Chapter 7: Files and Exceptions

Dona Hertel

CS161 – Winter 2013

Portland State University

Storing Your Results Permanently

- Notice that when you end your program, all of your variable values are gone.
- What if you need to keep these values for a later run of your program?
- This is where saving your values to files on disk comes in handy.

Overview of Using Files

- Files are stored on your computer's directories.
- There are two types of files: binary and text.
- we will be dealing with text files.
- Text files have only readable characters in them. They can be opened in a text editor to be modified or read.

Opening a Text File

- Use the builtin function open() to find and open a file.

```
f_in = open(“atreatise.txt”,”r”)
```

- The program finds a file named “atreatise.txt” in your current working directory. This is probably where your program resides.
- and opens it in “read-only” mode (**“r”**). [See Table 7.1 in the textbook (page 193)]
- It then returns a “handle” to the file. This is NOT a list of lines.

Reading From a Text File

- There several ways to get each line from the file:

```
f_in = open(“atreatise.txt”,”r”)
```

```
lines = f_in.readlines()
for line in lines:
    print(line)
```

- Note: this works in both Python 2 and Python 3.

```
for line in f_in:
    print(line)
```

- Note: Only works in Python 3

Closing a File

- It is good practice for both reading and writing to a file to let Python know when you are done using it.

```
f_in = open(“data.txt”,”r”)


... (do something with file)

....


f_in.close()
```

Reading a File Example

- (See moodle site for a zip file of code examples):
  - file_reading.py
  - songs.txt

Writing to a File

- Writing to a file is similar to reading from a file:

```
f_out = open(“data.txt”,”w”)

mydata = [“a”, “b”, “c”]

f_out.writelines(mydata)
f_out.close()
```

Writing to a File Example

- (See moodle site for a zip file of code examples):
  - file_writing.py
  - songs2.txt

What happens if the file can't be opened?

- If for some reason the file can't be opened (ie. not there, no permissions,etc), then the Python interpretor will throw an IOError exception. Use try and except statements to capture the exception and do something with it.

```
try:
    f_in = open(“data.txt”,”r”)
except IOError:
    print(“sorry, we can't open data.txt”)
    sys.exit(0)


lines = f_in.readlines()
```

Can 'catch' other exceptions

- You can also catch other exceptions as well by adding a except statement.

```
try:
    i  = int(input(“enter a number”))
    name = “data”+i+”.txt”
    f_in = open(name,”r”)
except ValueError:
    print(“You, need to enter a number”)
except IOError:
    print(“sorry, we can't open ”, name)
```

try/except example

- (See moodle site for a zip file of code examples):
  - exceptions.py