Python I/O

Michael Henry
Standard I/O

• Two ways of writing values:
  1. Expression statements
  2. Print statement
• How do you convert values to strings?
Fancier Output

• The `str()` function is meant to return representations of values which are fairly human-readable.

• The `repr()` is meant to generate representations which can be read by the interpreter.

• For objects which don’t have a particular representation for human consumption, `str()` will return the same value as `repr()`.

```python
x = 10 * 3.25
s = 'The value of x is ' + repr(x) + '.
print s
```
Fancier Output

• **rjust()** right-justifies a string by padding it with spaces on the left
• **ljust()** left-justifies a string by padding it with spaces on the right
• **center()** centers a string based on a specific width
• **zfill()** pads a numeric string on the left with zeros

```python
for x in range(1, 11):
    print repr(x).rjust(2), repr(x*x).rjust(3),
    print repr(x*x*x).rjust(4)

'12'.zfill(5)
'–3.14'.zfill(7)
```
New String Formatting

- `str.format()` performs a string formatting operation.
- For a complete overview of string formatting with `str.format()`, see the Built-in Types Documentation.
Reading and Writing

- **open()** returns a file object, and is most commonly used with two arguments: `open(filename, mode)`.
- `r` = read only, default
- `w` = write only
- `r+` = read/write
- `a` = append
File Object Methods

- `f.read(size)` reads a file and returns it as a string. Size is an optional argument and returns the whole file by default.
- `f.readline()` reads a file and returns one line at a time as a string. The newline character (`\n`) is left at the end.
- `f.readlines(size)` returns a list containing all the lines of data in the file. Only complete lines will be returned.
File Object Methods

- Instead of using `f.read()` to read the whole file it efficient to create a for loop using `f.readline()`.
- `f.write(string)` writes the contents of string to the file. Use `str()` to convert other data types to strings before passing them to `f.write()`
File Object Methods

- `f.tell()` returns an integer giving the file object’s current position in the file, measured in bytes from the beginning of the file.
- `f.seek(offset, from_what)` will advanced to the $n^{th}$ byte of the file from either the beginning (0), the current position (1) or the end (2). Use in conjunction with `f.read()` for precise file reading.
File Object Methods

- When you’re done with a file, call `f.close()` to close it and free up any system resources taken up by the open file. After calling `f.close()`, attempts to use the file object will automatically fail.

- `f.closed` will return a boolean statement, `true` if the file successfully closed and `false` if the file failed to close.
Pickle Module

• Writing a string to a file is easy and writing an int takes a bit more work, but what if you wanted to write a list or a dictionary to a file?
• Python provides a standard module called pickle.
• This module can take almost any Python object and convert it to string representation.
Pickle Module

• This process is known as *pickling* while the reverse is known as *unpickling*.

• `pickle.dump(x, f)`
• `x = pickle.load(f)`
• where `x` is an object and `f` is a file that has been opened for writing.

• The *pickle* module is widely used and developers ensure that new data types can be properly pickled and unpickled.
Questions?