the wrong kind of delimiters. While the similar-looking keyboard characters ' (single quote or apostrophe), and ' (acute accent or backquote) look like would be equivalent, they are use for other purposes in Maple.
Table 6.6: What happens when you forget to use the " delimiter in strong, or use the wrong character for the delimiter

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>plot(3.5*x^2 - 2, x = 1..5, labels = ['temperature (in degrees C)', 'pressure in kilopascals')]</code></td>
<td>Forgetting to include &quot;s around one of the titles gives a cryptic error message about an &quot;invalid in&quot;. If you were an experienced Maple user, you'd know that in is part of the Maple programming language, but including &quot;in&quot; within a string would never be flagged as an error. This would be a clue that there's something wrong around where you entered the first label.</td>
</tr>
<tr>
<td>Error, invalid in</td>
<td>The message is not very helpful about telling you how to fix the mistake, though. This unfortunately is typical in most computer programming languages, despite several decades' effort in building systems software to help people program.</td>
</tr>
<tr>
<td><code>plot(3.5*x^2 - 2, x = 1..5, labels = ['temperature (in degrees C)', 'pressure in kilopascals'])</code></td>
<td>Putting the wrong kind of quote -- ' instead of &quot; didn't make a string. We got the same indication of a problem as before even though the problem is &quot;wrong kind of quote&quot; rather than &quot;no quote&quot;.</td>
</tr>
</tbody>
</table>

6.7 Learning through on-line documentation and experimentation

All the options available in the Plot Builder available through the right-click (control-click) interface are also available in the textual version of plot. In fact, there are many additional options and varieties of plotting available. The way to find out what the features are and how to invoke them is to consult the on-line documentation.
We can find out more about the textual forms of plotting by invoking Help -> Maple Help and typing plot into the search field. When we do so, we see the information in the figure below:

Table 6.7: Plot command help
We scroll to the bottom of the page and find an example of this. We are looking for a version of plots where v1 and v2 are lists. We don't see something exactly like that but we do see something with Vectors which are similar. Since the document says this should work for lists or vectors, we take the example and see if we can modify it for our own purposes:

**Table 6.8: Examples of plot**

Evidently, the first list is the values of the $x$ (horizontal) coordinates, and the second list the values of the $y$ (vertical) coordinate. We copy and paste the example into a Maple worksheet and then see if we can get it to work.
According to the documentation, we should be able to get this to work if the first two arguments are lists or vectors. So we edit the example to do lists instead and re-execute the line to see if it works in the same way.

To learn about plot options such as colors and labels, we click on the plot,options item under the search results for plot (see green oval in the figure). Clicking on that item produces this information. We see information about color (with another link to see colors),
along with possibilities, for labels, symbols, styles, etc. Again, the way to learn the options is through copying and pasting the examples into a fresh worksheet, getting them to work, and then modifying them to suit your own purposes.

Table 6.11: Plot option help page

| Plot Options |
| Description |
| This option should be provided to constraints that generate 2-D plots. These options can be used with the plot command and are generally available to all Maple commands that generate 2-D plots. The help page for a particular command provides more detail about the plotting options it accepts. |

Options

- **adaptive = true or false**
  - When plotting a function over an interval, the command plots at a number of points, connected by sample and parametric. Adaptive plotting, when necessary, subdivides these intervals to attempt to get a better representation of the function. This subdivision is halted off by setting the adaptive option to false. By default, this option is set to true, and intervals are subdivided at most 6 times to try to improve the plot. By setting this option to a non-negative integer, you can control the maximum number of times that subdivisions are divided.

- **axes = [type, style, frame, subframe, normal]**
  - Specifies the type of axes, one of: boxed, frame, none, or normal.
  - **type**
    - For the labels on the tick marks of the axes, specified in the same manner as for the plot.
  - **style**
    - Specifies information about the x-axis and y-axis. The first form axis = style applies the information given to both axes. The second form allows the information to be specified for a single axis, with each taking the value 1 to mean a 2-D axis or a 3-D axis. For details, see **plot**.
  - **frame**
    - The coordinates system used for display of the axes. The value c can be either polar or cartesian. By default, Cartesian axes are displayed. If it is polar, then radial and angular axes are generated. This option is used together with the coords=polar option.
  - **subframe**
    - Specifies information about the subframe. The coordinate system used for display of the axes. The value c can be either polar or cartesian. By default, Cartesian axes are displayed. If it is polar, then radial and angular axes are generated. This option is used together with the coords=polar option.
  - **caption**
    - The caption for the plot. The value c can be an arbitrary expression. For information on how to specify c, see **plot**.
  - **captionfont**
    - The font for the caption. The value c can be an arbitrary expression. For information on how to specify c, see **plot**.
  - **captionstyle**
    - The style for the caption. The value c can be an arbitrary expression. For information on how to specify c, see **plot**.
  - **captioncolor**
    - The color for the caption. The value c can be an arbitrary expression. For information on how to specify c, see **plot**.
  - **captionfontsize**
    - The size for the caption. The value c can be an arbitrary expression. For information on how to specify c, see **plot**.
  - **captionorientation**
    - The orientation for the caption. The value c can be an arbitrary expression. For information on how to specify c, see **plot**.
  - **captionfontfamily**
    - The font family for the caption. The value c can be an arbitrary expression. For information on how to specify c, see **plot**.

Note: The "colour" -- this is a Canadian product. You will see things like this as well as other indications that it's not an all-American world out there. For example, you can convert pints into Imperial Gallons through Tools -> Assistants -> Units Calculator.

6.8 More operations on lists

So far we have talked only about creating lists, and assigning lists as the value of a variable. You can also generate a sublist of a list, find a particular item in a list by its position index, count the number of items in the list, and convert a list into other types of data.

Table 6.12: Operations on lists

<table>
<thead>
<tr>
<th>Operation</th>
<th>Example</th>
<th>Commentary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a list</td>
<td>$s1 := {a, b, c, a}$</td>
<td>Lists can contain symbols, numbers, expressions -- anything, even other lists.</td>
</tr>
<tr>
<td></td>
<td>$s2 := {1, 3.47, 97, -5.9, 2.1}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$s3 := {1, 3.47, 97, -5.9, 2.1}$</td>
<td></td>
</tr>
</tbody>
</table>
### Operation

| Specifying a sublist of values | $ts1 := s1[1..3]$  
|--------------------------------| $ts1 := [a, b, c]$  
| Commentary | If a list is followed by another pair of braces with a range inside, then a sublist is computed as a result. Here we have the list that's the first through third items of $s1$.  
| Example |  
|  
| Specifying one item from the list | $s1[2]$  
| Commentary |  
| Example |  
|  
| Specifying the last item in the list | $s1[-1]$  
| Commentary |  
| Example |  
|  
| Specifying a sublist with one item | $s2[3..3]$  
| Commentary |  
| Example |  
|  
| Specifying a sublist from the 3rd from the end to the end | $s1[-3..-1]$  
| Commentary |  
| Example |  
|  
| Counting the number of items in the list | $n := nops(s2)$  
| Commentary |  
| Example |  
|  
| Adding together all the items in the list | $\sum_{i=1}^{n}s2[i]$  
| Commentary |  
| Example |  
|  
| Computing the average of all the numbers in the list | $\frac{\sum_{i=1}^{n}s2[i]}{n}$  
| Commentary |  
| Example |  
|  
| Converting a list into a sequence | $op(ts1)$  
| Commentary |  
| Example |  
|  
| Converting a list into a string. | $convert(s2, string)$  
| Commentary |  
| Example |  

### 6.9 solve, lists and sequences

To solve a system of equations, use a list of expressions or equations for the first argument to `solve`. Use a list of variables as the second argument. `solve` will return a sequence of lists as the result.

When `solve` finds two solutions for an equation (such as if the equation is quadratic), it will return a sequence of solutions. You can recognize a sequence and distinguish it from a list because, the sequence is missing the enclosing brackets `[ ]` that a list has.

Part-selection operations work in sequences in a similar fashion as they were described here (page 85).

In `solve`, lists and sequences look as if they are almost interchangable. Later on we will see situations where lists and sequences must be handled differently.
Table 6.13: Solving systems of equations with solve

<table>
<thead>
<tr>
<th>Equation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( eqn := 3x = x^2 - 28 )</td>
<td>We get a sequence of solutions since there is a double root.</td>
</tr>
<tr>
<td>( solns := solve(eqn, x) )</td>
<td>( -4, 7 )</td>
</tr>
<tr>
<td>( eval(eqn, x = solns[1]) )</td>
<td>( -12 = -12 )</td>
</tr>
<tr>
<td>( eval(eqn, x = solns[2]) )</td>
<td>( 21 = 21 )</td>
</tr>
<tr>
<td>( system := [3x + 5y = 6, 2x - 5 = y] )</td>
<td>We want to assign a set as the value of the variable system. Maple tells us that the name is already in use as a built-in function, so it won't let us do that.</td>
</tr>
<tr>
<td>( sys := [3x + 5y = 6, 2x - 5 = y] )</td>
<td>We choose a different variable to assign the set to.</td>
</tr>
<tr>
<td>( vars := [x, y] )</td>
<td>We specify the set of variables we want to solve for, then call solve.</td>
</tr>
<tr>
<td>( solns := solve(sys, vars) )</td>
<td>We get a list with one element (which itself is a list) as a solution.</td>
</tr>
<tr>
<td>( solns[1] )</td>
<td>( x = \frac{31}{13}, y = -\frac{3}{13} )</td>
</tr>
<tr>
<td>( eval(sys, solns[1]) )</td>
<td>( 6 = 6, \frac{3}{13} = -\frac{3}{13} )</td>
</tr>
</tbody>
</table>
This system of equations has two distinct solutions, so we get a list with two elements in it. Each element is a distinct solution.

We can find the non-negative roots of an equation by including the appropriate inequality in the list of relations given to solve. The result by default is a set, however, if the second argument to solve is a list of variables, then the result will come back as a list. You can select items from a set using the same notation as with lists.

### 6.10 Evaluation, eval, and assignment

Here's a problem to consider:

We have an expression relating time $t$ to voltage registered by a capacitor as it is being charged by a battery. In a mathematics or electrical engineering textbook, we might see this written as $V(t) = 35 + (65 - 35) \cdot \left(1 - e^{-\frac{t}{3}}\right)$. We are interested in taking this expression for voltage and doing several calculations with it -- plotting it for a range of $t$, finding values of $t$ that correspond to a specified voltage (e.g. "find the time $t$ when the voltage reached 55 volts"), or finding a voltage corresponding to a specified time (e.g. "find the voltage at $t=2.5$ minutes after the start").

If we set up an assignment in Maple $V := 35 + (65 - 35) \cdot \left(1 - e^{-\frac{t}{3}}\right)$, then we could calculate the voltage at $t=2.5$ minutes by assigning $t$ the value 2.5 and then evaluating $V$. The second evaluation will cause the current value of $t$ to be used. However, if we wanted to plot the expression $V$ after that, then we'd have problems because whenever we would type $t$, Maple would use the value of $t$ rather than the symbol $t$. 
Table 6.14: Evaluating an expression using a particular value of one of the variables in the expression, and then plotting

\[
V := 35 + (65 - 35) \cdot \left(1 - e^{-\frac{t}{3}}\right)
\]

\[
V := 65 - 30e^{-\frac{t}{3}}
\]  
(6.41)

\[
t := 2.5
\]

\[
t := 2.5
\]  
(6.42)

\[
V
\]

\[
51.96205374
\]  
(6.43)

\[
\text{plot}(V, t = 0..10)
\]

Error, (in plot) unexpected option: 2.5 = 0..10

Rather than \(t=0..10\) Maple is seeing 2.5=0..10 because it is using the value of \(t\) when evaluating what we typed.

\[
\text{unassign}(V)
\]

\[
\text{plot}(V, t = 0..10)
\]

We can clear the path for plotting by unassigning \(t\) first. Note that if we did a \texttt{restart} instead of an unassign we would lose the assignment to \(V\). Restarting at this point is a bad idea, since it forces us to redefine \(V\) as well as \(t\).

The same problem would happen if we tried to solve an equation involving \(V\) if we had already assigned \(t\) a value.

Table 6.15: Evaluating an expression using a particular value of one of the variables in the expression, and then solving

\[
\text{restart}
\]

\[
V := 35 + (65 - 35) \cdot \left(1 - e^{-\frac{t}{3}}\right)
\]

We set up an expression and then evaluate it at \(t=4.7\) seconds.
Because Maple is evaluating the names \( V \) and \( t \) in the `solve` operation, it is seeing \( \text{solve} \left( 35 + (65 - 35) \cdot \left( 1 - e^{\frac{-t}{3}} \right), 4.7 \right) \) which it cannot solve because there are no variables in the equation to solve for.

We can clear the path for solving by unassigning \( t \) first. Doing a `restart` would not work, because that would also unassign everything, including \( V \). We would lose the expression we want to solve for.

It can be tedious to have to remember to unassign variables if we want to go back to using them as symbols in the expression. We recommend using the `eval` operation (also available in the clickable menu as \( f(x) \big|_{x = a} \)) instead of assignment, if you are switching back and forth between using values for a variable and using it as a symbol. `eval` returns the same result as if you had done assignment temporarily, but there is nothing to be undone afterwards.

You can evaluate using values for several variables by giving a list of equations instead of a single equation as the second argument to `eval`.

### Table 6.16: Evaluating an expression using a particular value of one of the variables in the expression using eval, and then solving

<table>
<thead>
<tr>
<th>( t := 4.7 )</th>
<th>( t := 4.7 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( V )</td>
<td>( 58.73780530 )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><code>solve(V = 55, t)</code></th>
<th>Because Maple is evaluating the names ( V ) and ( t ) in the <code>solve</code> operation, it is seeing ( \text{solve} \left( 35 + (65 - 35) \cdot \left( 1 - e^{\frac{-t}{3}} \right), 4.7 \right) ) which it cannot solve because there are no variables in the equation to solve for.</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>unassign('V')</code></td>
<td>We can clear the path for solving by unassigning ( t ) first. Doing a <code>restart</code> would not work, because that would also unassign everything, including ( V ). We would lose the expression we want to solve for.</td>
</tr>
<tr>
<td><code>solve(V = 55, t)</code></td>
<td>( 3 \ln(3) )</td>
</tr>
</tbody>
</table>

We set up an expression and then evaluate it at \( t=4.7 \) seconds.

<table>
<thead>
<tr>
<th>( V := 35 + (65 - 35) \cdot \left( 1 - e^{\frac{t}{3}} \right) )</th>
<th>Evaluating ( V ) produces the same result as before (an expression with the symbol ( t ) in it), so we can immediately set up an equation and solve it for ( t ).</th>
</tr>
</thead>
<tbody>
<tr>
<td>( V := 65 - 30 e^{\frac{t}{3}} )</td>
<td>( 51.96205374 )</td>
</tr>
<tr>
<td><code>eval(V, t=2.5)</code></td>
<td>( 51.96205374 )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><code>solve(V = 55, t)</code></th>
<th>( 3 \ln(3) )</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>t</code></td>
<td>( t )</td>
</tr>
</tbody>
</table>

Note that the `eval` did not assign a value to \( t \). It's still just a symbol at this point.
This is an example of how to set up a script using parameters while taking advantage of eval. Several symbols in the expression for voltage are set up and assigned as parameters. The expression that describes how voltage changes over time is not a parameter, although it is assigned a name for easier use in subsequent steps of the computation.

<table>
<thead>
<tr>
<th>restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Begin parameters</td>
</tr>
<tr>
<td>$V_i := 35$</td>
</tr>
<tr>
<td>$V_{max} := 65$</td>
</tr>
<tr>
<td>$v_t := 55$</td>
</tr>
<tr>
<td>$t_0 := 4.7$</td>
</tr>
<tr>
<td>End parameters</td>
</tr>
<tr>
<td>$V_{prime} := V_i + (V_{max} - V_i) \cdot \left(1 - e^{-\frac{t}{\tau}}\right)$</td>
</tr>
</tbody>
</table>

Table 6.17: An example using assignment and eval

This is an example of how to set up a script using parameters while taking advantage of eval. Several symbols in the expression for voltage are set up and assigned as parameters. But we use eval to maintain $\tau$ and $t$ as symbols in the expression $V_{prime}$. 

