Chapter 2: Introduction to Programming
Much of programming is about naming

- We name our data
  - Data: The “numbers” we manipulate
  - We call our names for data variables
    - So variables is what we will use to store and name our data
- We name our programs
- We name our functions
- Quality of names is very important!
  - Enough words to describe data, programs, files, etc.
  - Understandable and descriptive
  - Good names make programs readable
Naming our Encodings: types

- We also give names our encodings
  - Sometimes referred to as *types*
  - Numbers without decimals are called *integers*.
  - Numbers with decimal points are called *floating point* or *floats*.
  - Collections of characters (letters, symbols, digits) are called *strings*.
- Some programming languages are *strongly typed*
  - A variable has to be *declared* to have a type, before any data is associated with it
- Python has types but variables do not have to be declared to have a type
Examples of Types

- Integers: 31,364, 12, -12
- Floats: 34,654.01, 12.998, -1.01, 0.01
- Strings: Joe, 85 5th Street NW

Remember: Inside the computer, these are all just bits
Our programs work with a variety of names

- You will name your *data* *(variables)*
- You will name your *functions*
  - Just like functions you knew in math, like sine
- You will name the data that your functions work on
  - *Inputs*, like the 90 in `sine(90)`
- When we run our programs, the data, functions and inputs will be loaded in memory.
Names for things that are *not* in memory

- A common name that you’ll deal with is a file name
- A file is a collection of bytes, with a name, that resides on some external medium, like a hard disk.
- Files are typed, typically with three letter extensions
  - .jpg files are JPEG (pictures)
  - .wav are WAV (sounds)
Getting familiar with JES

- **Program Area**
  - Text editor where your write your programs
  - Program must be *saved* before it can be *loaded*
  - *Load Program* interprets the program so it can be ran

- **Command Area (console)**
  - Used to issue commands to the computer
  - Type the commands at the prompt `>>>` and hit *Enter* or *Return*

- **Help Area**
  - Select something on the command or program areas and click *Explain* button to get help on that item
Jython Environment for Students

Program Area

Help Area

Command Area
JES

Use Window Layout to get the view you want
Jython Environment for Students

- The *watcher button* opens up a debugger window with tools for watching how the computer executed the program.
- The *stop button* allows you to stop a running program.
**Note:** If JES runs slow, close other applications. Web browsers (like Firefox or Internet Explorer) and iTunes and chat tools, email program, etc., all take up memory. Closing some of them saves memory for JES.
Python understands commands

- We can print values, expressions, anything, using the command `print`
- Format: `print expression`
  - Example: `print 2 + 3`
    - print “Hi there!”
- Python understands a few standard math operations (`+ - * /`)
JES: using + - * /

Adding integers
Adding (concatenating) two strings

Dividing floats

Multiplying integers

Subtracting integers

Printing a string

>>> print 34 + 56
90

>>> print 34.1/46.5
0.7333333333333334

>>> print 22 * 33
726

>>> print 14 - 15
-1

>>> print "Hello"
Hello

>>> print "Hello" + "there"
Hellothere
Math may be surprising sometimes

If you only use integers (numbers without decimal points), Python thinks you only want integers.

```python
>>> print 1.0/2.0
0.5
```
Names can be (nearly) whatever we want

- We use the equal sign ( = ) to name data
  - age = 21
- Names must start with a letter
- Be careful not to use command names for your data, functions, etc.
  - print = 1  (won’t work)
- Case matters  -- *Python is case sensitive: A ≠ a*
  - “Print” is not the same as “print”
  - “myPicture” is not the same as “mypicture”
Command Area Editing

- Up/down arrows walk through *command history*
- You can edit the line at the bottom
  - Just put the cursor at the end of the line before hitting Return/Enter.
Syntax Errors

- All languages have a set of words and grammar rules.
- Python has a set of words as well, they are called reserved words or keywords.
  - These are the commands we will be using to write our programs.
- Python also have a set or grammar or syntax rules.
  - If you type something that Python don’t understand, you will get a syntax error.
- If you try to use a name that Python doesn’t know, you will receive an error message.
Python and Functions

