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 specialized program (text editor, word processor, spreadsheet, database) 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.

3. A command to close the file.

---

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.
- C. The file may be 'locked'

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

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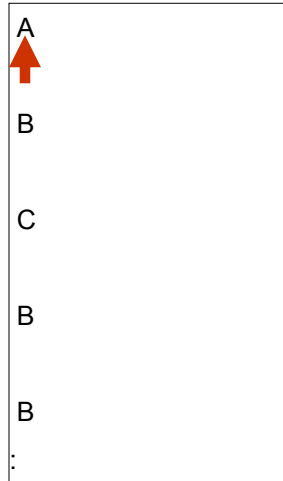
1 Assumes that the file is in the same directory/folder as the Python program.

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

Reading occurs on a line-by-line basis (into a string)

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

Example:
for line in inputFile:
  print(line)

Closing The File

•Although a file is automatically closed when your program ends it is still a good style to explicitly close your file as soon as the program is done with it.

•**Format**:

<name of file variable>.<close>()

•**Example**:

inputFile.close()

Reading From Files: Putting It All Together

Name of the online example: grades1.py

Input files: letters.txt or gpa.txt

```
inputFileName = input ("Enter name of input file: ")
inputFile = open (inputFileName, "r")
print("Opening file", inputFileName, " for reading.")

for line in inputFile:
    sys.stdout.write(line)


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

What You Need To Write Information To A File

1. Open the file and associate the file with a file variable (file is "locked" for writing).
2. A command to write the information.
3. A command to close the file.

---

1. Opening The File

Format[1]:

    <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 = input ("Enter the name of the output file to record the GPA's to: ")
    outputFile = open (outputFileName, "w")

1 Typically the file is created in the same directory/folder as the Python program.

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>

•Name of the online example: grades2.py

•Input file: "letters.txt" (sample output file name: gpa.txt)

```
inputFileName = input ("Enter the name of input file to read the grades from: ")
outputFileName = input ("Enter the name of the output file to record the GPA's
to: ")

inputFile = open (inputFileName, "r")
outputFile = open (outputFileName, "w")

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)

```
    # (Body of for-loop continued)
    temp = str (gpa)
    temp = temp + '\n'
    print (line[0], '\t', gpa)

    outputFile.write (temp)

# Finished writing to file, provide feedback to user and close files.
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: Access Individual Characters

•Name of the online example: file_list1.py

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

i = 1
for line in inputFile:
    print("Line %d vowels:" %i)
    for ch in line:
        if (ch in ('A','a','E','e','I','i','O','o','U','u')):
            sys.stdout.write(ch)
    i = i + 1
    print()

print ("Completed reading of file input.txt")
inputFile.close()
```

Building An Arbitrary Sized List By Reading From File

•Name of the online example: 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):
      temp = str(row) + line
      sys.stdout.write(temp)
   else:
      temp = (row - 10) + ord('A')
      ch = chr(temp)
      temp = ch + line
      sys.stdout.write(temp)
   row = row + 1
```

Error Handling With Exceptions

• Exceptions are used to deal with extraordinary errors.

• Typically these are fatal runtime errors.

• Example: trying to open a non-existent file

Exceptions: File Example

• Name of the online example: file_exception.py
• Input file name: Most of the input files will work e.g. "input1.txt"

```
inputFileOK = False
while (inputFileOK == False):
    try:
        inputFileName = input ("Enter name of input file: ")
        inputFile = open (inputFileName, "r")
    except IOError:
        print ("File", inputFileName, "could not be opened")
    else:
        print ("Opening file", inputFileName, " for reading.")
        inputFileOK = True

        for line in inputFile:
            sys.stdout.write(line)
        print ("Completed reading of file", inputFileName)
        inputFile.close()
        print ("Closed file", inputFileName)
```

Exceptions: File Example (2)

```
# Body of the while loop (continued)
finally:
    if (inputFileOK == True):
        print ("Successfully read information from file", inputFileName)
    else:
        print ("Unsuccessfully attempted to read information from file",
                inputFileName)
```

Exception Handling: Keyboard Input

•Name of the online example: exception_validation.py

```
inputOK = False
while (inputOK == False):
   try:
      num = input("Enter a number: ")
      num = float(num)
   except ValueError:
      print("Non-numeric type entered")
   else:
      inputOK = True
num = num * 2
print(num)
```

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

•How to read from a file of arbitrary size

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

•How exceptions can be used in conjunction with file input