<table>
<thead>
<tr>
<th>restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Begin parameters</td>
</tr>
<tr>
<td>$t_{exp} := \text{eval}(V_{prime}, [t = t0])$</td>
</tr>
<tr>
<td>$t_{exp} := 65 - 30 \cdot e^{-\frac{4.7}{\tau}}$</td>
</tr>
<tr>
<td>$t_{val} := \text{solve}(t_{exp} = v_t, \tau)$</td>
</tr>
<tr>
<td>$t_{val} := 4.278124365$</td>
</tr>
<tr>
<td>$t_{exp} := \text{eval}(V_{prime}, \tau = t_{val})$</td>
</tr>
<tr>
<td>$t_{exp} := 65 - 30 \cdot e^{-0.2337472955 \cdot t}$</td>
</tr>
<tr>
<td>$\text{solve}(t_{exp} = v_t, t)$</td>
</tr>
<tr>
<td>$4.699999999$</td>
</tr>
</tbody>
</table>
\[
V_i \ := \ 35
\]

\[
V_{\text{max}} \ := \ 65
\]

\[
v_t \ := \ 55
\]

\[
t_0 \ := \ 4.7
\]

**End parameters**

\[
V_{\text{prime}} \ := \ V_i + (V_{\text{max}} - V_i) \left( 1 - e^{-\frac{t}{\tau}} \right)
\]

\[
V_{\text{prime}} \ := \ 65 - 30 e^{-\frac{t}{\tau}}
\]

We use the information that the capacitor is observed to at \(v_t\) volts at time \(t=t_0\) to find the value of \(\tau\) that is consistent with this.

\[
tauExpr := \ \text{eval}(V_{\text{prime}}, [t = t_0])
\]

\[
tauExpr \ := \ 65 - 30 e^{-\frac{4.7}{\tau}}
\]

\[
tauValue := \ \text{solve}(tauExpr = v_t, tau)
\]

\[
tauValue \ := \ 4.278124365
\]

\[
tExpr := \ \text{eval}(V_{\text{prime}}, \text{tau} = tauValue)
\]

\[
tExpr \ := \ 65 - 30 e^{-0.2337472955 \cdot t}
\]

\[
tExpr\text{ is the value of the expression with the values of the parameters and the calculated value of } \tau \text{ plugged into } V_{\text{prime}}. \text{ We can solve an equation based on this formula to find the time when we achieve a voltage of 55 volts in this configuration.}
\]

\[
solve(tExpr = v_t, t)
\]

\[
4.6999999999
\]
6.11 Summary of Chapter 6 material

<table>
<thead>
<tr>
<th>Troubleshooting textual input in Maple</th>
<th>Examples with error(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Remember to...</strong></td>
<td></td>
</tr>
<tr>
<td>Supply a function name and arguments (parameters)</td>
<td><code>solve(x = 3 \cdot x^2 - 2)</code> should be <code>solve(x = 3 \cdot x^2 - 2, x)</code></td>
</tr>
<tr>
<td>Match delimiters (parenthesis and brackets)</td>
<td><code>solve(x = 3 \cdot x^2 - 2, x)</code> should be <code>solve(x = 3 \cdot x^2 - 2, x)</code></td>
</tr>
<tr>
<td>Press the right arrow key to exit from variable exponents</td>
<td><code>solve(x = 3 \cdot x^2 - 2, x)</code> should be <code>solve(x = 3 \cdot x^2 = 2, x)</code></td>
</tr>
<tr>
<td>Set ranges correctly when plotting</td>
<td><code>plot(x = 3 \cdot x^2 - 2, x, -3..3)</code> should be <code>plot(x = 3 \cdot x^2 - 2, x = -3..3)</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Plotting</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plotting multiple expressions</td>
<td><code>plot([x^2, \sin(x^2)], x = 0..4)</code></td>
</tr>
</tbody>
</table>
| Plotting with lists | `xList := [1, 2, 3, 4]`
`yList := [5, 6, 7, -1]`
`plot(xList, yList, style = point)` (6.70) (6.71) |
| Using multiple colors in a multi-plot | `plot([\sin(x), \sin(\frac{x}{2}), \sin(2 \cdot x)], x = -4 \cdot Pi..4 \cdot Pi, color = ["red", "green", "blue"]`) |
| Set the titles of the axes | `plot(3.5 \cdot x^2 - 2, x = 1..5, labels = ["temperature (in degrees C)", "pressure in kilopascals"]` |

<table>
<thead>
<tr>
<th>Using Maple's built-in help</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Help&gt;Maple Help or press Ctrl-F1 (Command-F1 on a Mac)</td>
<td>Remember that you can click on related topics when viewing the help for a particular command</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operations on lists</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a list</td>
<td><code>s1 := [a, b, c, a]</code> <code>[a, b, c, a]</code> (6.72)</td>
</tr>
<tr>
<td>Specify a sublist of values</td>
<td><code>ts1 := s1[1..3]</code> <code>[a, b, c]</code> (6.73)</td>
</tr>
<tr>
<td>Specify one item from the list</td>
<td><code>s1[1]</code> <code>a</code> (6.74)</td>
</tr>
<tr>
<td>Specify a sublist with one item</td>
<td><code>s2[3..3]</code> <code>[97]</code> (6.75)</td>
</tr>
<tr>
<td>Count the number of items in the list</td>
<td>( n := \text{nops}(s2) )</td>
</tr>
<tr>
<td>Add together all the items in the list</td>
<td>( \sum_{i=1}^{n} s2[i] )</td>
</tr>
<tr>
<td>Compute the average of all the numbers in the list.</td>
<td>( \frac{\sum_{i=1}^{n} s2[i]}{n} )</td>
</tr>
<tr>
<td>Convert a list into a sequence</td>
<td>( op(ts1) )</td>
</tr>
<tr>
<td>Convert a list into a string</td>
<td>( \text{convert}(s2, \text{string}) )</td>
</tr>
</tbody>
</table>

### Solving a system of equations using lists

| Create the system of equations | \( \text{sys} := \{3 \cdot x + 5 \cdot y = 6, 2 \cdot x - 5 = y\} \) | \( \{3 \cdot x + 5 \cdot y = 6, 2 \cdot x - 5 = y\} \) | (6.81) |
| Set the variables of the system | \( \text{vars} := [x, y] \) | \( [x, y] \) | (6.82) |
| Solve the system and extract the first solution of possibly many solutions | \( \text{solns} := \text{solve}(\text{sys}, \text{vars})[1] \) | \( \{x = \frac{31}{13}, y = -\frac{3}{13}\} \) | (6.83) |

### Evaluating an expression with eval instead of assignment

| eval allows you to substitute a value for a variable within an expression, without assigning that value to the variable. | \( V := 35 + (65 - 35) \cdot \left(1 - e^{-\frac{t}{3}}\right) \) | \( V := 65 - 30 \cdot e^{-\frac{t}{3}} \) | (6.84) |
| eval\((V, t = 2.5)\) | | 51.96205374 | (6.85) |
| \( t \) | \( t \) | | (6.86) |
Chapter 6 More sophisticated scripting: textual entry, multiplots, point plots, character strings, lists, solving equations, and evaluation
7 Chapter 7 Using and Defining Functions

7.1 Chapter overview

Functions occur so much in mathematics that it's natural that Maple knows a lot about them and how to compute with them. You can also define your own functions in Maple, as with most other programming languages.

There are a few pitfalls in the use of functions in Maple:

a) The names of common mathematical functions used in Maple may differ from what you are used to from high school math.

b) Some functions in Maple do non-mathematical things, such as solve, and plot. Others take novel arguments -- lists, equations, and ranges, rather than numbers.

c) Maple function definition uses := and -> rather than equals (=) as found in conventional math textbook notation. This is because the "context-free" language processing of Maple thinks an equation is being defined whenever it sees an equal sign. It's currently infeasible for conventional programming language-processing technology to use context.

7.2 Functions in computer languages: a way of producing an output from inputs

Everyone is introduced to the idea of a function in secondary school mathematics: Calculators can compute many of the common functions found in high school algebra and pre-calculus: sin, cos, ln, \( \sqrt{} \), etc.

Maple can evaluate these functions. The common ones are found in the Expression palette but there are hundreds more. When entering a functional expression, the syntax used is:

```
function name ( sequence of arguments )
```

The parentheses are mandatory in Maple. Forgetting parentheses can be irksome because often the computer leaves you to figure out yourself that the answer you're getting is wrong.

<table>
<thead>
<tr>
<th>Table 7.1: Function results in Maple</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example</strong></td>
</tr>
<tr>
<td>( \sin(0.35), \cos\left(\frac{3.14159265}{2}\right), \ln(3.72), \sqrt{2.0} )</td>
</tr>
<tr>
<td>0.3428978075, 1.794896619 ( \times 10^{-9} ), 1.313723668, 1.259921050</td>
</tr>
<tr>
<td>( \cos\left(\frac{\pi}{2}\right), \ln(1), \sqrt{2}, \sin\left(\frac{35}{100}\right), \left(\frac{5}{2}\right), \log_{10}(1001) ), \log(1001.0)</td>
</tr>
<tr>
<td>0, 0, 2(^{1/2}), ( \sin\left(\frac{7}{20}\right) ), 10, ( \ln(1001) ), \ln(10), 3.000434077</td>
</tr>
<tr>
<td>( \cos([a, b, c]) )</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
Example | Commentary
--- | ---
\( \cos(2, \pi) \) | This mistake might come about if you typed a comma instead of a \(*\).\nError, (in \( \cos \)) expecting 1 argument, got 2
\( \cos 1 2 \) | What was this person thinking? Whatever it was, Maple doesn't know what to do with it.
Error, missing operation
\( \cos \left( \frac{\pi}{4} \right) \) | Another delimiter message. Look for extra or missing parentheses.
Error, unable to match delimiters
\( \cos \left( \frac{\pi}{4} \right) \) | This is what happens if you forget the mandatory parentheses. There is no error message, but what Maple is giving you is the product of \( \pi \), the symbol \"cos\", and \( \frac{1}{4} \). One of the ways you can tell that this isn't \( \cos \left( \frac{\pi}{4} \right) \) is what happens when you ask Maple to numerically approximate this expression.
\( \frac{1}{4} \cos \pi \) | \( (7.5) \)
at 5 digits
0.78540 cos | \( (7.6) \)
\( \cos \left( \frac{\pi}{4} \right) \) | This is the correct way to compute the cosine of \( \frac{\pi}{4} \) radians.
\( \frac{1}{2} \sqrt{2} \) | \( (7.7) \)
at 5 digits
0.70710 | \( (7.8) \)

The attachment at the end of this chapter shows some of the many other functions available in Maple. Some of them work on lists, sets, equations rather than on numbers or expressions. However, the same principle applies: they have a rule for taking the value of their inputs (also known as arguments) and computing a result from them.

7.3 The textual names of common functions: doing math calculations using the keyboard.

Some built-in mathematical functions have textual names that are already quite familiar from their use in mathematics textbooks: \( \sin, \cos, \ln \). Some are used so often that the most convenient thing to do is to remember their names. \( \sqrt{\text{sgt}}, \abs, \min, \text{and max} \) are straightforward -- they are naturally thought of as functions and have names that are abbreviations of the standard nomenclature. Others, such as \( \arcsin \), or \( \log_{10} \) have names that make sense but you'd have to look them up in the Maple on-line documentation to know.
### Table 7.2: Textual names for common math functions

<table>
<thead>
<tr>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>expr := abs(arcsin(log10(t)))</code></td>
<td>Maple knows &quot;log10&quot; is the base 10 logarithm. It automatically converts it into the equivalent expression using ( \ln ), the natural (base ( e )) logarithm. (7.9)</td>
</tr>
<tr>
<td><code>plot(expr,t=-1..1, title = &quot;A funky non-linear plot&quot;)</code></td>
<td>Many users will find it faster to type in the textual version of <code>plot</code> than to use the plot builder. Furthermore, if we want to change the expression or the range or labels of the plot, it's easier to edit the text and re-execute the region than it would be with the clickable interface.</td>
</tr>
<tr>
<td><code>max(-3,92,43.7,0,sqrt(16))</code></td>
<td><code>max</code> can find the maximum when all of its arguments are numbers. <code>max</code> and <code>min</code> can do limited reasoning if some of the arguments are not numeric. For example, in the third example, Maple knows that ( \text{abs}(x) - 2 ) is not going to be the maximum because ( \text{abs}(x) ) is must be larger. (7.10)</td>
</tr>
<tr>
<td><code>min(5,7,29,x,3)</code></td>
<td>(7.11)</td>
</tr>
<tr>
<td><code>max(\text{abs}(x), \frac{x}{2}, 5.7, \text{abs}(x) - 2, 0)</code></td>
<td>(7.12)</td>
</tr>
</tbody>
</table>

### 7.4 A function name to commit to memory: exp

The name of the exponential function in Maple is \( \text{exp} \). Instead of writing \( e^x \), use \( \text{exp}(x) \).

The textual doppelganger \( e^x \) does not work as a way of calculating a power of \( e \), the base of the natural logarithm (where "e" is just the letter typed at the keyboard, not augmented by command completion as described in the section on command completion (page 98)). The orientation in college-level mathematics to view "a power of \( e \)" as a function is a pervasive change in point...
of view from what you may have seen in high school. Making a point to use the new notation frequently is the best way to make the switch.

Table 7.3: exp is the name of the exponential function

<table>
<thead>
<tr>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \exp(x) \cdot \exp(y) )</td>
<td>( e^x e^y ) \hspace{1cm} \text{(7.13)} is the textual way of writing &quot;the symbolic constant } e \text{ raised to the power } x. \text{ Maple knows how to simplify symbolic expressions with the simplify → assuming real operations in the clickable menu.}</td>
</tr>
<tr>
<td>assuming real</td>
<td>( e^x + y ) \hspace{1cm} \text{(7.14)} \text{ We can calculate powers of } e \text{ from the keyboard. For comparison we do the same calculation using the Common Symbols and Expressions Palette. While more of a &quot;sure thing&quot;, proficient users would be able to get the keyboard version entered more quickly.}</td>
</tr>
<tr>
<td>( \exp(1) )</td>
<td>( 1.105170918 ) \hspace{1cm} \text{(7.15)} \text{ We can enter an expression involving exponentials using only the keyboard. Maple will regard it as meaning the same thing as the expression entered using the combination of the keyboard and the Palette.}</td>
</tr>
<tr>
<td>( e^1 )</td>
<td>( 1.105170918 ) \hspace{1cm} \text{(7.16)} \text{ We can calculate powers of } e \text{ from the keyboard. For comparison we do the same calculation using the Common Symbols and Expressions Palette. While more of a &quot;sure thing&quot;, proficient users would be able to get the keyboard version entered more quickly.}</td>
</tr>
<tr>
<td>( V_i + (V_{\text{max}} - V_i) \left( 1 - \exp\left( -\frac{t}{\tau}\right) \right) )</td>
<td>( V_i + (V_{\text{max}} - V_i) \left( 1 - e^{-\frac{t}{\tau}} \right) ) \hspace{1cm} \text{(7.17)} \text{ We can enter an expression involving exponentials using only the keyboard. Maple will regard it as meaning the same thing as the expression entered using the combination of the keyboard and the Palette.}</td>
</tr>
</tbody>
</table>

7.5 How can I remember so many functions? Use command completion and on-line documentation.

A well-developed system such as Maple, Matlab, or C# has thousands of built-in functions. You may need to use on a regular basis the names of at least several dozen functions to get your work done. The good news is that learning about functions is not that taxing -- if you own a scientific calculator you've already dealt with a situation where you can operate a dozen functions.

A reasonable stance to take is to be familiar (i.e. know "by heart") functions and symbols that you often use and to be adept at using documentation to look up the details of the ones that you need only occasionally. In exams and tests that let you use a computer, quick recall of the most common will be useful and expected. However, you should also expect to need to use additional functions and information that you haven't committed to memory. No one will expect you to memorize it all -- but they will expect you to be able to use it quickly anyway with the help of on-line documentation.

A quick-recall method for access to common functions with textual entry is to type the escape key (Esc) after typing the first few characters of the function's name. A pop-up menu will appear that will list possible ways of completing what you typed. This is called command completion. Recall that you've already used this feature to enter symbolic constants such as \( e \), the natural logarithm base. If what you are entering is a Maple operation such as \textit{solve}, command completion will provide a textual template to fill in the rest of the arguments. It will also provide the parentheses required when entering functional notation textually.

Table 7.4: Command Completion

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( sol )</td>
<td>We type ( sol ) and then type the ESC key. A pop up menu shows option, including several forms of \textit{solve}.</td>
</tr>
</tbody>
</table>
Clicking on one of the options provides something to fill out. We are not compelled to have a second variable of \( x \), it's just short-hand reminder that the second argument is the variable and the first one is the equation. We can then go on to edit the "auto completed" template so that we turn the general form into exactly what we need.

Experienced users refer to the on-line documentation to help remember details about functions. As has mentioned earlier, this is available through Help -> Maple Help menu feature (key shortcut: press the control key and then the F1 key). If you recall a phrase or a name of a function, you can type it into the search field and the on-line help system will, like Google, produce the pages it has about your text entry. You can then explore further by pressing links. Trying the examples typically given at the end of the description is a good way to get a form that you can use for your own purposes.