• Python understands functions
• Function: a command that may receive a value, performs a task, and it may return an answer

Examples:
• The `ord` function:  integer = ord (character)
  >> print ord(“A”)
  65 → the ASCII code of the character A
• The `abs` function:  number = abs (number)
  >> print abs (-10)
  10 → the absolute value of -10
More on Functions

- Some functions need input data in order to complete their task
  - The data we give the function to perform the task is often called **input value** or **argument**
  - If we know that the function needs a piece of data to complete the task, but we do not have that piece of data yet, then we say the function is expecting a **parameter**
- Some functions will return a value after they have completed the task
  - This value is usually called the **return value**
JES Functions

• A bunch of functions are pre-defined in JES for sound and picture manipulations
  • `pickAFile()`: gets the complete filename for a file
  • `makePicture()`: loads a picture file in memory
  • `makeSound()`: loads a sound file in memory
  • `show()`: displays a picture file
  • `play()`: plays a sound file

• Some of these functions accept *input* values and some don’t
• Some of these functions return values and some don’t
Example

>>> print pickAFile()
C:\MediaSources\arch.jpg

Path to the file:  C:\MediaSources\arch.jpg
Base File Name:  arch.jpg
File extension:  .jpg
pickAFile() leads to The File Picker!
Example

>>> print makePicture (pickAFile())
Picture, filename C:\MediaSources\arch.jpg height 480 width 360

- To display the picture use `show`. **Problem:** we don’t have a name for this picture.
- Second try:
  >>> file = pickAFile()
  >>> pict = makePicture(file)
  >>> show (pict)
- This can also be done in one line:
  >>> show (makePicture(pickAFile()))
Summary: Picture Functions

- `makePicture(filename)` creates and returns a picture object, from the JPEG file at the filename

- `show(picture)` displays a picture in a window

- We’ll learn functions for manipulating pictures later, like `getColor`, `setColor`, and `repaint`

- **NOTE**: `filename` and `picture` are in this case parameters, because we do not have yet a real value (file or picture) to send to the functions, but we know we will need them when we are ready to use the functions
Sound Functions

- `makeSound(filename)` creates and returns a sound object, from the WAV file at the filename
- `play(sound)` makes the sound play
- We’ll learn more later like `getSample` and `setSample`
Example

```python
>>> soundFile = pickAFile()
>>> print soundFile
C:\MediaSources\aah.wav
>>> sound = makeSound(soundFile)
>>> print (sound)
Sound file: C:\MediaSources\aah.wav number of samples: 43009
>>> play(sound)
>>>print play(sound)
None
```

• **NOTE**: Some functions do not return values
Grabbing media from the Web

- Right-click (Windows) or Control-Click (Mac)
- Save Target As...

- Note: Most images on the Internet are copyrighted.
  - You can download and use them for your use only without permission.
- Warning: do not choose pictures that are too big
  - processing time will take longer
  - 300K (480 x 640) is usually a good size
Media Sources Folder

- Can be found at:
  - [https://www.cs.drexel.edu/~aalban/CS140/MediaSources/](https://www.cs.drexel.edu/~aalban/CS140/MediaSources/)

- Or if you are using the Korman or iCommons laptops:
  - there should be a directory with media files for this course located at **C:\MediaSources**
Naming variables -- recap

- We name data using =
- Examples:
  - >>> firstName = “Joe”
  - >>> age = 19
  - >>> gpa = 3.5
- = in this case does not mean “equal”. The interpretation is in this case “becomes a name for”
- Example:
  - >>> age = 19
    age becomes a name for 19
Naming values

- The association between the name and the data only exists until:
  - The name gets assigned to something else
    Example:
    >>> firstName = "Mike"
    >>> print firstName
    Mike
    >>> firstName = "Sarah"
  - You quit JES: The relationship between name and data only exits during a session of JES
- We can assign names to the results of functions:
  >>> pictureFile = pickAFile()
Rules for names

- Must begin with a letter or underscore
- Can include any number of letters (upper or lower case), digits, or underscores
- Cannot begin with a digit
- Cannot include any characters other than letters, digits, or underscores
- Cannot be any keywords or reserved names
- Spaces are not allowed
- Reminder: pick meaningful, descriptive names!
Naming Conventions

