File I/O

Ruth Anderson

UW CSE 160

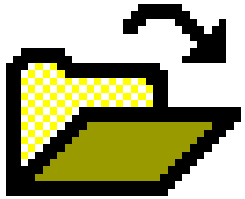
Spring 2015

File Input and Output

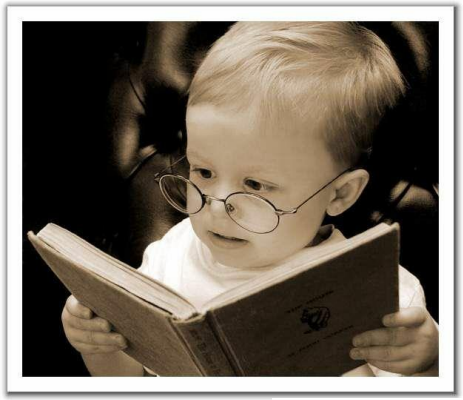
- As a programmer, when would one use a file?
- As a programmer, what does one do with a file?

Files store information when a program is not running

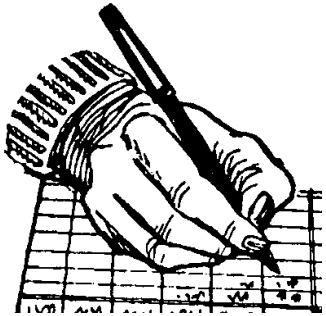
Important operations:



- open a file
- close a file



- read data
- write data



Files and filenames

- A file object represents data on your disk drive
  - Can read from it and write to it
- A filename (usually a string) states where to find the data on your disk drive
  - Can be used to find/create a file
  - Examples:
    - Linux/Mac:`"/home/rea/class/160/lectures/file_io.pptx"`
    - Windows:`"C:\Users\rea\My Documents\cute_dog.jpg"`
    - Linux/Mac: `"homework3/images/Husky.png"`
    - `"Husky.png"`

Two types of filenames

- An Absolute filename gives a specific location on disk:
  `"/home/rea/class/160/15sp/lectures/file_io.pptx"` or
  `"C:\Users\rea\My Documents\homework3\images\Husky.png"`
  - Starts with "/" (Unix) or "C:\" (Windows)
  - Warning: code will fail to find the file if you move/rename files or run your program on a different computer
- A Relative filename gives a location relative to the *current working directory*:
  `"lectures/file_io.pptx"` or `" images\Husky.png"`
  - Warning: code will fail to find the file unless you run your program from a directory that contains the given contents
- *A relative filename is usually a better choice*

Examples

Linux/Mac: These *could* all refer to the same file:

```
"/home/rea/class/160/homework3/images/Husky.png"
"homework3/images/Husky.png"
"images/Husky.png"
"Husky.png"
```

Windows: These *could* all refer to the same file:

```
"C:\Users\rea\My Documents\class\160\homework3\images\Husky.png"
"homework3\images\Husky.png"
"images\Husky.png"
"Husky.png"
```

“Current Working Directory” in Python

The directory from which you ran Python

To determine it from a Python program:

```
>>> import os   # "os" stands for "operating system"
>>> os.getcwd()
'/Users/johndoe/Documents'
```

Can be the source of confusion: where are my files?

Opening a file in python

To open a file for reading:

```
# Open takes a filename and returns a file object.
# This fails if the file cannot be found & opened.
myfile = open("datafile.dat")
```

By default, file is opened for reading

- Or equivalently:

```
myfile = open("datafile.dat", "r")
```

To open a file for writing:

```
# Will create datafile.dat if it does not already
# exist, if datafile.dat already exists, then it
# will be OVERWRITTEN
myfile = open("datafile.dat", "w")
# If datafile.dat already exists, then we will
# append what we write to the end of that file
myfile = open("datafile.dat", "a")
```

Reading a file in python

```
# Open takes a filename and returns a file object.
# This fails if the file cannot be found & opened.
myfile = open("datafile.dat")

# Approach 1: Process one line at a time
for line_of_text in myfile:
   … process line_of_text

# Approach 2: Process entire file at once
all_data_as_a_big_string = myfile.read()

myfile.close() # close the file when done reading
```

Assumption: file is a sequence of lines

Where does Python expect to find this file (note the relative pathname)?

Simple Reading a file Example

```
# Reads in file one line at a time and
# prints the contents of the file.
in_file = "student_info.txt"
myfile = open(in_file)
for line_of_text in myfile:
    print line_of_text
myfile.close()
```

Reading a file Example

```
# Count the number of words in a text file
in_file = "thesis.txt"
myfile = open(in_file)
num_words = 0
for line_of_text in myfile:
    word_list = line_of_text.split()
    num_words += len(word_list)
myfile.close()

print "Total words in file: ", num_words
```

Reading a file multiple times

You can iterate over a **list** as many times as you like:

```
mylist = [ 3, 1, 4, 1, 5, 9 ]
for elt in mylist:
  … process elt
for elt in mylist:
  … process elt
```

Iterating over a **file** uses it up:

```
myfile = open("datafile.dat")
for line_of_text in myfile:
   … process line_of_text
for line_of_text in myfile:
   … process line_of_text
```

This loop body will never be executed!

How to read a file multiple times?

Solution 1: Read into a list, then iterate over it

```
myfile = open("datafile.dat")
mylines = []
for line_of_text in myfile:
  mylines.append(line_of_text)
for line_of_text in mylines:
    … process line_of_text
for line_of_text in mylines:
    … process line_of_text
```

Solution 2: Re-create the file object (slower, but a better choice if the file does not fit in memory)

```
myfile = open("datafile.dat")
for line_of_text in myfile:
   … process line_of_text
myfile = open("datafile.dat")
for line_of_text in myfile:
   … process line_of_text
```

Writing to a file in python

```
#  Replaces any existing file of this name
myfile = open("output.dat", "w")

#  Just like printing output
myfile.write("a bunch of data")
myfile.write("a line of text\n")

myfile.write(4)
myfile.write(str(4))

myfile.close()
```

open for **W**riting (no argument, or "**r**", for **R**eading)

"\n" means end of line (**N**ewline)

Wrong; results in:
`TypeError: expected a character buffer object`

Right. Argument must be a string

close when done with all writing