We want to find information about how to use the inverse sine function in Maple. We start up on-line help and type in "inverse trigonometric" into the search field, then hit the "search" button. The page we see does tell us that it's probably called "arcsin" but we'd like to see more. We see a link in the "see also" which we click on.
Using on-line help

This uncovers more links. The one that says “invtrig” seems promising so we click on that link.

This seems to be the page we want to spend time looking at. There are plenty of examples at the bottom.

7.7 Defining your own functions with -> (arrow)

Maple allows you to define simple functions with the use of ->. The general form is

\[
\text{function name} := (\text{sequence of arguments}) \rightarrow \text{expression that describes result}.
\]

These can be entered through the Expression Palette, or textually. The arrow is entered textually by typing a - and then a >, with no spaces separating them. We give an example of function definition in the following example.

Table 7.5: A function definition in a math textbook

Fruit flies are placed in a half-pint milk bottle with a banana (for food) and yeast plants (for food and to provide a stimulus to lay eggs). Suppose that the fruit fly population after \( t \) days is given by

\[
P(t) = \frac{230}{1 + 56.5 \cdot e^{-0.37t}}
\]

(a) What is the population after five days?
(b) How long does it take for the population to reach 180?

Analyzing the text, we see that it defines a function named $P$. It takes one input (argument), $t$, and produces as output whatever you get from evaluating the expression $\frac{230}{1 + 56.5 \cdot e^{-0.37 \cdot t}}$. Assuming that Maple understands the definition of $P$ like it does a built-in function, then the description of what happens when you evaluate $P(5)$ would be:

"Substitute 5 for wherever you see $t$ in the expression $\frac{230}{1 + 56.5 \cdot e^{-0.37 \cdot t}}$. This gives you $\frac{230}{1 + 56.5 \cdot e^{-0.37 \cdot 5}}$. Perform all the arithmetic and relevant simplifications in this expression and return that as the result of the function."

Even though 5 is an exact number, because there are limited precision numbers in the expression we expect that the result will be a limited-precision number. If the expression had only exact numbers in it, then the calculation would be done exactly.

What we would like to do is to tell Maple about the definition of $P$ and use it in our work.

We can do this through the clickable interface. We anticipate reuse of this for other days and population levels, and turn it into a parameterized script:

Table 7.6: User-defined functions through the Expression Palette

<table>
<thead>
<tr>
<th>User-defined functions through the Expression Palette</th>
<th>Commentary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Start of parameters</strong></td>
<td>The general form is from the $f := a \rightarrow y$ in the Expression palette. We alter slots in the template to mention the specific function by name, the name of the input, and the expression that describes how to calculate the output.</td>
</tr>
<tr>
<td>$\text{numDays} := 5$</td>
<td></td>
</tr>
<tr>
<td>$\text{popLevel} := 180$</td>
<td></td>
</tr>
<tr>
<td>$P := t \rightarrow \frac{230}{1 + 56.5 \cdot e^{-0.37 \cdot t}}$</td>
<td></td>
</tr>
<tr>
<td>$t \rightarrow \frac{230}{1 + 56.5 \cdot e^{(-1) \cdot 0.37 \cdot t}}$</td>
<td>(7.20)</td>
</tr>
</tbody>
</table>
User-defined functions through the Expression Palette

Compute the number of flies after $numDays = 5$ days.

\[ P(numDays) \]

\[ \text{23.27016688} \]  

(7.21)

We can plot $P(t)$ like we would any other expression. One of the options to plot (see plot options in on-line help) is the ability to specify the title of the graph by giving `title=string` as an additional argument to the `plot` function. Note that we are not using the clickable interface to do plotting.

We plot the function to see how the population grows. This is not needed by the problem but it helps us understand the situation better.

\[
plot(P(t), t=0..numDays, title = "Fruit flies like a banana", labels = ['t', '# of flies'])
\]

In anticipation of using the value in later work, we assign it to the variable `soln`. We would then intend to take further steps using `soln`.

Compute when the population reaches desired level by solving the equation $P(t) = \text{popLevel}$.

\[ soln := solve(P(t) = \text{popLevel}, t) \]

\[ 14.36533644 \]  

(7.22)

End of script
We could have defined the function textually just by typing \( P := (t) \rightarrow \ldots \) followed by the rest of the expression instead of using the Expression Palette.

### 7.8 Using functional evaluation instead of using assignment or eval

The problem with assigning variables is that it may cause unwanted side-effects, such as when trying to solve or plot with expressions involving those variables. We have seen in the section on eval (page 85) that we can use eval instead of assignment, which saves us the chore of having to assign and then unassign variables so that they remain as symbols when solving or plotting. We can more succinctly avoid this problem by defining a function based on the expression, and then causing evaluation to occur with standard functional notation.

#### Table 7.7: Comparing evaluation using eval and functional notation

<table>
<thead>
<tr>
<th>restart</th>
<th>( V := 35 + (65 - 35) \cdot \left(1 - e^{-\frac{t}{3}}\right) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( V := 65 - 30 e^{-\frac{t}{3}} )</td>
<td>( V := 65 - 30 e^{-\frac{t}{3}} ) (7.23)</td>
</tr>
<tr>
<td>eval(( V, t = 2.5 ))</td>
<td>51.96205374 (7.24)</td>
</tr>
<tr>
<td>eval(( V, t = -2.5 ))</td>
<td>-4.02927673 (7.25)</td>
</tr>
</tbody>
</table>

\( v := (t) \rightarrow 35 + (65 - 35) \cdot \left(1 - e^{-\frac{t}{3}}\right) \)

\( v := t \rightarrow 65 - 30 e^{-\frac{t}{3}} \) (7.26)

\( v(2.5) \)

51.96205374 (7.27)

\( v(-2.5) \)

-4.02927673 (7.28)
7.9 Troubleshooting function definitions

Function definition in Maple differs from the standard notation in high school mathematics. This is because computers find it hard to detect the context where language is being used, so the same notation has to work both for mathematics and programming. The standard notation can't be made to do this, so extensions and modifications to the notation need to be made. (In other words, it's not a conspiracy to make your life harder gratuitously.)

The technology that is standard in most computer language understanding systems needs to assign a unique meaning to input from
the way it looks. Equations already use "=" so if you use f(x) = ... Maple will understand you to be talking about an equation, not a function definition. Use := when assigning a function a name.

Table 7.8: Troubleshooting function definitions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( F := (m_1, m_2, r) \rightarrow \frac{g \cdot m_1 \cdot m_2}{r^2} )</td>
<td>This defines a function for the gravitational attraction force. ( m_1, m_2 ) and ( r ) are the arguments. ( g ) is another symbol used in the expression but it isn't included as an argument.</td>
</tr>
<tr>
<td>( g := 6.673 \times 10^{-11} )</td>
<td>We define the value of ( g ). We expect not to change this while using the function. We found the value by typing &quot;gravitation&quot; into Maple online help.</td>
</tr>
<tr>
<td>( F(1, 2, 10) )</td>
<td>This calculates the attraction force between a 1 kilogram and 2 kilogram mass that is 10 meters apart. The force is in units of Newtons.</td>
</tr>
<tr>
<td>( Fbad := (m_1, m_2, r) \rightarrow \frac{g \cdot m_1 \cdot m_2}{r^2} )</td>
<td>This doesn't work at all. Do you see the difference between the definition of ( F ) and ( Fbad )? One uses assignment := the other is either mistyped or mistaken.</td>
</tr>
<tr>
<td>( FF := (m_1, m_2, r) = \frac{g \cdot m_1 \cdot m_2}{r^2} )</td>
<td>This tries to use &quot;=&quot; instead of &quot;-&gt;&quot; to define the function ( FF ). There is no error message, however, because this assigns ( FF ) an equation rather than a function definition, which is perfectly valid, even if it isn't what we were thinking of.</td>
</tr>
<tr>
<td>( FF(1, 2, 10) )</td>
<td>We can tell that this supposed function definition isn't working when we test it. When we try to use the function in the usual way we don't get the number we were expecting, as with ( F(1,2,10) ). With testing, you verify, looking for evidence stronger than &quot;no error message&quot; that what you entered is what you want.</td>
</tr>
<tr>
<td>( F2(m_1, m_2, r) := \frac{g \cdot m_1 \cdot m_2}{r^2} )</td>
<td>This illustrates an alternative way of defining a function, although it is not the form recommend by these notes. Rather than &quot;=&quot; as would appear in a math textbook, the assignment operation &quot;:=&quot; is used instead. This produces the following pop-up:</td>
</tr>
</tbody>
</table>

\[
F := (m_1, m_2, r) \rightarrow \frac{g \cdot m_1 \cdot m_2}{r^2}
\]

\[
g := 6.673 \times 10^{-11}
\]

\[
F(1, 2, 10)
\]

\[
Fbad := (m_1, m_2, r) \rightarrow \frac{g \cdot m_1 \cdot m_2}{r^2}
\]

\[
FF := (m_1, m_2, r) = \frac{g \cdot m_1 \cdot m_2}{r^2}
\]

\[
FF(1, 2, 10)
\]

\[
F2(m_1, m_2, r) := \frac{g \cdot m_1 \cdot m_2}{r^2}
\]
Clicking "ok" to function definition will create the proper function definition, as the subsequent line of the computation indicates.

Testing this function definition, we see that we get the same result as with F

\[
F2(1, 2, 10) = 1.334600000 \times 10^{12}
\]  

(7.36)

### 7.10 Using functions from library packages, with

As with other systems for technical computation, Maple has thousands of built-in functions. The definitions for some of them such as \( \text{cos} \) or \( \text{solve} \) are automatically loaded into Maple when it starts up. However, if all definitions were loaded, Maple would take a long time to start up and would require large amounts of memory even before you had done any work in it. To avoid this, most functions are loaded only when the user requests it.

Most built-in functions are organized into collections called *packages*. The general way to access a function belonging to a package is through \( \text{package}[\text{function}] \). The least squares function belongs to a package named *CurveFitting*, hence its full name is *CurveFitting[LeastSquares]*.

The \( \text{with} \) operation in Maple will load all the functions in the specified package into Maple. After this operation, functions can be referred to with just their "short name", e.g. \( \text{LeastSquares} \) rather than \( \text{CurveFitting[LeastSquares]} \). Doing a \( \text{with} \) can save you typing if you expect to use a function, or several functions, from a package several times during a Maple session. Ending the line with a colon (:) will suppress printing of all the functions in the package that usually occurs.

<table>
<thead>
<tr>
<th>Table 7.9: with, the package loading operation</th>
</tr>
</thead>
</table>

\[
\text{restart}
\]

\[
pData := \text{[134.2, 142.5, 155.0, 159.8, 171.1, 184.2]};
\]

\[
\begin{align*}
&[134.2, 142.5, 155.0, 159.8, 171.1, 184.2] \\
&\text{(7.37)}
\end{align*}
\]

\[
tData := \text{[0, 20.1, 39.8, 60.0, 79.9, 100.3]};
\]

\[
\begin{align*}
&[0, 20.1, 39.8, 60.0, 79.9, 100.3] \\
&\text{(7.38)}
\end{align*}
\]

\[
p\text{ressureFormula} := \text{LeastSquares}(tData, pData, t)
\]

\[
\text{LeastSquares}(0, 20.1, 39.8, 60.0, 79.9, 100.3), [134.2, 142.5, 155.0, 159.8, 171.1, 184.2], t)
\]

\[
\text{7.39)
\]

The CurveFitting package has a number of functions for data fitting. One is called \( \text{LeastSquares} \). However, \( \text{LeastSquares}(...) \) does nothing. Although there is no error message, this is a mistake -- we didn't load the package in with a "with" so the least squares data fitting function isn't known. We get the same kind of behavior as if we had typed in \( f(1,2,3) \) with \( f\) undefined -- Maple just outputs what we typed in.
7.11 Attachment: some built-in problem-solving functions

The functions in the Expression palette have the same name and work similarly to those described in math textbooks. The operations discussed in this attachment are also found in math textbooks, but they are usually not given function names. It may seem novel to you that the rules for solving equations, factoring polynomials, or plotting can be collected together and given a function name. Yet this way of writing about such actions allows us to combine mathematics and programming. Thus `solve`, `plot`, `factor`, etc. are true functions -- they have names, they are invoked with arguments, and return results that can be assigned to a variable.

Table 7.10: Textual names of common operations in Maple

<table>
<thead>
<tr>
<th>Function</th>
<th>Textual name of function</th>
<th>Example</th>
<th>Commentary</th>
</tr>
</thead>
<tbody>
<tr>
<td>solve an expression or an</td>
<td><code>solve</code></td>
<td><code>solve(x^2 - 7*x - 98)</code></td>
<td>If there is more than one root, returns a</td>
</tr>
<tr>
<td>equation</td>
<td></td>
<td>14, -7</td>
<td>sequence of values</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>solve(x^2 = 7*x + 98)</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>14, -7</td>
<td></td>
</tr>
<tr>
<td>solve an expression or an</td>
<td><code>fsolve</code></td>
<td><code>fsolve(x^2 = cos(x))</code></td>
<td>If there is more than one root, returns a</td>
</tr>
<tr>
<td>equation numerically</td>
<td></td>
<td>0.8241323123</td>
<td>sequence of values</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>fsolve(x^2 - 7*x - 97)</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>-6.952272480, 13.95227248</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>fsolve(a*x^2 + b*x + c = 0, x)</code></td>
<td><code>fsolve</code> won't work if the answer is not a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Error, (in fsolve) {a, b, c} are in the</td>
<td>number.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>equation, and are not solved for</td>
<td></td>
</tr>
</tbody>
</table>
plot an expression

## plot

<table>
<thead>
<tr>
<th>Plot takes two inputs. The first is an expression, the second is an equation naming the variable and the horizontal plot range.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{plot}(x^2 - 7x - 98, x = -20..20)$</td>
</tr>
<tr>
<td>$\text{plot}([t^2 + 7t, 4], t = -8..2, \text{color} = [&quot;DodgerBlue&quot;, &quot;Purple&quot;], \text{labels} = [&quot;time&quot;, &quot;velocity&quot;]$</td>
</tr>
</tbody>
</table>

| evaluate at a point | eval |
| --- |
| eval takes two arguments (inputs). The first is an expression, the second is an equation or a list of equations indicating what values to use for one or more variables in the expression. |
| $\text{eval}(x^2 + y^2, x = 1)$ |
| $1 + y^2$ |
| $\text{eval}(x^2 + y^2, [x = 1, y = \pi + 2])$ |
| $1 + (\pi + 2)^2$ |

| right hand side, left hand side of an equation | rhs |
| --- |
| rhs takes two arguments. The first is an expression, the second is an equation. |
| $\text{rhs}(x = 3x^2 + 1)$ |
| $3x^2 + 1$ |
The third example shows that Maple thinks that \( x=0..10 \) is an equation even if it isn't one in the standard mathematical sense.

evalf has an optional second argument. If it's not there, Maple will compute a 10 digit approximation. If the second argument provided is a positive integer, then Maple will compute that many digits.

convert does many things. When it is given four arguments and the second argument it units, then it expects the first argument to be a number, and the third and fourth to be expressions describing the units being converted from and to. Note that the units can be ratios or products rather than just names.

### 7.12 Summary of Chapter 7 material

**Common mathematical functions (see on-line help for index of functions)**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sin, cos, tan, sec, csc, cot</td>
<td>Trigonometric functions</td>
</tr>
<tr>
<td>exp</td>
<td>Exponential function ( \exp(x) ) means ( e^x )</td>
</tr>
<tr>
<td>arcsin, arccos, arctan, arcsec, arccose, arccot</td>
<td>Inverse trigonometric functions</td>
</tr>
<tr>
<td>sinh, cosh, tanh, ....</td>
<td>Hyperbolic trigonometric functions</td>
</tr>
<tr>
<td>ln, log10, log[b]</td>
<td>Logarithm, log base 10, log base b</td>
</tr>
<tr>
<td>min, max</td>
<td>Minimum and maximum</td>
</tr>
<tr>
<td>abs, sqrt</td>
<td>Absolute value, square root</td>
</tr>
<tr>
<td>ceil, floor, trunc, frac</td>
<td>Ceiling, floor, truncation, fractional part</td>
</tr>
</tbody>
</table>

**Command Completion**

When entering math, type the first part of the name of the function and then hit the escape key. A pop-up window will appear with a list of alternative competitions of what you typed. Pick one, and a template will automatically be entered for you. Edit the spots of the template to fit your situation.

\( sol \) (then hit the ESC key)
This is a way to get the computer to automatically type the right kind of delimiters.

