Introduction To Files In Python

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

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.

Format:[1]

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

Example:

  (Constant file name)

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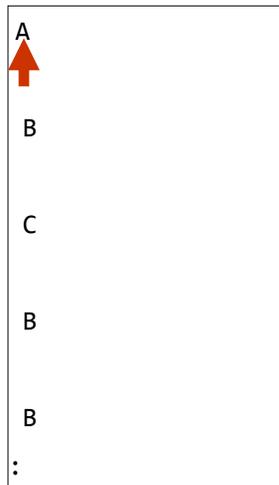
        OR

  (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

2. Reading Information From Files

- Typically reading is done within the body of a loop
- Each execution of the loop will read a line from file 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.
  - What if the program encounters a runtime error and crashes before it reaches the end? The input file may remain 'locked' an inaccessible state because it's still open.
- **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. Writing To A File

- You can use the 'write()' function in conjunction with a file variable.
- Note however that this function will ONLY take a string parameter (everything else must be converted to this type first).

Format:
```
outputFile.write(temp)
```

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

Writing To A File: Putting It All Together

- 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.")
gpa = 0
```

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

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

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

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

Data Processing: Files

- Files can be used to store extensive databases given that there exists a predefined format.
- Format of the example input file: 'employees.txt'
  <Last name><SP><First Name>,<Occupation>,<Income>

James Tam

Example Program: `data_processing.py`

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

print ("Reading from file input.txt")
for line in inputFile:
    name,job,income = line.split(',')
    last,first = name.split()
    for ch in income:
        print(ch)
    input()
    income = int(income)
    income = income + (income * BONUS)
    print("Name: %s, %s\t\t\tJob: %s\t\tIncome $%.2f"
            %(first,last,job,income))

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

 # EMPLOYEES.TXT
Adama Lee,CAG,30000
Morris Heather,Heroine,0
Lee Bruce,JKD master,100000

James Tam

Error Handling With Exceptions

- Exceptions are used to deal with extraordinary errors ('exceptional ones').
- Typically these are fatal runtime errors ("crashes" program)s
- Example: trying to open a non-existent file
- Basic structure of handling exceptions

```
Try:
    Attempt something where exception error may happen
Except:
    React to the error
Else:  # Not always needed
    What do if no error is encountered
Finally:  # Not always needed
    Actions that must be performed
```

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"

```python
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)

```python
# 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)
```

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
- Data storage and processing using files and string functions
- How exceptions can be used in conjunction with file input and with invalid keyboard/console input