Introduction To Files In Python

In this section of notes you will learn how to read from and write to files in your programs.

Why Bother With Files?

• Many reasons:
- Too much information to input all at once
- The information must be persistent (RAM is volatile)
- Data entry of information is easier via a text editor rather than through the computer program that you write.
- Etc.

What You Need In Order To Read Information From A File

1. Open the file and associate the file with a file variable
2. A command to read the information

---

1. Opening Files

Prepares the file for reading:
    A. Links the file variable with the physical file (references to the file variable are references to the physical file).
    B. Positions the file pointer at the start of the file.

Format:[1]

    <file variable> = open (*<file name>,* "r")

Example:

  (Constant file name)

  inputFile = open ("data.txt ", "r")

          OR

  (Variable file name: entered by user at runtime)

  filename = raw_input ("Enter name of input file: ")

  inputFile = open (filename, "r")
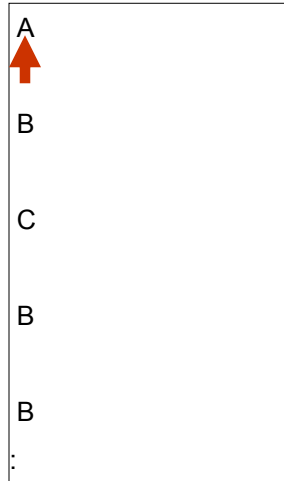
1 Assumes that the file is in the same directory/folder as the Python program.

B. Positioning The File Pointer

letters.txt

```
A
↑
B

C

B

B
:
```

---

2. Reading Information From Files

Typically reading is done within the body of a loop

Format:

for *<variable to store a string>* in *<name of file variable>*:
   <Do something with the string read from a file>

Example:

for line in inputFile:
   print line

Reading From Files: Putting It All Together

A complete version of this program can be found in UNIX under /home/231/examples/files/grades.py:
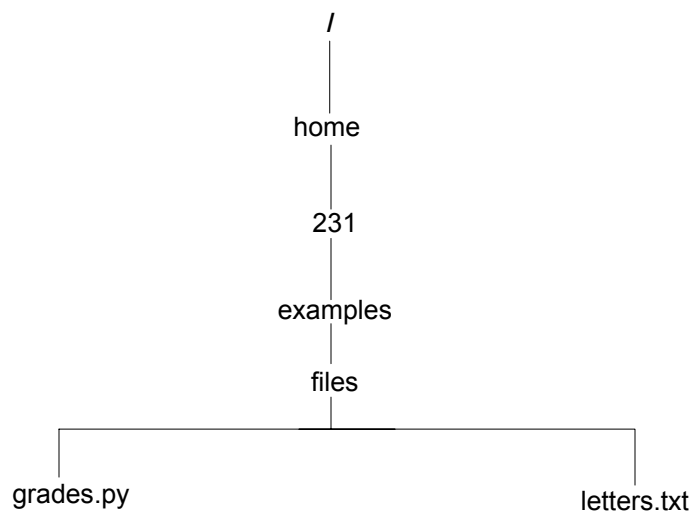
```
inputFileName = raw_input ("Enter name of input file: ")
inputFile = open (inputFileName, "r")
print "Opening file", inputFileName, " for reading."

# While we haven't read past the end of the file continue reading from
# it.
for line in inputFile:
    print line


inputFile.close()
print "Completed reading of file", inputFileName,
```

View Of File Example In UNIX

An Alternate Method Of Getting Input

- Command line arguments: inputs given to a program as it's run.
- The executable program can be found in UNIX under:
  /home/231/examples/files/command_line1.py
- Example: "python command_line.py first_input   2ndInput"

```
<< Filename: command_line1.py >>
import sys
def main ():
    first = sys.argv[0]        # Name of program (command_line.py)
    second = sys.argv[1]       # First input (first_input)
    third = sys.argv[2]        # Second input (2ndInput)

main ()
```

An Alternate Method Of Getting Input (2)

- The executable program can be found in UNIX under:
  /home/231/examples/files/command_line2.py

```
<< Filename: command_line1.py >>
import sys
def main ():
    arguments = sys.argv[0:]
    for argument in arguments:
        print argument
main ()
```

What You Need To Write Information To A File

1. Open the file and associate the file with a file variable
2. A command to write the information

---

1. Opening The File

Format:

<name of file variable> = open (*<file name>*, "w")

Example:

(Constant file name)
outputFile = open ("gpa.txt", "w")

(Variable file name: entered by user at runtime)
outputFileName = raw_input ("Enter the name of the output file to record the GPA's to: ")
outputFile = open (outputFileName, "w")

3. <u>Writing To A File</u>

Format:

  outputFile.write (temp)

Example:

  # Assume that temp contains a string of characters.
  outputFile.write (temp)

<u>Writing To A File: Putting It All Together</u>

A complete version of this program can be found in UNIX under:
/home/231/examples/files/grades2.py

inputFileName = raw_input ("Enter the name of input file to read the grades from: ")

outputFileName = raw_input ("Enter the name of the output file to record the GPA's to: ")

# Open file for reading
inputFile = open (inputFileName, "r")
outputFile = open (outputFileName, "w")

Update user on what is happening.
print "Opening file", inputFileName, " for reading."
print "Opening file", outputFileName, " for writing."

Writing To A File: Putting It All Together (2)

```
gpa = 0
for line in inputFile:
    if (line[0] == "A"):
        gpa = 4
    elif (line[0] == "B"):
        gpa = 3
    elif (line[0] == "C"):
        gpa = 2
    elif (line[0] == "D"):
        gpa = 1
    elif (line[0] == "F"):
        gpa = 0
    else:
        gpa = -1
```

Writing To A File: Putting It All Together (3)

```
    temp = str (gpa)
    temp = temp + '\n'
    print letter[0], '\t', gpa
    outputFile.write (temp)

inputFile.close ()
outputFile.close ()
print "Completed reading of file", inputFileName,
print "Completed writing to file", outputFileName,
```

Another Example Reading From A File Into A String

A complete version of this program can be found in UNIX under:
/home/231/examples/files/file_list.py

```
inputFile = open ("input.txt", "r")
for line in inputFile:
    i = 0
    for ch in line:
        print i, ch,
        i = i + 1
    print
```

Building An Arbitrary Sized List By Reading From File

A complete version of this program can be found in UNIX under:
/home/231/examples/files/file_list2.py

```
inputFile = open ("input2.txt", "r")

myList = []
for line in inputFile:
    myList.append(line)

inputFile.close()
```

Building An Arbitrary Sized List By Reading From File (2)

```
row = 0
for line in myList:
  if (row < 10):
      print row, line,
  else:
      temp = row + 55
      ch = chr(temp)
      print ch, line,
  row = row + 1
```

You Should Now Know

•How to open a file for reading

•How to open a file a file for writing

•The details of how information is read from and written to a file

•How to close a file and why it is good practice to do this explicitly

•What is a command line argument

•How to read and use command line arguments

•How to read from a file of arbitrary size

•How to build an arbitrary sized list by reading the information from a file