**Defining custom functions with the arrow \((\rightarrow)\) notation**

Create a custom function by naming it and its parameters. A custom function defined using arrow notation is of the form:

\[
FunctionName := ParameterName \rightarrow Function
\]

In this case the function name is \(P\) and its sole parameter is \(t\). For multiple parameters, use the form:

\[
FunctionName := (ParameterName1, ParameterName2, ...) \rightarrow Function
\]

To call the function, use the form:

\[
FunctionName(ParameterValues)
\]

To plot the function, set the range of an independent variable, and use it as a parameter to your custom function.

To solve the function for a particular value, use the solve function.

**Troubleshooting function definitions**

Error

<table>
<thead>
<tr>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error, invalid operator parameter name</td>
</tr>
</tbody>
</table>

Error, invalid operator parameter name

<table>
<thead>
<tr>
<th>Examples</th>
</tr>
</thead>
</table>
| \[
F_{bad} = (m_1, m_2, r) \rightarrow \frac{g \cdot m_1 \cdot m_2}{r^2}
\] |

Error, invalid operator parameter name

<table>
<thead>
<tr>
<th>Examples</th>
</tr>
</thead>
</table>
| \[
F_{bad} = (m_1, m_2, r) \rightarrow \frac{g \cdot m_1 \cdot m_2}{2}
\] |
### Troubleshooting function definitions

Using the equality operator (=) instead of the arrow in function definition.

\[
FF := (m_1, m_2, r) = \frac{g \cdot m_1 \cdot m_2}{r^2}
\]

\[
(m_1, m_2, r) = \frac{m_1 \cdot m_2}{r^2}
\]  \hspace{1cm} (7.65)

Although technically not an error, defining a function without using the arrow notation is not prescribed by these notes.

\[
F2(m_1, m_2, r) := \frac{g \cdot m_1 \cdot m_2}{r^2}
\]

\[
(m_1, m_2, r) \rightarrow \frac{m_1 \cdot m_2}{r^2}
\]  \hspace{1cm} (7.66)
8 Chapter 8 Programming with functions

8.1 Chapter overview

In computer languages, functions or procedures are the building blocks for programming. An individual function can be programmed to do a lot, but a single function with a lot of programming can be expensive and time-consuming to build. Often a large program is built by combining functions through daisy chaining -- by making the output of one function the input of another. This is sometimes referred to as functional composition. Frequently used daisy chains can themselves be defined as a function, making them easier to reuse.

Most computer languages extend the concept of functions to go beyond the numbers or formulas that "mathematical functions" provide. In Maple, as in most languages, a function can return other kinds of results. It is fairly common in Maple and other languages to have functions that produce as output a list, an equation, or a string as a result. As we shall see, functions can even return a plot as the result of a function. It is also possible for computer functions to have lists, equations, plots, or strings as inputs.

8.2 Designing functions from context

In doing technical work, we often see functions defined as an equation relating the name of the function, its argument(s), and the function definition. Those are easy to translate into Maple's notation and use. For example if we see in a mathematics book "define \( f(x) = x^2 + 2 \cdot x - 4 \) " then we can just transcribe it into the Maple function notation: \( f := (x) \rightarrow x^2 + 2 \cdot x - 4 \).

In word problems, we have to "read between the lines" and design the function. This requires answering the questions:

a) What will the inputs to the function be? Try to give symbolic names for it.

b) What will the output be? Sometimes to realize what the output is, you can create a worksheet with several steps. If there is only one final result, then that should be the output.

c) How do you calculate the output from the inputs? Hopefully, there's a simple formula that describes this.

We illustrate this process with an example:

**Designing a function for pressure/temperature problems**


The Ideal Gas Law, as stated in *Introduction to Engineering* is:

\[
P \cdot V = n \cdot R \cdot T
\]

\[
P \cdot V = n \cdot R \cdot T \tag{8.1}
\]

where

- \( P \) is pressure in Pascals (Pa)
- \( V \) is volume in \( m^3 \)
- \( n \) is the amount of gas in moles (mol),
- \( T \) is the temperature in degrees K,
- \( R \) is the gas constant, approximately \( \frac{8.31}{[K \cdot mol]} \cdot [Pa] \) .
We want to solve the following problem (actually, various versions of it):

**Problem**

We measure the temperature and pressure of a gas. It has a pressure of 100 \( \text{kPa} \) and a temperature 473 degree Kelvin. Then cool it so it has a pressure of 89.8 \( \text{kPa} \). What is its temperature?.

Finding the answer

First, we do the mathematical thinking and informal calculation that allows us to build a function that will solve all problems of this type:

For a fixed cylinder volume, according to the Ideal Gas Law:

\[
\frac{P_1}{T_1} = \frac{P_2}{T_2}, \quad \text{so} \quad \frac{P_2}{P_1} \cdot T_1 = T_2.
\]

\[
T_2 = \frac{89.8}{100} \cdot 473 = 424.7540000
\]

**Function design**

We see that the problem wants us to calculate a temperature \( T_2 \) given the atmospheric pressure \( P_1 \), the internal pressure \( P_2 \), and the first temperature, \( T_1 \).

The output is \( T_2 \), the inputs are \( P_1, P_2, \) and \( T_1 \). We are free to name the function anything we want since the problem statement doesn't name this. We decide to call it something that reminds us of the purpose.

\[
\text{secondTemp} := (P_1, P_2, T_1) \rightarrow \frac{P_2}{P_1} \cdot T_1
\]

\[
(P_1, P_2, T_1) \rightarrow \frac{P_2T_1}{P_1}
\]
Testing and troubleshooting the function

We see whether we get the intended result with the numbers we've already worked out. Note that if we hadn't done the analysis, we wouldn't have any way of testing what we designed.

\[
\text{secondTemp}(89.8, 100, 473)
\]

526.7260579 \hspace{1cm} (8.3)

That isn't the same result. What did we do wrong? The formula 1.2.2 seems like the right thing. What else could go wrong? Close inspection indicates that the first argument to \( \text{internalTemp} \) is \( P_1 \), which appears in the denominator of the formula for the output. In (1.2.3), that would put the "89.8" in the denominator, but our example had 89.8 in the numerator. Oops, we gave the values in the wrong order for the function. There's nothing wrong except that we should invoke the function with the information given in the correct order:

\[
\text{secondTemp}(100, 89.8, 473)
\]

424.7540000 \hspace{1cm} (8.4)

This result agrees with what we think the result should be from our previous calculations. Note that in testing functions, it may be necessary to work through at least one calculation on your own in order to have a way to verify that the function is working.

Using the function

We are given a different version of the problem:

We measure the temperature and pressure of a gas. It has a pressure of 2000 \( \text{kPa} \) and a temperature 473 degree Kelvin. Then cool it so it has a pressure of 53.6 \( \text{kPa} \). What is its temperature?

Answer:

\[
\text{secondTemp}(2000, 56.6, 473)
\]

13.38590000 \hspace{1cm} (8.5)

Since the answer is in degrees Kelvin, this is only about 14 degrees above absolute zero. That's pretty cold!

The usefulness of alternative function designs

Suppose we had this new problem:

We measure the temperature and pressure of a gas. It has a pressure of 100 \( \text{kPa} \) and a temperature 473 degree \( \text{K} \). We then heat it to 512 degrees Kelvin. What is its pressure then?

Another function designed

A little thought produces the calculation:

\[
P_2 = \frac{P_1 T_1}{T_2} = \frac{100 \\ 473}{452.0} = 108.2452431
\]
This leads to the function definition:

\[
\text{secondPressure} := (\, P1, T1, T2 \, ) \rightarrow \frac{P1}{T1} \frac{T2}{T2}
\]

\[
(P1, T1, T2) \rightarrow \frac{P1 \cdot T2}{T1}
\] (8.6)

We test this (remembering what happened before about the order of arguments)

\[
\text{secondPressure}(100, 473, 512.0) \]

\[
108.2452431
\] (8.7)

**Conclusion**

To develop functions, it helps to have worked through some typical calculations interactively. Once you have realized which quantities you are starting with and named them, and have developed the formula for the calculation using those names, you can create a function definition. You can use the names given in the problem description, or you can make up names based on their purpose. Unlike mathematics, you are not limited to single letters for names of variables or names of functions. Computer programmers know that longer names are often easier to remember or understand.

### 8.3 Function composition: daisy-chaining functions together

In the scripts we have developed so far, we have developed a result through a sequence of actions. These sequences can often be described through functional *composition* -- an expression that chains together several actions. Consider the following example:

**Problem**

On November 1, 2007, one Euro was worth 1.002908434 US dollars. We are buying widgets that cost 30 Euros each and importing them into the US. We then put the widgets into packages that cost .075 dollars. How much should we sell the product in the US if we want a 10% markup from our costs?
**Finding the solution: step 1, first do the calculation interactively.**

Doing this in the style of scripts, we first assign the values to variables, and then do the calculational steps.

\[
\text{convRate} := 1.002908434
\]

\[1.002908434\] (8.8)

\[
\text{costInEuros} := 30
\]

\[30\] (8.9)

\[
\text{pkgCost} := 0.075
\]

\[0.075\] (8.10)

\[
\text{markupPct} := 0.10
\]

\[0.10\] (8.11)

\[
\text{totalCost} := \text{pkgCost} + \text{convRate} \cdot \text{costInEuros}
\]

\[30.16225302\] (8.12)

\[
\text{sellingPrice} := (1 + \text{markupPct}) \cdot \text{totalCost}
\]

\[33.17847832\] (8.13)

We foresee using this calculation several times as the conversion Rate, the manufacturing cost in Europe, and the packaging cost change. We even see that the markup might change. We can try to boil down these steps into a few functions.

**Designing the solution: step 2, design functions to do the calculational steps**

\[
\text{totalCostFunc} := (\text{convRate}, \text{costInEuros}, \text{pkgCost}) \rightarrow (\text{pkgCost} + \text{convRate} \cdot \text{costInEuros})
\]

\[
\text{priceFunc} := (\text{markupPct}, \text{totalCost}) \rightarrow (1 + \text{markupPct}) \cdot \text{totalCost}
\]

\[
(\text{markupPct}, \text{totalCost}) \rightarrow (1 + \text{markupPct}) \cdot \text{totalCost}
\] (8.14)

\[
\text{sellingPriceFunc} := (\text{convRate}, \text{costInEuros}, \text{pkgCost}, \text{markupPct}) \rightarrow \text{priceFunc}(\text{markupPct}, \text{totalCostFunc}(\text{convRate}, \text{costInEuros}, \text{pkgCost}))
\]

\[(\text{convRate}, \text{costInEuros}, \text{pkgCost}, \text{markupPct}) \rightarrow \text{priceFunc}(\text{markupPct}, \text{totalCostFunc}(\text{convRate}, \text{costInEuros}, \text{pkgCost}))\] (8.15)

Note that the way that the third function \text{sellingPriceFunc} is defined, it daisy chains \text{totalCostFunc} and makes the output of its invocation into one of the inputs to \text{priceFunc}.
Testing the solution: step 3, test the building blocks in the order that they are used

We test the first two functions. After we see that they agree with our preliminary version of the calculations, we test the third function that depends on the correctness of the first two.

\[
\text{totalCosts} \leftarrow \text{totalCostFunc}(\text{convRate}, \text{costInEuros}, \text{pkgCost})
\]

\[
30.16225302 \quad (8.16)
\]

\[
\text{priceFunc}(0.10, \text{totalCosts})
\]

\[
33.17847832 \quad (8.17)
\]

\[
\text{sellingPriceFunc}(\text{convRate}, \text{costInEuros}, \text{pkgCost}, \text{markupPct})
\]

\[
33.17847832 \quad (8.18)
\]

We could write a script that just used totalCostFunc and priceFunc, but by designing and using a third function, we reduce the work of handling an instance of the problem to just pasting in the values for the four parameters in one line. This is probably less work than changing four lines of parameters that we have had with our previous approach to scripts.

While function composition is a succinct way of ordering many operations, its advantages are apparent only after the chain is built and tested as working correctly. It maybe easier to develop the chain as a script of assignments and then refactor -- rewrite without changing the meaning -- the script so that it uses user-defined functions to replace some of the chains of assignments. In this situation, the advantage of refactoring is that the user-defined function is much easier to reuse than a sequence of assignments (which must be copied and edited for reuse).

Using the solution: step 4, present a script that defines the functions, then invoke the "answer function" repeatedly to handle various versions of the problem

### Table 8.1: A script that uses functional (chaining), and its use

<table>
<thead>
<tr>
<th>Begin function definitions</th>
<th>End function definitions</th>
</tr>
</thead>
</table>
| \[
\text{totalCostFunc} \leftarrow (\text{convRate}, \text{costInEuros}, \text{pkgCost}) \rightarrow (\text{pkgCost} + \text{convRate} \cdot \text{costInEuros})
\] | \[
\text{sellingPriceFunc} \leftarrow (\text{convRate}, \text{costInEuros}, \text{pkgCost}, \text{markupPct}) \rightarrow \text{priceFunc}(\text{markupPct}, \text{totalCostFunc}(\text{convRate}, \text{costInEuros}, \text{pkgCost}))
\] |
| \[
\text{priceFunc} \leftarrow (\text{markupPct}, \text{totalCost}) \rightarrow (1 + \text{markupPct}) \cdot \text{totalCost}
\] | \[
\text{totalCostFunc} \leftarrow (\text{convRate}, \text{costInEuros}, \text{pkgCost}) \rightarrow (\text{pkgCost} + \text{convRate} \cdot \text{costInEuros})
\] |
| \[
\text{totalCostFunc} \leftarrow (\text{convRate}, \text{costInEuros}, \text{pkgCost}) \rightarrow (\text{pkgCost} + \text{convRate} \cdot \text{costInEuros})
\] | \[
\text{sellingPriceFunc} \leftarrow (\text{convRate}, \text{costInEuros}, \text{pkgCost}, \text{markupPct}) \rightarrow \text{priceFunc}(\text{markupPct}, \text{totalCostFunc}(\text{convRate}, \text{costInEuros}, \text{pkgCost}))
\] |

<table>
<thead>
<tr>
<th>Problem solving</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Version 1</td>
<td>---------------------------</td>
</tr>
</tbody>
</table>

Testing the solution: step 3, test the building blocks in the order that they are used

We test the first two functions. After we see that they agree with our preliminary version of the calculations, we test the third function that depends on the correctness of the first two.

\[
\text{totalCosts} := \text{totalCostFunc}(\text{convRate}, \text{costInEuros}, \text{pkgCost})
\]

\[
30.16225302 \quad (8.16)
\]

\[
\text{priceFunc}(0.10, \text{totalCosts})
\]

\[
33.17847832 \quad (8.17)
\]

\[
\text{sellingPriceFunc}(\text{convRate}, \text{costInEuros}, \text{pkgCost}, \text{markupPct})
\]

\[
33.17847832 \quad (8.18)
\]

We could write a script that just used totalCostFunc and priceFunc, but by designing and using a third function, we reduce the work of handling an instance of the problem to just pasting in the values for the four parameters in one line. This is probably less work than changing four lines of parameters that we have had with our previous approach to scripts.

While function composition is a succinct way of ordering many operations, its advantages are apparent only after the chain is built and tested as working correctly. It maybe easier to develop the chain as a script of assignments and then refactor -- rewrite without changing the meaning -- the script so that it uses user-defined functions to replace some of the chains of assignments. In this situation, the advantage of refactoring is that the user-defined function is much easier to reuse than a sequence of assignments (which must be copied and edited for reuse).

Using the solution: step 4, present a script that defines the functions, then invoke the "answer function" repeatedly to handle various versions of the problem

<table>
<thead>
<tr>
<th>Begin function definitions</th>
<th>End function definitions</th>
</tr>
</thead>
</table>
| \[
\text{totalCostFunc} := (\text{convRate}, \text{costInEuros}, \text{pkgCost}) \rightarrow (\text{pkgCost} + \text{convRate} \cdot \text{costInEuros})
\] | \[
\text{sellingPriceFunc} := (\text{convRate}, \text{costInEuros}, \text{pkgCost}, \text{markupPct}) \rightarrow \text{priceFunc}(\text{markupPct}, \text{totalCostFunc}(\text{convRate}, \text{costInEuros}, \text{pkgCost}))
\] |
| \[
\text{priceFunc} := (\text{markupPct}, \text{totalCost}) \rightarrow (1 + \text{markupPct}) \cdot \text{totalCost}
\] | \[
\text{totalCostFunc} := (\text{convRate}, \text{costInEuros}, \text{pkgCost}) \rightarrow (\text{pkgCost} + \text{convRate} \cdot \text{costInEuros})
\] |
| \[
\text{sellingPriceFunc} := (\text{convRate}, \text{costInEuros}, \text{pkgCost}, \text{markupPct}) \rightarrow \text{priceFunc}(\text{markupPct}, \text{totalCostFunc}(\text{convRate}, \text{costInEuros}, \text{pkgCost}))
\] | \[
\text{totalCostFunc} := (\text{convRate}, \text{costInEuros}, \text{pkgCost}) \rightarrow (\text{pkgCost} + \text{convRate} \cdot \text{costInEuros})
\] |

<table>
<thead>
<tr>
<th>Problem solving</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Version 1</td>
<td>---------------------------</td>
</tr>
</tbody>
</table>
On November 1, 2002, one Euro was worth \( \frac{1}{0.9971} = 1.002908434 \) US dollars. We are buying widgets that cost 30 Euros each and importing them into the US. We then put the widgets into packages that cost .075 dollars. How much should we sell the product in the US if we want a 10% markup from our costs?

\[
sellingPriceFunc(1.002908434, 30, .075, .10)
\]

\( $33.18 \) (8.22)

(We got the number formatted to currency by right-click->Numeric Formatting->Currency.)