- Sometimes to make a name more descriptive you will need to use more than one word
  - Example: you want a variable for the number of students

- There are two style conventions for your function and variable names. You can use either one, but be consistent through the program:
  - Pothole case: lowercase letters with words separated by underscores.
    - Example: `number_of_students`
  - Camel case: no spaces in between words, words after the first one start with uppercase letter.
    - Example: `numberOfStudents`
Writing a program: Making our own functions

- A program in Python is a collection of one or more functions that perform a useful task.
- To create a function, use the command `def`
- Then, the name of the function, and the names for the input values between parentheses (“(input1)”) 
- End the line with a colon (“:”)
- The body of the function is indented (Hint: Use two spaces)
  - That’s called a block
Blocking is indicated for you in JES

- Statements that are indented the same, are in the same block.
- Statements that are in the same block as where the line where the cursor is are are enclosed in a blue box.
The Most Common JES Error: Forgetting to Load

- Your function does NOT exist for JES until you load it
  - Before you load it, the program is just a bunch of characters.
  - Loading encodes it as an executable function
- Save and Save As
  - You must Save before Loading
  - You must Load before you can use your function
  - Use the extension .py to indicate that this is a Python file

An “Unloaded” function doesn’t exist yet.
A program for playing picked sound files

```python
def pickAndPlay() :
    myfile = pickAFile()
    mysound = makeSound(myfile)
    play(mysound)
```

**Note:** `myfile` and `mysound`, inside `pickAndPlay()`, are *completely different* from the same names in the command area.
Command Area vs. Program Area

- These are two different domains
- We can create a variable in the command area and another one in the program area with the same name, however these two variables are not the same, each one belongs to a different domain.
- For example:
  - Joe Smith from Seattle, WA
  - Joe Smith from Philadelphia, PA
  - Same name, different domains, definitely two different people
- The same idea applies to functions
  - You can have variables with the same names in two different functions
  - Each variable is a different object, with a different domain
A function for displaying picked picture files

def pickAndShow() :
    myfile = pickAFile()
    mypict = makePicture(myfile)
    show(mypict)
What if you forget your variable names?

- Use the function `showVars()` in the command area
A function for a specific sound or picture

```python
def playSound():
    myfile = "FILENAME"
    mysound = makeSound(myfile)
    play(mysound)

def showPicture():
    myfile = "FILENAME"
    mypict = makePicture(myfile)
    show(mypict)
```

You can always replace data with a variable that holds that data—or vice versa.

Put r in front of Windows filenames:
r“C:\mediasources\pic.jpg”
What to do about Windows filenames?

- Python doesn’t like you to use “\” in filenames, like “C:\mediasources\barbara.jpg”

What to do?

- **Option #1**: Put an `r` in front of Windows filenames: `r"C:\mediasources\pic.jpg"

- **Option #2**: Use forward slashes. Python will translate it for you: “C:/mediasources/pic.jpg”
A function that takes input

- To make a program more general, it is useful to write functions that use variables to specify input.
- This kind of generalization is called *abstraction* → general solutions that work in lots of situations.
- The function definition will now have a **parameter** or **input variable**.
- When the function is evaluated we will specify an **input value** or **argument**.

```python
def showNamed(myfile):
    mypict = makePicture(myfile)
    show(mypict)
def playNamed(myfile):
    mysound = makeSound(myfile)
    play(mysound)
```
What can go wrong?

• Did you use the *exact* same names (case, spelling)?
• All the lines in the block must be *indented*, and *indented the same amount*.
  • Indentation is very, very important in Python
• Variables in the command area don’t exist in your functions, and variables in your functions don’t exist in the command area.
• The computer can’t read your mind.
  • It will only do exactly what you tell it to do.
For the lab this week

- We will review how to access the Bb Learn page for the course
- We will work on a small program in JES
- You must:
  - Make sure you know your Drexel userid and password
  - Bring your textbook, you will need it in the lab
  - Bring a thumb drive so you can save your work and take it home