Version 2

On November 1, 2007, one Euro was worth 1.4487 US dollars. We are buying widgets that cost 33 Euros each and importing them into the US. We then put the widgets into packages that cost .09 dollars. How much should we sell the product in the US if we want a 10% markup from our costs?

\[
sellingPriceFunc(1.4467, 33, .09, .10)
\]

\( $52.61 \) (8.23)

Version 3

On November 1, 2010, one Euro was worth 1.4728 US dollars. We are buying widgets that cost 35 Euros each and importing them into the US. We then put the widgets into packages that cost .10 dollars. How much should we sell the product in the US if we want an 8% markup from our costs?

\[
sellingPriceFunc(1.4728, 35, .10, .08)
\]

\( $55.78 \) (8.24)

By defining our solution as a sequence of function definitions, we can solve several versions of a problem just by invoking the function with different parameters, without needing to copy lines of calculations.

8.4 Expressions with units of measurements: convert

Maple has facilities for converting between various English and metric units. It is useful for doing multi-step calculations because the conversions happen automatically.

In the first way of using convert, one thinks of a value as implicitly expressing a number of units and wants another number expressing those number of units converted to another unit. One uses \( convert(\text{value}, \text{units}, \text{fromUnit}, \text{toUnit}). \)

<table>
<thead>
<tr>
<th>Examples of unit conversion</th>
</tr>
</thead>
<tbody>
<tr>
<td>How many inches in a meter? How many feet in a kilometer? How many millimeters in a mile?</td>
</tr>
</tbody>
</table>
### Examples of unit conversion

<table>
<thead>
<tr>
<th>Conversion</th>
<th>Result</th>
<th>Note</th>
</tr>
</thead>
</table>
| \(\text{convert}(1,\text{units, inch, meter})\) | \[
\frac{127}{5000}
\]            | (8.25)                                                                             |
| \(\text{convert}(1,\text{units, ft, km})\)   | \[
\frac{381}{1250000}
\]             | (8.26)                                                                             |
| \(\text{convert}(1.0,\text{units, mile, mm})\) | \[1.6093440 \times 10^6\] | Note that the answer to this was expressed as a floating point number rather than a fraction because the input was floating point (1.0) |
| \(\text{convert}(5.4,\text{units, kilowatt, horsepower})\) | \[7.241519284\]                | Maple can convert between most compatible units.                      |
| \(\text{convert}(2.0,\text{units, angstroms, micrometers})\) | \[0.000200000000\] | (8.29)                                                                             |
| \(\text{convert}(15.0,\text{units, miles, hour, second})\) | \[6.70560000\] | (8.30)                                                                             |
| \(\text{convert}(13.3,\text{units, gallons, liters, meter^3})\) | \[65.85005144\] | (8.31)                                                                             |

In many examples in the Maple documentation, some of the arguments to \textit{convert} are \textit{quoted} -- surrounded by apostrophes -- to prevent evaluation from using the value of the names of the units. For example, if you have assigned a value to the variable \(s\), then you cannot convert to seconds with this name without quotation.

### Troubleshooting unit conversion

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\text{seconds} := \text{convert}(3,\text{units, days, seconds}))</td>
<td>[259200]</td>
<td>As long as the various names used as arguments to the \textit{convert} function don't have values, things work fine.</td>
</tr>
<tr>
<td>(\text{seconds2} := \text{convert}(4,\text{units, minutes, seconds}))</td>
<td></td>
<td>Maple performs evaluation of names as it figures out what the inputs to \textit{convert} is. Since \textit{seconds} has a value, Maple tries to compute \textit{convert}(4,\text{units, minutes, 259200}). Since the 4th argument to \textit{convert} has to be a name, an error results.</td>
</tr>
<tr>
<td>(\text{seconds2} := \text{convert}(4,\text{units, minutes, 'seconds'}))</td>
<td>[240]</td>
<td>Quoting the 4th argument causes the name \textit{seconds} to be given as the 4th input to \textit{convert}. This works.</td>
</tr>
</tbody>
</table>
### Troubleshooting unit conversion

**seconds3 := convert(5,'units','hours','seconds')**

\[ 18000 \]

Quoting all the names as a prophylactic measure is acceptable. You see this in a lot of the Maple on-line documentation.

In *Star Wars Episode IV: A New Hope*, Han Solo says that the Millennium Falcon made the Kessel Run in "less than twelve parsecs". We want to know how many days a parsec is.

**convert(12.0,'units','parsecs','days')**

Error, (in convert/units) unable to convert `pc` to `d`

This is the error message you see when you are trying to convert between incompatible units, *e.g.* trying to convert a gallon into a meter. *pc* seems to be Maple's internal name for parsec, *d* the name for days.

**convert(12.0,'units','parsecs','miles')**

\[ 2.300821388 \times 10^{14} \]

A parsec is a non-fictional unit of distance, not time, so we can convert 12 parsecs to miles, kilometers, inches... But we can't convert it to days any more than we can convert inches to volts.

<table>
<thead>
<tr>
<th>Table 8.3: A problem solved with a script using function definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A problem solved, a script built using function definitions</strong></td>
</tr>
</tbody>
</table>

A car travels a 45 miles per hour. How many minutes does it take to travel 900 kilometers?

- **distance := 900.0**
  \[ 900.0 \]  
  (8.36)

- **speed := 45**
  \[ 45 \]  
  (8.37)

- **d := convert(distance,'units','kilometers','miles')**
  \[ 559.2340730 \]  
  (8.38)

- **t := d/speed**
  \[ 12.42742384 \]  
  (8.39)

- **convert(t,'units','hours','minutes')**
  \[ 745.6454304 \]  
  (8.40)

It would pretty obvious how to make a script out of this to handle any problem of the form: A car travels *speed* miles per hour. How many minutes does it take to travel *distance* kilometers... With a few user defined functions, we can get the answer with less typing/cutting/pasting.

We build a sequence of calculations to understand how to solve this problem. This is the informal phase of development, while we are trying to understand what to do. Once we have an idea, we start designing functions and testing them.

The first step was to convert the distance from kilometers to miles. We create a function that does it.

We test it to the result that we got in the script above and see that it agrees.
A problem solved, a script built using function definitions

<table>
<thead>
<tr>
<th>Equation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(d\text{Convert} :) ((\text{distance}) \rightarrow \text{convert(}\text{distance, units, kilometers, miles}))</td>
<td>The next step was to calculate the time (in hours) from the distance in miles and the speed in mph.</td>
</tr>
<tr>
<td>(d\text{Convert}(900))</td>
<td></td>
</tr>
</tbody>
</table>
| \[
\begin{array}{c}
781250 \\
1397
\end{array}
\] | (8.42)                                                                                     |
| \(t\text{Calc} :\) \((d, \text{speed}) \rightarrow \frac{d}{\text{speed}}\) | The test shows that \(t\text{Calc}\) seems to be built correctly. |
| \(t\text{Calc}(1.418, 45)\) |                                                                                      |
| \[
\begin{array}{c}
156250 \\
12573
\end{array}
\] | (8.44)                                                                                     |
| \(t\text{Conv} :\) \((t) \rightarrow \text{convert(}t, \text{units, 'hours', 'minutes'})\) | The third step was to convert the time from hours to minutes. |
| \(t\text{Conv}(1.415)\) |                                                                                      |
| \[
\begin{array}{c}
312500 \\
4191
\end{array}
\] | (8.46)                                                                                     |
| \(\text{solveIt} :\) \((\text{speed, distance}) \rightarrow t\text{Conv}(t\text{Calc}(d\text{Convert}(\text{distance, speed}))\) | The solution function chains together the three functions we've developed. This concludes the development and testing. We present a script and several solved problems in the figure below. |
| \(\text{solveIt}(45, 900.0)\) |                                                                                      |
| \[
745.6454304
\] | (8.48)                                                                                     |

The above table showed the thinking behind the design and testing of the multi-step calculation. However, unless the reader wanted to see how the programming was developed, we wouldn't show the testing or the initial script. We would just the definition of the functions, and the results of repeated invocation of the "solution function".

Table 8.4: Solving multiple versions of a problem through functions

<table>
<thead>
<tr>
<th>Solving multiple versions of a problem through functions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Begin function definitions</td>
<td>The travelSoln function indicates the parameters of the script, (\text{speed}) and (\text{distance}). In setting up the solution method as a function, we lose some intelligibility because of the &quot;inside out&quot; style of following the operation of daisy-chained function composition.</td>
</tr>
<tr>
<td>(d\text{Convert} :) ((\text{distance}) \rightarrow \text{convert(}\text{distance, units, kilometers, miles}))</td>
<td></td>
</tr>
<tr>
<td>(d\text{Convert}(900))</td>
<td></td>
</tr>
</tbody>
</table>
| \[
\begin{array}{c}
781250 \\
1397
\end{array}
\] | (8.49)                                                                                              |
| \(t\text{Calc} :\) \((d, \text{speed}) \rightarrow \frac{d}{\text{speed}}\) |                                                                                              |
| \(t\text{Calc}(1.418, 45)\) |                                                                                              |
| \[
\begin{array}{c}
156250 \\
12573
\end{array}
\] |                                                                                              |
| \(t\text{Conv} :\) \((t) \rightarrow \text{convert(}t, \text{units, 'hours', 'minutes'})\) |                                                                                              |
| \(t\text{Conv}(1.415)\) |                                                                                              |
| \[
\begin{array}{c}
312500 \\
4191
\end{array}
\] |                                                                                              |
| \(\text{solveIt} :\) \((\text{speed, distance}) \rightarrow t\text{Conv}(t\text{Calc}(d\text{Convert}(\text{distance, speed}))\) |                                                                                              |
| \(\text{solveIt}(45, 900.0)\) |                                                                                              |
| \[
745.6454304
\] |                                                                                              |
Solving multiple versions of a problem through functions

\[ t_{\text{Calc}} := (d, \text{speed}) \rightarrow \frac{d}{\text{speed}} \]

\[ (d, \text{speed}) \rightarrow \frac{d}{\text{speed}} \quad (8.50) \]

\[ t_{\text{Conv}} := (t) \rightarrow \text{convert}(t, \text{units}, \text{'hours', 'minutes'}) \]

\[ t \rightarrow \text{convert}(t, \text{units}, \text{'hours', 'minutes'}) \quad (8.51) \]

\[ t_{\text{travelSoln}} := \text{(speed, distance)} \]

\[ \rightarrow t_{\text{Conv}}(t_{\text{Calc}}(d_{\text{Convert}}(\text{distance}), \text{speed})) \]

\[ (\text{speed, distance}) \rightarrow t_{\text{Conv}}(t_{\text{Calc}}(d_{\text{Convert}}(\text{distance}), \text{speed})) \quad (8.52) \]

End of function definitions

Problem version A

A car travels at 45 miles per hour. How many minutes does it take to travel 900 kilometers?

\[ \text{travelSoln}(45, 900.0) \]

\[ 745.6454304 \quad (8.53) \]

Problem version B

A car travels at 45 miles per hour. How many minutes does it take to travel 452 kilometers?

\[ \text{travelSoln}(45, 452.0) \]

\[ 374.4797052 \quad (8.54) \]

Problem version C

A car travels at 65 miles per hour. How many minutes does it take to travel 1500 kilometers?

\[ \text{travelSoln}(65, 1500.0) \]

\[ 860.3601126 \quad (8.55) \]

For casual unit conversion, it can still be useful to rely upon Maple's encyclopaedic knowledge of how to convert units. You can access this through Tools->Assistants->Unit Calculator
8.5 User-defined functions whose inputs and outputs aren't numbers

Although you don't see much mention of this in mathematics texts, it is fairly common while programming to define and use functions that take inputs and produce outputs that are not numbers. For example, if we have a list $L$ of numbers, we can create a function that takes a list as input and produces the average of all the numbers as its output.

Table 8.6: A function that takes a list as its input

<table>
<thead>
<tr>
<th>Example</th>
<th>Commentary</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{average} := L \rightarrow \frac{\sum_{i=1}^{\text{nops}(L)} L[i]}{\text{nops}(L)}$</td>
<td>We are expecting the input to be a list of numbers. As explained in More operations on lists (page 84), $L[i]$ uses indexing to get the $i$-th value of the list, and nops($L$) is the number of elements in the list. (8.56)</td>
</tr>
<tr>
<td>$\text{average}([5, 7, -3, 2, 6])$</td>
<td>When we invoke the function, $L$ is $[5, 7, -3, 2, 6]$, so nops($L$) is 5. (8.57)</td>
</tr>
<tr>
<td>$\frac{17}{5}$</td>
<td></td>
</tr>
<tr>
<td>$\text{average}([5, 7, -3, 2, 2, 6])$</td>
<td>With a different list as input, nops($L$) is 7. Since at least one of the elements of the list was a limited precision number ($5.$ has a decimal point), the limited precision arithmetic is performed with it and subsequent steps of the sum. (8.58)</td>
</tr>
<tr>
<td>$3.375000000$</td>
<td></td>
</tr>
</tbody>
</table>

In analyzing mathematical models as we have been doing, it is also useful to produce abbreviations for common combinations of plot options by creating a function that produces a plot as its result.

Table 8.7: A function that returns a plot as its output, rather than a number

<table>
<thead>
<tr>
<th>Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>We are given two lists of data, $pData$, and $tData$. Plot $pData$ as a function of $tData$, and vice versa.</td>
</tr>
</tbody>
</table>
A function that returns a plot as its output, rather than a number

Solution

Build a function that becomes an abbreviation for the operations in the plot. Provide a third argument that is the string for the color.

\[
\text{PlotIt} := (x\text{Data}, y\text{Data}, c, L) \rightarrow \text{plot}(x\text{Data}, y\text{Data}, \text{style = point, color = c, labels = L})
\]

Plot pressure versus temperature, in red. Note: there seems to be a bug in Maple that suppresses the printing of the horizontal axis label.

\[
\text{PlotIt}(p\text{Data}, t\text{Data}, \text{"red"}, \text{["pressure", "temperature"]})
\]

Plot temperature versus pressure, in red. Copying and pasting the invocation of the \text{PlotIt} function we defined is easier than changing the insides of the original plot operation.
A function that returns a plot as its output, rather than a number

PlotIt(tData, pData, "blue", ["temperature", "pressure"])

Because we have parameterized the data and the axis labels, it's easy to plot pressure versus temperature instead. We don't have to rewrite the directions in PlotIt, we just invoke it with the values and labels in reverse order.

PlotIt(pData, tData, "black", ["pressure", "temperature"])
A function that returns a plot as its output, rather than a number

It is possible to return a list or sequence as a result of a function. Such a function can be put in a chain.

Table 8.8: A problem solved with a function that outputs a sequence of two numbers

<table>
<thead>
<tr>
<th>Problem A</th>
</tr>
</thead>
</table>

A right triangle has a hypotenuse of length 5 feet. The angle between it and one of its sides is 10 degrees. What is the length of the perimeter?

We first build a function that computes the two sides of the right triangle and returns the two values as a sequence. We have to convert degrees into radians in order to do this because the Maple trig functions all use radians.

\[
\text{sideSide} := (\text{hypo}, \text{angle}) \rightarrow (\text{hypo} \cdot \sin(\text{convert(\text{angle}, \text{degrees}, \text{radians}))}, \text{hypo} \cdot \cos(\text{convert(\text{angle}, \text{degrees}, \text{radians}))})
\]

(8.62)

Let's test the sideSide function.

\[
\text{sideSide}(5, 10.0) = [5 \cdot \sin(0.05555555556 \pi), 5 \cdot \cos(0.05555555556 \pi)]
\]

(8.63)

Now, develop a function that takes a sequence of three numbers and adds them together.

\[
\text{sumSides} := (a, b, c) \rightarrow a + b + c
\]

(8.64)

By chaining together the output of sideSide and making it part of the input of sumSides, we can get the whole computation done in one function.

\[
\text{perimeter} := (\text{hypo}, \text{angle}) \rightarrow \text{sumSides}(\text{sideSide}(\text{hypo}, \text{angle}), \text{hypo})
\]

(8.65)

\[
\text{perimeter}(5, 10) = 5 \sin\left(\frac{1}{18} \pi\right) + 5 \cos\left(\frac{1}{18} \pi\right) + 5
\]

at 5 digits

\[
10.792
\]

(8.67)
A problem solved with a function that outputs a sequence of two numbers

Problem B

A right triangle has a hypotenuse of length 10 feet. The angle between it and one of its sides is 42 degrees. What is the length of the perimeter?

\[ \text{evalf}(\text{perimeter}(10,42)) \]

\[ 24.12275432 \]  

Once we have done the work to design and test the functions out on a problem, we can present a script that can solve several different versions of the problem:

Table 8.9: several versions of a function with function definitions

<table>
<thead>
<tr>
<th>Solving several versions of a function with function definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Begin function definitions</strong></td>
</tr>
<tr>
<td>A function that computes the two sides of a right triangle given the angle and the length of the hypotenuse</td>
</tr>
<tr>
<td>[ \text{sideSide} := (\text{hypo}, \text{angle}) \rightarrow (\text{hypo} \cdot \sin(\text{convert}(\text{angle} \cdot \text{degrees}, \text{radians})), \text{hypo} \cdot \cos(\text{convert}(\text{angle} \cdot \text{degrees}, \text{radians}))) ]</td>
</tr>
</tbody>
</table>
| \[ (\text{hypo}, \text{angle}) \rightarrow (\text{hypo} \sin(\text{convert}(\text{angle} \cdot \text{degrees}, \text{radians})), \text{hypo} \cos(\text{convert}(\text{angle} \cdot \text{degrees}, \text{radians}))) \]  
| (8.69) |
| A function that takes a sequence of three numbers and adds them together. |
| \[ \text{sumSides} := (a, b, c) \rightarrow a + b + c \] |
| \[ (a, b, c) \rightarrow a + b + c \]  
| (8.70) |
| Compute the perimeter by summing the three sides. |
| \[ \text{perimeter} := (\text{hypo}, \text{angle}) \rightarrow \text{sumSides}(\text{sideSide}(\text{hypo}, \text{angle}), \text{hypo}) \] |
| \[ (\text{hypo}, \text{angle}) \rightarrow \text{sumSides}(\text{sideSide}(\text{hypo}, \text{angle}), \text{hypo}) \]  
| (8.71) |
| **End function definitions**                                   |

Problem A Solution
Solving several versions of a function with function definitions

A right triangle has a hypotenuse of length 5 feet. The angle between it and one of its sides is 10 degrees. Approximately, what is the length of the perimeter in feet?

\[\text{evalf(perimeter(5, 10))} \]

10.79227965 \hspace{1cm} (8.72)

Problem B Solution

A right triangle has a hypotenuse of length 10 feet. The angle between it and one of its sides is 42 degrees. Approximately, what is the length of the perimeter in feet?

\[\text{evalf(perimeter(10, 42))} \]

24.12275432 \hspace{1cm} (8.73)

### 8.6 Chapter Summary

**Function design**

<table>
<thead>
<tr>
<th>Designing functions from context</th>
<th>a) What will the inputs be?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>b) What will the output be?</td>
</tr>
<tr>
<td></td>
<td>c) How do we calculate the output from the inputs?</td>
</tr>
</tbody>
</table>

**Function composition**

\[ A := (x, y) \mapsto \frac{1}{x} + 3x^3 + 3y \]

\[ (x, y) \mapsto \frac{1}{x} + 3x^3 + 3y \] \hspace{1cm} (8.74)

\[ B := (x, y) \mapsto \frac{3}{A(x, y)} \]

\[ (x, y) \mapsto \frac{3}{A(x, y)} \] \hspace{1cm} (8.75)

\[ B(3, 1) \]

\[ \frac{9}{253} \] \hspace{1cm} (8.76)

**Unit conversion**

Units can be converted directly into compatible units.

\[ \text{convert(1, units, inch, meter)} \]

\[ \frac{127}{5000} \] \hspace{1cm} (8.77)
### Unit conversion

Compound units expressed in ratio form can also be converted into compatible compound units.

\[
\text{convert}(\frac{15.0, \text{units}}{\text{hour}}, \frac{\text{miles}}{\text{second}})
\]

\[6.705600000\]  

(8.78)

Converting between incompatible units will generate an error message.

\[
\text{convert}(3, \text{units}, \text{days}, \text{miles})
\]

Error, (in convert/units) unable to convert ‘d’ to ‘mi’

(8.79)

If we create a variable with the same name as a unit, trying to convert using the variable name will throw an error.

\[
\text{seconds} := 5
\]

\[5\]  

(8.79)

\[
\text{convert}(3, \text{units}, \text{days}, \text{seconds})
\]

Error, (in convert/units) unable to convert ‘d’ to ‘s’

(8.80)

To avoid this, we can use single quotes around the unit names to specify using the unit string, as opposed to the variable value.

\[
\text{convert}(3, \text{units}, \text{'days'}, \text{'seconds'})
\]

\[259200\]  

(8.80)

We can use Maple's built-in unit converter to convert units using drop-down menus.

This tool is located in Tools>Assistants>Unit Calculator

### Non-number inputs and outputs of a function

Using a list of numbers as an input

\[
\text{average} := L \rightarrow \frac{\sum_{i=1}^{\text{nops}(L)} L[i]}{\text{nops}(L)}
\]

\[
\frac{\sum_{i=1}^{\text{nops}(L)} L[i]}{\text{nops}(L)}
\]

(8.81)

\[
\text{average}([5, 7, -3, 2, 6])
\]

\[
\frac{17}{5}
\]

(8.82)

Returning a plot instead of a number

\[
\text{PlotIt} := (\text{xData, yData, c, L}) \rightarrow \text{plot}(\text{xData, yData, style = point, color = c, labels = L})
\]

\[(\text{xData, yData, c, L}) \rightarrow \text{plot}(\text{xData, yData, style = point, color = c, labels = L})\]

(8.83)
### Non-number inputs and outputs of a function

```latex
PlotR([1, 2, 3], [2.1, 2, 1], 'blue', ['x', 'y'])
```

| Returning a sequence of numbers instead of a single number |
|----------------------------------|---------------------------------------------------------------------|
| sideSide := (hypo, angle) \rightarrow (hypo \cdot \sin(\text{convert(\text{angle \cdot degrees}, \text{radians})}), \ hypo \cdot \cos(\text{convert(\text{angle \cdot degrees}, \text{radians})})) | (hypo, angle) \rightarrow (hypo \cdot \sin(\text{convert(\text{angle \cdot degrees}, \text{radians})}), \ hypo \cdot \cos(\text{convert(\text{angle \cdot degrees}, \text{radians})})) |
| sideSide(1, 45) | \[ \frac{1}{2} \sqrt{2}, \frac{1}{2} \sqrt{2} \] |
| | (8.84) |
| | (8.85) |
Chapter 9 Visualization, modeling, and simulation

9.1 Chapter Overview

1. Plot structures are the result of the plot function. They can be assigned to variables through := just as numbers, formulas, lists, and function definitions, can be.

2. The display operation of the plots package allows you to combine together plots. Often times a single picture can be more enlightening or easier to understand than looking at multiple pictures separately.

3. Simulation is the art of predicting the behavior of system entities as they change over time, through the use of mathematical models. It can be as simple as using functions that, given the time \( t \) as input, calculate the position, size, weight, or other changing properties of a situation. With the appropriate mathematics, personal computer or supercomputer-class calculations can be used to come up with reasonably accurate descriptions of phenomena. Computational simulations have become a mainstay of modern engineering because "build it and see" can be much slower and more expensive to do for serious projects.

4. The animate operation of the plots package is explained. Its use is illustrated with a session of question-answering using a mathematical models of moving bodies. Computer-generated animations are another useful tool besides solve, and plot.

9.2 plot structures

Like solve, Maple plot is a function: it has inputs and produces outputs. What kind of output does the plot function produce? In Maple, the result of plot is a special type of result called a plot structure. When you evaluate an expression in Maple that invokes the plot function, a plot structure is created.

As with any expression in Maple, you have the choice of just typing in the expression in and seeing the result of evaluating it, or assigning the value to a variable through :=. Just typing in the plot expression displays the plot picture. When the plot structure is then assigned to a variable , then an ellipsis of the plot structure is displayed rather than the plot picture.

Table 9.1: Plot results, displayed and not displayed

<table>
<thead>
<tr>
<th>Example</th>
<th>Commentary</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{plot}(\sin(x), x = 0..10) )</td>
<td>The plot structure result of this operation is not being assigned to a variable through :=. So Maple displays the structure in a pretty way -- by drawing a picture of all the points and labels established by the plot operation and put into the plot structure.</td>
</tr>
</tbody>
</table>
CommentaryExample

Evaluating an expression and then assigning it to the name \( p \). This does not display the plot. We just see \( \text{PLOT}(...) \) which is a sign that the value of \( p \) is a plot structure.

Typing in \( p \) causes Maple to evaluate it. Since \( p \)'s value is the plot structure assigned to it previously, Maple will display the plot.

9.3 plots[display ] and combining plots

The display function from the plots package takes as its first argument a list of plot structures. It will produce a plot structure that combines all the plots together. \( \text{display} \) is the way to get a multi-plot in a script without doing cutting and pasting of plots.

Table 9.2: Display combines plots

<table>
<thead>
<tr>
<th>display combines plots</th>
<th>Commentary</th>
</tr>
</thead>
<tbody>
<tr>
<td>( timeData := [4.5, 5, 5.5, 6, 6.5, 7, 7.5, 8, 8.5, 9, 9.5, 10, 10.5, 11, 11.5, 12, 12.5, 13] )</td>
<td>Someone we know has used a variant on LeastSquares Curve Fitting, to derive a formula with that fits the data. We wish to plot both the data points and the formula on a single graph.</td>
</tr>
<tr>
<td>( \text{tempData} := [58, 57.5, 57, 56, 55.5, 54, 53, 52, 51, 50, 49, 48.5, 48.5, 47.5, 47.5, 46, 45.5, 45] )</td>
<td></td>
</tr>
<tr>
<td>( \text{timeData} := [4.5, 5, 5.5, 6, 6.5, 7, 7.5, 8, 8.5, 9, 9.5, 10, 10.5, 11, 11.5, 12, 12.5, 13] )</td>
<td></td>
</tr>
<tr>
<td>( \text{tempData} := [58, 57.5, 57, 56, 55.5, 54, 53, 52, 51, 50, 49, 48.5, 48.5, 47.5, 47.5, 46, 45.5, 45] )</td>
<td></td>
</tr>
<tr>
<td>display combines plots</td>
<td>This is the point plot we get from this data.</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>( \text{plot}(\text{timeData, tempData, style = point, color = &quot;red&quot;}) )</td>
<td>( p1 ) is assigned the plot structure of the point plot. Because the result is assigned to the variable ( p1 ), only an ellipsis of the plot structure is displayed rather than the picture of the plot.</td>
</tr>
<tr>
<td>( \text{formula} := 25.0 + 33.61425396e^{-0.06172329065 - 0.2777548079} )</td>
<td>Let's assume that we have a friend who has done curve fitting to ( \text{timeData} ) and ( \text{tempData} ) and gotten this formula. In ENGR 101 Fall 2010, Matlab was used to do exponential curve fitting like this. We could replication that calculation in Maple, but don't show it here. If you're interested in seeing how to do it, ask your instructor about it.</td>
</tr>
<tr>
<td>( \text{plot}(\text{formula}, t = \text{min}(\text{timeData}) .. \text{max}(\text{timeData}), \text{color = &quot;blue&quot;}) )</td>
<td>( p2 ) is assigned another plot structure, the smooth plot of the formula. We skip the step where we show this plot as a separate picture because we are more interested in seeing it combined with the point plot.</td>
</tr>
<tr>
<td>( \text{with(plots)} : )</td>
<td>We load the plots package. Because we ended the line with a colon (:), the list of functions in the package is suppressed. In general, ending a line with a colon suppresses the normal output.</td>
</tr>
</tbody>
</table>
9.4 plottools, lines, and other shapes

The plottools package has a number of functions that are useful for inclusion in visualizations (plots). For example, you can create a line segment of any desired color and line thickness with the `line` function.

One can then use the `display` function to merge together lines, plots, and other shapes.

The on-line documentation on plottools contains links to further description.

Table 9.3: BROKEN - MISSING TITLE!

We create a bit of "modern art" by drawing a circle and a point plot.

```latex
with(plottools);

[arc, arrow, circle, cone, cuboid, curve, cutin, cutout, cylinder, disk, dodecahedron, ellipse, ellipticArc, hemisphere, hexahedron, homothety, hyperbola, icosahedron, line, octahedron, parallelepiped, pieSlice, point, polygon, project, rectangle, reflect, rotate, scale, semitorus, sphere, stellate, tetrahedron, torus, transform, translate]  
```

As the on-line documentation indicates, the first two arguments to `line` are lists indicating the coordinates of the starting point and ending point of the line segment. There are optional arguments that indicate color and line thickness, etc.

\[ l1 := \text{line}([0, 1], [1, 1], \text{color} = \text{"orange"}); \]

\[ \text{CURVES}([[0, 1], [1, 1]], \text{COLOUR}(\text{RGB}, 0.80000000, 0.19607843, 0.19607843)) \]  

\[ l2 := \text{line}([1, 1], [1, 2], \text{color} = \text{"blue"}, \text{thickness} = 20) \]

\[ \text{CURVES}([[1, 1], [1, 2]], \text{COLOUR}(\text{RGB}, 0., 0., 1.00000000), \text{THICKNESS}(20)) \]
We suppress printing of the plot structure CURVES(...) for c3 with a colon because it's long and we don't want to see it, we want to see the picture it describes. Note that Maple exposes its Canadian roots by using "colour".

\[ c3 := \text{circle}(\{2, 2.5\}, 3, \text{color} = "Purple", \text{thickness} = 5) : \]

\[ \text{with(plots)} : \]

\[ \text{display}([11, 12, c3, \text{plot}(2.r^2, t = 0 .. 2, \text{style} = \text{point}, \text{symbol} = \text{circle}, \text{symbolsize} = 30)]) \]

One of the options to plot (and display) is to not show the axes. Another option is to indicate that the scaling should be constrained to be equivalent in both horizontal and vertical directions (to make the circle look like a circle). That gives us an unframed work of art!

\[ \text{display}([11, 12, c3, \text{plot}(2.r^2, t = 0 .. 2, \text{style} = \text{point}, \text{symbol} = \text{circle}, \text{symbolsize} = 30)], \text{axes} = \text{none}, \text{scaling} = \text{constrained}) \]

We can use lines and circles for less frivolous purposes, too.
Table 9.4: Combining a line with other plots using display

<table>
<thead>
<tr>
<th>Line segment</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vertLine := line([12, 25], [12, 58], color = &quot;green&quot;)</td>
<td>We create a plot structure that is a green line segment running between the points (12, 25), and (12, 58).</td>
</tr>
<tr>
<td>CURVES([[12, 25], [12, 58]], COLOUR(RGB, 0., 1.0000000, 0.))</td>
<td>(9.10)</td>
</tr>
<tr>
<td>display([p1, p2, vertLine], title = &quot;data fit curve with attention to t=12&quot;)</td>
<td>We combine the line with the data and curve p1 and p2 we created in combining plots (page 134). This highlights the value of the curve at t=12 minutes, which would be good graphic design if we were to include this in a report on a situation where that time was of special interest.</td>
</tr>
</tbody>
</table>

9.5 Mathematical models and simulation

Mathematical models try to describe a "real" situation in terms of equations and formulae. The point of modeling is to try, through mathematical or computational means, to determine what will happen without having to run experiments in the "real" situation. As Dr. Jay Brockman of the University of Notre Dame says in his book *Introduction to Engineering*:

Some engineering students have been fortunate enough to participate in pre-engineering programs such as the first LEGO(TM) League robotics design competition or American Society of Civil Engineering bridge-building contests. In addition to fostering creative problem-solving skills, such projects also introduce students to the important notion that seemingly good ideas don't always work out in practice. Often in such programs, students have ample opportunity to test and modify their designs before they formally evaluate them. If the design doesn't work, then like a sculptor working with clay, the designer adds something here or removes something there until the design is acceptable.

This cut-and-try methodology is also sometimes used in industry, particularly in circumstances where the design is simple, or where, the risk or cost of failure is low. In many situations, however, there is no second chance in the event of failure. For engineering systems such as buildings, bridges, or airplanes -- top name just a few -- failure to meet specifications could mean a loss of life. For others -- such as the integrated circuit chip -- the cost of fabrication is so high that a company may not be able to afford a second chance. In these situations, it's critical for the engineering team to be highly confident that a design will be acceptable before it's built. To do this, engineers use models to predict the behavior of their designs. A model is an approximation to a real system, such that when actions are performed on the model, it will respond in a manner similar to the real system. Models can have many different forms, ranging from physical prototypes such as a crash-test dummy to complex computer simulations.
We can think of a mathematical model as a kind of virtual system... whose input is a set of variables that represent either aspects of the design or aspects of the environment, and whose output is a set of variables that represent the behavior of the system. Inside is a set of mathematical relationships that describe the operation of the system.


The point of expressing a situation mathematically is to use mathematics and computation to better understand the situation. Usually we are given or derive a formulas that allow us to calculate key properties of the system. For models involving only a few variables, this can involve the following kinds of actions:

1. Get a single number, by evaluating a formula or function.
2. Get a single number, by solving an equation.
3. Gain an understanding of a relationship between one or more entities of interest and the "input variables" by producing a formula.
4. Gain a visual understanding of the relationship by plotting a function, or possibly several plots merged together.
5. Gain an understanding of how a a system changes over time. Rather than computing the value of a variable once, we repeatedly compute the value of the variable at several different points at time. This is called *computational simulation* of the system. We can view plots changing over time by producing an animation.

### Examples of the first four types of computation

<table>
<thead>
<tr>
<th>Third type: producing a formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>In the temperature-pressure data fitting example (page 106) the mathematical model is the formula that expresses the relationship between temperature and pressure. We had only data and no formula to begin with, but we developed the formula using the CurveFitting[LeastSquares] operation. This was a computation of the third type mentioned above.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>First type: Evaluating a formula or function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Once we had the formula, we used the relationship to calculate pressure at several given temperatures. This was a computation of the first type mentioned above.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Second type: solving an equation to get a desired value</th>
</tr>
</thead>
<tbody>
<tr>
<td>We also found a temperature corresponding to a specified pressure by using <em>solve</em> -- a computation of the second kind mentioned above.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fourth type: visualization (plotting)</th>
</tr>
</thead>
<tbody>
<tr>
<td>In the example with an exponential curve fit (page 134), we got a visual impression of how the formula fit the data by combining the point plot of the data and the plot of the formula together -- the fourth kind of computation mentioned above.</td>
</tr>
</tbody>
</table>

We haven't explained how to do the fifth type of computation -- animation -- yet. This will be discussed in upcoming section *Animations* (movies) using *animate* (page 141)

### 9.6 Drawing x-y position as a function of time through parameterized plots

The mathematical models often describe the position of a system entity as a function(s) of time. If the entity's position is two dimensional, then we have two functions, often called $x(t)$ and $y(t)$. We can generate a plot of position for various values of $t$ with a special form of `plot`.

```
plot( [ x-position expression, y-position expression, var = low..high], plot options)
```

will draw a two dimensional graph connecting the (x,y) points traced out for the values of the expression as the variable `var` takes on values between `low` and `high`. 
Table 9.5: Examples of plots where the x and y positions are parameterized by t

<table>
<thead>
<tr>
<th>Examples of plots where the x and y positions are expressions parameterized by t</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example 1</strong></td>
</tr>
<tr>
<td>In this example, we describe the x and y positions as periodic functions of time. Every $\frac{2}{\pi}$ time units, the positions return back to where they were originally.</td>
</tr>
</tbody>
</table>

\[
x_{pos} := (t) \rightarrow 3 \cdot \cos(t) \\
y_{pos} := (t) \rightarrow 2 \cdot \sin(t)
\]  

(9.12)  
(9.13)

We plot the position using parameterized plotting. By default the scaling used for the horizontal and vertical axes are different, but in this case the default makes the graph look misleadingly like a circle when it should look more like an ellipse. To compensate, we use the `scaling=constrained` option to `plot`.

\[
plot([x_{pos}(t), y_{pos}(t), t = 0..2\cdot\pi], scaling=constrained)
\]

Example 2

In this example, we have parameterized expressions for an object shot out of a cannon with horizontal velocity 10 feet/second and vertical velocity 10 feet/second minus the acceleration due to gravity.

\[
x_{pos2} := (t) \rightarrow 10 \cdot t \\
y_{pos2} := (t) \rightarrow 10 \cdot t - \frac{32 \cdot t^2}{2}
\]  

(9.14)  
(9.15)
Examples of plots where the x and y positions are expressions parameterized by t

\[
\text{plot}([xpos2(t), ypos2(t), t = 0 .. 5], \, \text{scaling} = \text{constrained}, \, \text{color} = \text{"blue"}, \, \text{labels} = \text{["x", "y"]})
\]

9.7 A first animation example using animate, animation controls

Let's look at an animation. First we have to create it. We can use the \textit{animate} function of the \textit{plots} package. For the time being, let's not worry the details of why the operation is entered the way it is. Rather we focus on what the animation is trying to do, and how to view it in Maple once it has been created.

Table 9.6: First animation example

This produces a movie of a point moving through the points \((0,0), (1,0.01), (2,0.04), \text{etc. up to (10, 100)}.\)

\[
\text{with(plots)}: 
\]
If we click on the plot, the Maple tool bar changes and shows us animation controls.

Table 9.7: Animation controls

These controls are highly similar to video playback controls found in many applications (e.g. You Tube), so we won't discuss at length here. Note that using them you can:

1. Start playing the animation.
2. Stop the animation.
3. Display only a particular frame, "frozen".
4. Control the number of frames per second it plays.
5. Set it to play once or continually repeat in a loop.

See Graphics->Animation->Animation Toolbar under the Table of Contents of the on-line Maple help.

Right-clicking on the animation will also produce a menu of operations that provide an alternative for controlling the animation.
Table 9.8: Animation pop-up menu

<table>
<thead>
<tr>
<th>Animation pop-up menu</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Animation pop-up menu" /></td>
</tr>
</tbody>
</table>

If we play the animation, we will see a point move upwards in a parabolic path:

Table 9.9: Frames 1, 5, 10, 20, and 25 of the animation

<table>
<thead>
<tr>
<th>Frames 1, 5, 10, 20, and 25 of the animation</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image2" alt="Frames 1, 5, 10, 20, and 25 of the animation" /></td>
</tr>
</tbody>
</table>
9.8 The first animate example, part 2

Now that we've gotten the general idea of what animate's results are like, let's look again at what was entered and look at the details of what was computed. The general form of the operation is:

**General form of the animate operation**

```plaintext
with(plottools):
    animate(plotting function, [ parameters to plotting function], time variable = time range)
```

What this does is to apply the plotting function to the parameters for various values of the time variable. Each value of the time variable used is a separate frame of the movie.

In the previous section we used:

```plaintext
animate(plot, [ [t], [t^2]], style = point, symbolsize = 30, color = "Purple", t = 0 .. 10)
```

In the first frame of the movie, it used the value \( t=0 \) for the time variable \( t \). Thus it plotted

```plaintext
plot([0],[0],style=point, symbolsize = 30, color="Purple"
```

plots a small circle at the coordinate (0,0).

The next time value used is \( t=0.4667 \). The second frame of the movie is the result of

```plaintext
plot([.41667],[.41667^2], style=point, symbolsize = 30, color="Purple"
```

which plots a small circle at the coordinate (.41667, 0.1736138889).

9.9 Designing a function to use in animate

In this section, we explore the design of a function that we use with animate to create an animation.

**Problem**

Create an animation of a circle whose radius expands from \( r=1 \) at time \( t=0 \) to \( r=5 \) at time \( t=10 \).
Solution

The actions we create are a short script that defines a function `drawCircle(t)`. Then it uses it in `animate`. While we there are other ways to use `animate` other than this style, we choose to do things this way because the `drawCircle` function allows us to test the plotting one frame at a time before we try to create the animation.

**Solution, part 1**

```plaintext
with(plots):
with(plottools):

display(circle([0, 0], 1, color = "red"))
```

![Graph of a circle with radius 1](image)

While this seems to do the job that we wish for the starting frame, we have to consider how this will vary as t=0 to t=10. We want the radius of the circle to be 1 when t=0 (as in the figure above), and 5 when t=10. We can do some line fitting to come up with the formula if we can't remember enough high school algebra to figure it out ourselves:

**Figuring out the formula for the radius as a function of time**

\[
\text{radT} := \text{CurveFitting}[\text{LeastSquares}][[0, 10], [1, 5], t]
\]

\[
1 + \frac{2}{5} t \tag{9.16}
\]

```
with(plots):

We plot the formula we get from the data fitting with the data points to convince ourselves that the line agrees exactly with the two points. We'd
```
get a similar result for curve fitting a line any time we fit a line with only two data points.

Testing the solution function

We can create a function that creates a circle of the appropriate size given a value for \( t \):

\[
\text{drawCircle} := (t) \rightarrow \text{display}(\text{circle}(\{0, 0\}, 1 + \frac{2}{5} \cdot t, \text{color} = \text{"red"}));
\]

\[
t \rightarrow \text{plots:-display}(\text{plots:-circle}(\{0, 0\}, 1 + \frac{2}{5} \cdot t, \text{color} = \text{"red"}))
\]
We try this function out at \( t = 0 \).

The function evaluated at \( t = 5 \) looks the same, but we see from the axis labels that it is actually a bigger circle.
Once we are convinced that the function `drawCircle` works for the range of values of \( t \) that we need it to, we can use it in `animate`. The way we invoke `drawCircle` within `animate` is different than when we were using it one frame at a time. The name of the function is kept separate from the parameters, and the range. This delays the creation of any plot structures until `animate` starts computing them.

**Complete script solving the "expanding circle animation" problem**

```plaintext
with(plots):
with(plottools):

drawCircle := (t) -> display(
  [circle([0, 0], 1 + \frac{2}{5}, t,
    color = "red")])

t->plots:-display([plottools:-circle([0, 0], 1 + \frac{2}{5}, t,
    color = "red")])
```

We see the first frame of the animation below. When we click on the animation, the animation toolbar appears and we can then play it. We see the circle expand over time. Unlike the individual plots we created during our testing, all the circles use the same scaling.
Some frames from the expanding circle animation
9.10 Designing a more elaborate animation

Problem

A satellite is in a circular orbit around the earth, at a distance of five earth radii. Create an animation of it circling.

Solution — discussion

We will create a function drawPlanetAndSat(t) that for any time \( t \) draws both the Earth and the satellite at time \( t \). We browse through the on-line plottools package and discover the function disk\((c,r, \text{color}=\ldots)\) that draws a solid disk whose center is at the point \( c \), has radius \( r \), and specified color, in a fashion similar to the circle function we used in the Designing a function to use in animate (page 144). Browsing through on-line help for the color names known to plot reveals that one color is "DarkKhaki". We decide to draw the Earth as a disk of radius 1 centered at \((0,0)\).

```maple
with(plots) :

with(plottools) :

display([disk([0, 0], 1, color = "DarkKhaki")])
```

We decide to represent the satellite as a point. Recall that \((\cos(t), \sin(t))\) describes circular motion moving around a circle of radius \( r=1 \). To parameterize orbital motion at radius 5, we use \((5\cos(t), 5\sin(t))\).

```maple
with(plots) :

with(plottools) :
```
We can combine the two together using `display`.

```
with(plots):
    with(plottools):
    display([disk([0, 0], 1, color = "DarkKhaki"), plot([5*cos(0), 5*sin(0)], style = point, symbolsize = 20, color = "red")])
```

We invoke `display` with a list of two plot structures: the disk, and the point plot.

After we draw this we see the two plots together, but it isn't exactly what we want, because the horizontal and vertical scaling is not equal.
We include the plot option `scaling=constrained` as an extra argument to `display`.

![Diagram](image)

We're almost done. As with the animation design of the previous section, we write a function of \( t \) that describes the disk and the position of the point at time \( t \).

**A function describing a frame to draw at time \( t \)**

```plaintext
with(plots):

with(plottools):

drawEarthAndSat := (t) -> display(
    [disk([0, 0], 1, color = "DarkKhaki"),
     plot([5*cos(t)], [5*sin(t)], style = point,
          symbolsize = 20, color = "red"),
     scaling = constrained])

t -> plots:-display(
    [plottools:-disk([0, 0], 1, color = "DarkKhaki"),
     plot([5*cos(t)], [5*sin(t)], style = plottools:-point, symbolsize = 20, color = "red"),
     scaling = constrained])
```
We test this function out at a few values of $t$. The idea is that the satellite makes one orbit during the period $t=0...2\pi$. 

drawEarthAndSat(0) 

\[ \text{drawEarthAndSat}(0) \]

\[ 0.8 \]
\[ 0.2 \]
\[ -0.2 \]
\[ -0.8 \]

\[ 1 \]
\[ 2 \]
\[ 3 \]
\[ 4 \]
\[ 5 \]

\[ -1 \]
\[ -0.5 \]
\[ 0 \]
\[ 0.5 \]
\[ 1 \]

\[ 4 \]
\[ 3 \]
\[ 2 \]

\[ \text{drawEarthAndSat}(1.5) \]

\[ 0.8 \]
\[ 0.2 \]
\[ -0.2 \]
\[ -0.8 \]

\[ 1 \]
\[ 2 \]
\[ 3 \]
\[ 4 \]
\[ 5 \]

\[ -1 \]
\[ -0.5 \]
\[ 0 \]
\[ 0.5 \]
\[ 1 \]
We now can use this function with `animate` to draw the orbiting satellite.

**Complete script to solve the "orbiting satellite animation problem"**

```
with(plots):

with(plottools):

drawEarthAndSat := (t) -> display([
  display(disk([0, 0], 1, color = "DarkKhaki"),
  plot([5*cos(t), 5*sin(t)], style = point,
       symbolsize = 20, color = "red"), scaling = constrained)

  t->plots:-display([plottools:-disk([0, 0], 1, color = "DarkKhaki"),
    plot([5*cos(t), 5*sin(t)], style = plottools:-point, symbolsize = 20, color = "red"), scaling = constrained])
```

9.11 Solving a problem with the help of animation

Problem
A boy throws a ball straight up in the air with an initial velocity of 15 miles per hour. Once released, the ball's position is described by the function \( x(t) = v_0 t - \frac{1}{2} g t^2 \), where \( v_0 \) is the initial velocity, and \( g \) is the force of gravity, \( g = -\frac{32 \text{ feet}}{\text{sec}^2} \).

(a) How many seconds does the ball stay in the air?

(b) Generate an animation that shows the ball's motion. \( t \) should be measured in seconds, and position in feet from ground level. Assume that the ball starts at 0 feet altitude even though this would be unrealistic for a boy to do unless he was standing in a pit!
(c) Use the animation to determine roughly when the maximum altitude is, and what that altitude is.

(d) Find how fast the initial velocity should be so that the ball goes up over 30 feet.

**Solution**

First convert 15 miles per hour into a velocity in feet per second using the `convert` function first described in the section on problem solving functions in Chapter 7 (page 109).

\[ v_0 := \text{convert}(15, \text{units}, \frac{\text{miles}}{\text{hour}}, \frac{\text{feet}}{\text{second}}) \]

\[ v_0 := 22 \]  

(9.22)

Define the gravitational constant

\[ g := -32 \]

(9.23)

Next, define the function for position.

\[ x := (t) \rightarrow v_0 \cdot t + \frac{g \cdot t^2}{2} \]

\[ x := t \rightarrow v_0 \cdot t + \frac{1}{2} g t^2 \]  

(9.24)

Doing a rough plot of position versus time shows us roughly when the ball will hit the ground. We guess that it might take three seconds.

**Table 9.10: A rough-draft plot of ball versus time.**

\[ \text{plot}(x(t), t = 0..3) \]
That's too much. The answer seems to be about 1.4 seconds, but we can use `solve` to come up with an exact value.

\[ \text{solve}(0 = x(t), t) \]

\[ 0, \frac{11}{8} \]  

(9.26)

There are actually two solutions. The obvious one is when \( t=0 \) and the ball hasn't yet been thrown.

The non-obvious one is the larger one. To get that one, we daisy-chain the `max` function with `solve`.

\[ \text{flightTime} := \text{max}(\text{solve}(x(t) = 0, t)) \]

\[ \frac{11}{8} \]  

(9.27)

This answers part (a) of the problem.

We verify our computation by evaluating \( x \) at that time and finding that the position of the ball really is at altitude 0.

\[ x(\text{flightTime}) \]

\[ 0 \]  

(9.28)

Now we need to make the movie. We need to create a function which for time \( t \), draws a point at the coordinate \((0, x(t))\). To illustrate what we mean, let's compute the position of the ball at \( t=1 \) second.

\[ x(1.0) \]

\[ 6.00000000 \]  

(9.29)

We want the ball to be at position \((0,6)\). A plot command that would do this would be:

**Table 9.11: A plot that draws a ball as a circle at \((0,6)\).**

\[ \text{plot}([0], [6], \text{style} = \text{point}, \text{color} = \text{red}, \text{symbol} = \text{circle}, \text{symbolsize} = 30) \]

We've drawn the circle larger (symbolsize=30) to make it stand out.
Similarly, at t=.5, the ball's position would be at x(0.5):

**Table 9.12: A plot that draws a ball as a time t=0.5.**

```
plot([0], [x(0.5)], style = point, color = red, symbol = circle, symbolsize = 30)
```

Evidently x(.5) is 7.

We create a user-defined function that creates a plot structure as a result.

\[
ballFrame := (t) \rightarrow plot([0], [x(t)], style = point, color = red, symbol = circle, symbolsize = 30)
\]

\[
t \rightarrow plot([0], [x(t)], style = point, color = red, symbol = circle, symbolsize = 30)
\]  

(9.30)

Let's try this out for t=.5 and see if we get the same result.
Table 9.13: Trying out a function that draws the ball at a particular time.

We can now use this function with `animate`. Note that we needed to compute `flightTime` in order to make the movie run exactly as long as it takes for the ball to come back down to the ground.

Table 9.14: Animating the `ballFrame` function to produce a movie of the ball in flight.

By playing the movie, we see that the maximum altitude is about 7.5 feet, at time $t=0.687$ seconds.
This answers part (c) of the problem.

To answer part (d), we need to do more programming. We first modify the plot so that it draws a line segment at 30 feet as well as plotting the position of the ball. We use the `line` function of the plottools package discussed in *plottools, lines, and other shapes* (page 136) to draw a line segment at (-5,30) to (5,30), and to color it green:

\[
\text{height} := 30
\]

\[
\text{with(plottools)}:
\]

\[
pLine := \text{line([-5, height], [5, height], color = "green")}
\]

\[
\text{CURVES([[-5., 30.], [5., 30.]], COLOUR(RGB, 0., 1.0000000, 0.))}
\]

The display function can be used to combine this line with a frame of the movie. Here is an example of this:

Table 9.15: A ball with a vertical target line.

\[
\text{with(plots)}:
\]

\[
\text{display([ballFrame(0.5), pLine])}
\]

The automatic scaling of plot chops off the vertical distance between 6 feet and 0 because there is nothing in this frame that needs that. In the animation, the scale is adjusted so that all frames operate in the same axes.

Now we can create a new user-defined function that plots both the line and the ball.

\[
bballWithLine := (t) \rightarrow \text{display([ballFrame(t), pLine])}
\]

\[
t \rightarrow \text{plots:-display([ballFrame(t), pLine])}
\]

To look at the behavior of the ball at a particular velocity, we can now execute a two line script, consisting of assigning \( v_0 \) to the desired initial velocity, and then the operation that draws the movie.

Table 9.16: A first try at shooting the ball up 30 feet: 15 miles per hour

\[
v0 := \text{convert}\left(15, \text{units}, \frac{\text{miles}}{\text{hour}, \text{second}}\right)
\]

\[
22
\]
As we already have seen, $v_0 = 22$ is not fast enough. We set it higher and recalculate the movie:

Table 9.17: Second try: shoot upwards at 22 feet per second

$v_0 := 50$

animate(ballWithLine, \( [t], t = 0..max(solve(x(t) = 0, t)) \))

We can continue with trial-and-error. All we need to do is vary the velocity parameter. The reusability of the ballWithLine function is heavily exploited.

Third and fourth tries: shoot upwards at 40 and 45 feet per second

$v_0 := 40$

$v_0 := 45$
According to the movies, 45 feet per second seems to be about right. We could get a more precise determination through movie-watching, but for high accuracy we should use more mathematics. In a subsequent chapter, we will introduce additional Maple operations that can calculate the velocity exactly (or a close approximation) without the trial-and-error of movie watching. However, using the movies did give us a better understanding of the phenomenon, and allowed us to do trial-and-error fairly easily.

9.12 Exporting animations and non-animated plots

One operation available in the popup menu is Export. Right-click (or control-click) -> Export -> Graphics Interchange Format will produce an animation file in .gif format. As the animation file is being created, a dialog box will appear asking you to specify the directory where the .gif file should be written. Once created, the file can be included on web pages or other documents.

This feature is also available for ordinary (non-animated) plots. Right-clicking (control-click for Macintosh) will create a file of the plot in .gif, .jpeg, or .ps format. However, .gif file is the only format of the three that is supported by web browsers for animations.
### 9.13 Summary of Chapter 9

#### Combining plots and shapes using `display`

<table>
<thead>
<tr>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supplying the <code>display</code> function with a list of two or more plots will cause those plots to be plotted on top of one another.</td>
<td><code>display([plot1, plot2, plot3])</code></td>
</tr>
<tr>
<td>We can save a plot in a variable to display later.</td>
<td><code>plot4 := plot(sin(x), x = 0 .. 2*Pi)</code></td>
</tr>
<tr>
<td></td>
<td><code>PLOT(...)</code></td>
</tr>
<tr>
<td></td>
<td>(9.39)</td>
</tr>
<tr>
<td>Marking a specific value on the plot can be accomplished using the <code>line</code> or <code>circle</code> function.</td>
<td><code>maxAmp := line([0, 1], [2*Pi, 1], color = &quot;blue&quot;)</code></td>
</tr>
<tr>
<td></td>
<td><code>CURVES([[0, 1, [[6.283185308, 1.]]], COLOUR(RGB, 0, 0, 1.00000000)])</code></td>
</tr>
<tr>
<td></td>
<td><code>display([plot4, maxAmp])</code></td>
</tr>
</tbody>
</table>
### Parameterized plots

For plots that may have values corresponding to 2d positions, we use multiple functions to define both $x(t)$ and $y(t)$.

\[
x_{pos} := (t) \rightarrow 3 \cdot \cos(t)
\]
\[
y_{pos} := (t) \rightarrow 2 \cdot \sin(t)
\]

We put these functions in a list, along with the range of the independent variable $t$, and plot.

\[
plot([x_{pos}(t), y_{pos}(t), t = 0..2\cdot\pi], scaling = constrained)
\]

### Animating plots using animate

Creating an animation

\[
x := (t) \rightarrow 50 \cdot \sin\left(\frac{\pi}{4}\right) \cdot t :
\]
\[
y := t \rightarrow 50 \cos\left(\frac{1 \cdot \pi}{4}\right) t - 16 t^2 :
\]

\[
posPlot := (t) \rightarrow plot([x(t)], [y(t)], style = point, symbol = circle)
\]

\[
animate(posPlot, [t], t = 0..2)
\]

Controlling the animation using the animation tool bar or the animation popup menu

The animation tool bar is located above the workspace window, while the popup menu can be displayed by right-clicking the animation.

Exporting an animation to a graphics file

Right click animation>Export>Graphics Interchange Format
Index

Symbols
! (exclamation point factorial), 11
!!! (execute worksheet), 62
" (double-quote), 71
' (apostrophe), 78
(, 10
), 10
- (minus or dash), 10
:=, 42
= (equation), 20
^ (caret), 10
_ (underscore), 41
` (backquote), 78

π
symbolic constant, 25, 29, 30

, 101

A
abs (absolute value), 96
actual parameter, 71
acute accent marks ('), 78
addition (+), 9
altitude, 156
animate (plots package), 141, 160
animation controls, 142
animations
exporting to GIF format, 162
apostrophe, 120
apostrophes, 78
arccos (inverse of cosine), 25
arcsin (inverse of sine), 25, 96
arctan (inverse of tangent), 25
arguments
of function, 71, 95
arithmetic, 8
approximate, 26
exact, 12
operations, 9
arrow, 101
arrow key
right, 11
assignment, 21
mental model of, 43
of a plot structure to a variable, 133
undoing, 44
assistant
Unit Converter, 123
asterisk, 9
autosaving, 19
average
of numbers in a list, 85
Axiom, 4

B
backquotes, 78
backspace key, 15
backups
retrieving, 19
ball, 155
base of natural logarithm (e), 29
blue, 136
brackets, 58

C
C (programming language), 4, 6, 50
C++ (programming language), 5
Canadian, 84
Canadian roots, 137
Canvas (Maple word processor feature), 38
caret (power), 10
circle, 137, 140, 157
function, 137
circle-X, 7
clickable interface, 74
Colorado, 59
colornames (plot), 150
colors
plotting, 83
comma, 58
command, 37
command-C (copy operation), 15, 61
command-R, 37
switch to math mode, 36
command-T
switch to text entry mode, 36
command-V (paste operation), 15, 61
command-X (cut operation), 15
completion
command, 98
composition
of functions, 113, 116, 117
computing
algebraic, 1
graphical, 1
high performance, 2
logical, 1
numerical, 1
constant
symbolic, 52
control-=, 37
control-C (copy operation), 15, 61
control-click, 20
control-R, 37
  switch to math mode, 36
control-T, 37
  switch to text entry mode, 36
control-V (paste operation), 15, 61
control-X (cut operation), 15
convert, 156
  a list into a sequence, 85
  a list into a string, 85
  between units, 109, 123
copy (editing operation), 15
copy and paste, 61
crash
  Maple, 19
curly braces, 58
currency, 119
current state of a Maple session, 43
cut (editing operation), 15

d
  daisy chaining, 113
  functions, 117
dashed rectangle, 8
days, 120
degrees, 109
delimiters in Maple: ( ) [ ] { } , 13
digits, 26
disk (plottools function), 150
  display, 135, 160
    function, 134, 137
divide and conquer, 63
division (/), 9
documentation
  need for, 1
dot
  centered, 11
double-quotes, 78

e
  e
    from keyboard not a symbolic constant, 29
    natural logarithm and symbolic constant, 25, 29, 30, 38
  Edit->Execute->Selection, 62
  Edit->Execute->Worksheet., 62
dash
  ellipse, 140
  ENGR101, 63
der
  enter, 36, 45, 62
department, 20, 57
derror
  attempting to assign to Pi which is protected, 52
  attempting to assign to 'solve' which is protected, 52
  controlling variable of for loop must be a name, 52
  illegal use of an object as a name, 51
  invalid fraction, 14
  invalid matrix/vector product, 13
  invalid operator parameter name, 105
  invalid power, 72
  invalid sequence, 14
  invalid sum/difference, 12
  logic, 18
  missing operation, 12
  numeric operation: division by zero, 13
  semantic, 17
  unable to match delimiters, 13, 72
  unexpected option, 77
  vocabulary, 17
derror message, 12
  escape key, 38
  escape key (ESC), 98
  Euro, 116
eval, 108
evalf, 109
evaluating
  preventing, 120
  state, 19
evaluation, 42
    at a point, 62
    of a formula, 139
    of a function, 72
    of expressions, 9
  exp (exponential function), 97
exponentiation, 16
Exporting animations, 162
expressions
  algebraic, 20
  from palette, 25

F
  factorial (!), 11
  fence post error, 28
  find something similar, 63
  Format -> Create Document Block, 38
  Formatting
    output
      as currency, 119
  Fortran, 4
  freeze frame
    animation, 142
  fruit flies, 100
  fsolve, 107
  ft, 119
  function
    idea of, 95
    function definition, 104
    function definition alternative, 106
function evaluation, 72
functions
define, 100
designing from word problems, 115
thousands of built-in, 98
fuzzy notions, 63

G
gallons, 109, 120
Imperial, 84
gigaflap, 2
graphical user interface (GUI), 4
Graphics Exchange Format (GIF), 162
green, 136

H
help
on-line, 81, 92, 99
quick, 7
through command completion, 98
using on-line, 100
Help (Maple menu), 38
horsepower, 120
hour, 109, 120
hypotenuse, 127

I
I (square root of -1), 52, 59
Ideal Gas Law, 113
Image (Maple word processor feature), 38
inch, 109, 119
index, 57, 137
inequalities, 57
infin (shortcut for infinity in math mode), 38
Insert (Maple menu item), 38
Insert->2-D Math, 36
Insert->Text, 36
interactive, 5
inverse trig functions, 25

J
Java, 4, 5, 50

K
Kessel Run, 121
kilometer, 119, 121
kilopascals, 114
kilowatt, 120

L
label, 9
labels for axes
plotting, 83
LeastSquares, 106
left arrow key, 15
left hand side, 57
left hand side (right-click menu), 27, 31
lhs, 108
line
function, 136
linear algebra, 13
list, 58, 75
add together items from, 85
count number of items in, 85
one item from, 85
operations on, 84
part of, 85
literals, 109
log10 (base 10 logarithm), 96
logarithm
base e, 25
logic error, 18, 28
Los Alamos National Laboratory, 2

M
Macsyma, 4
Maple, 4
as a word processor, 35
document mode, 7
math entry mode, 35
Mathematica, 4, 6
mathematical model, 1
mathematical models, 138
Matlab, 4, 50
Symbolic Toolkit, 6
max (maximum of arguments), 96, 107
maxdigits, 16
meter, 109, 119
micrometer, 120
mile, 119
millimeter, 119
min (minimum of arguments), 96, 107
minus, 10
minute, 120
mistake
typographical
how to fix, 15
multiple systems, 5
multiplication (*), 9
MuPad, 4, 50

N
name
assignment to, 41
National Forest Service, 61
negation (-), 10

O
Oak Ridge National Laboratory, 2
Octave, 4
omega (Greek letter), 38
Omega (upper case Greek letter), 38
on-line documentation, 83
options
    plot, 108
orange, 136
oscilloscope, 63

P
palette, 25
    Expression, 74, 95, 100, 102, 107
    square root, 25
palettes, 38
parameter, 62
parameterized plots, 139
parentheses, 10, 71
parsec, 121
paste (editing operation), 15
petaflop, 2
Pi, 52
Pi (symbolic constant), 52
pick names, 62
pitfalls
    common
        in Maple, 95
playing
    animation, 142
plot, 108
    a formula, 21
    builder, 80
    colors, 78
    for insight into a mathematical relationship, 139
    labels, 78
    options, 83
    structure, 133
    symbol for point plot, 157
    symbol size for point plotting, 157
plot options, 108
PlotBuilder, 23
plots
    package, 135
plottools, 136
plus, 9
pop-up, 105
power (\(^\)), 10
pressure, 125
preventing
    evaluation, 120
Python, 6

Q
Quick Reference (help menu item), 38
quotation, 120

R
radians, 109
ready (status of Maple), 19
rectangle
    dashed (entry), 8
red, 136
refactor
    code, 118
replay
    animation, 142
restart, 51, 52, 63
reuse
    of software, 1
rhs, 108
right arrow key, 11
right hand side, 57
right hand side (of equation), 27
right hand side (right-click menu), 27, 31
right-click, 20

S
SAGE, 4
scaling=constrained (option to plot), 140
scripting, 61
search field
    for help, 81
select
    entry, 57
    item from a set, 27
select entry (right-click menu), 27, 31
semantic error, 17
sequence, 14, 58, 127
set, 58
    empty, 69
sheep, 59
shortcuts
    keyboard, 38
Show Al Palettes, 38
simplify -> assuming real (clickable operation), 98
simulation, 1
single-quotes, 78
slash, 9
small steps, 63
Solo
    Han, 121
solve, 57, 72, 92, 107
    a system of equations, 85
by right-clicking or control-clicking, 20
numerical, 26
won't work with an assigned variable, 52
solving an equation, 139
speed
animation, 142
sqrt (square root), 96
square root
from palette, 25
Star Wars, 121
state
current of Maple session, 43
state display
of Maple, 19
state of a Maple session, 45
stop playing
animation, 142
sublist, 85
subscript, 37
subtraction (-), 10
supercomputer, 2
symbol
for point plotting, 157
symbolsize, 157

T
Table (Maple word processor feature), 38
temperature, 125
template
textual, 98
teraflap, 2
test, 63
testing, 115
a program, 105, 106
functions, 115
text entry mode, 36
text mode to math mode
can't switch back, 38
textual entry, 71
Tools->Spellcheck, 39
triangle, 127
troubleshooting, 63
assignment, 53
function definitions, 105
textual input, 92
unit conversion, 121
word processing, 38

U
unassign, 43, 44, 51
underscore, 37
underscore (_), 41
Unit Converter assistant, 123

units, 156
Units Calculator, 84

V
value
of an unassigned name, 42
variable
programming, 41
vertical range, 74
View (Maple menu), 38
visualization, 1

W
what you see is what you get, 35
WYSIWYG, 35

Y
yard, 120