

Introduction To Files In Python

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

James Tam

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.

James Tam

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

James Tam

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")
```

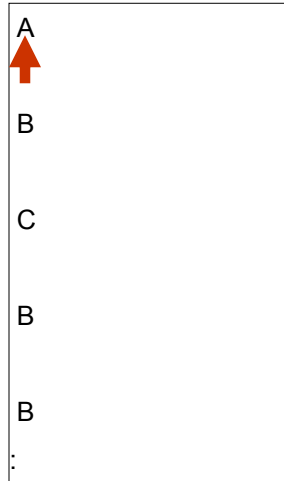
¹ Assumes that the file is in the same directory/folder as the Python program.

James Tam

B. Positioning The File Pointer

letters.txt

```
A
B
C
B
B
:
```



James Tam

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 file>
```

Example:

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

James Tam

Closing The File

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

- **Format:**

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

- **Example:**

```
inputFile.close()
```

James Tam

Reading From Files: Putting It All Together

Name of the online example: grades1.py

```
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)
```

James Tam

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)

James Tam

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

```
outputFile = open (outputFileName, "w")
```

James Tam

3. Writing To A File

Format:

```
outputFile.write (temp)
```

Example:

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

James Tam

Writing To A File: Putting It All Together

- Name of the online example: grades2.py

```
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.")
```

James Tam

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
```

James Tam

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

James Tam

Another Example Reading From A File Into A String: Access Individual Characters

- Name of the online example: file_list.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()
```

James Tam

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()
```

James Tam

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
```

James Tam

str ()

- A special method that is automatically called when an object is passed into the 'print' function.
- Because print takes a string or strings as parameters the `__str__()` method must return a string (print calls `__str__`)
 - Normally the string returned contains information about the attributes.

•Format:

```
def __str__(self, <other parameters>):
    :
    :
    :
    return <string>
```

•Example¹:

```
class Foo:
    x = 12
    def __str__(self):
        return(str(self.x))
```

¹ Full example to follow.

James Tam

Reading File Information Into A List Of Objects

- Name of the online example: client_tracker.py

```
NET_WORTH_CUTOFF = 1000000
```

```
class Client:
```

```
    def __init__(self,name,worth):
        self.name = name
        self.worth = worth

    def __str__(self):
        temp = "Client name: " + self.name
        temp = temp + "\n"
        temp = temp + "Net worth $" + str(self.worth)
        return temp
```

James Tam

Reading File Information Into A List Of Objects (2)

```
class SpecialClient:
```

```
    WORTH_TO_HAPPINESS_RATIO = 0.0001
```

```
    def __init__(self,name,worth):
        self.name = name
        self.worth = worth

        # Amount of money we will spend on the client to keep his/her
        # business (i.e., we will spend more on richer clients).
        self.expenseAccount = int(self.worth * \
            self.WORTH_TO_HAPPINESS_RATIO)
```

```
    def __str__(self):
        temp = "Client name: " + self.name
        temp = temp + "\n"
        temp = temp + "Net worth $" + str(self.worth)
        temp = temp + "\n"
        temp = temp + "Client budget $" + str(self.expenseAccount)
        return temp
```

James Tam

Reading File Information Into A List Of Objects (3)

```
def readClientInformationFromFile():
    clients = []
    inputFile = open("clients.txt", "r")
    for line in inputFile:
        name,worth = line.split('$')
        worth = int(worth)
        if (worth >= NET_WORTH_CUTOFF):
            aClient = SpecialClient(name,worth)
        else:
            aClient = Client(name,worth)
        clients.append(aClient)
    return clients
```

James Tam

Reading File Information Into A List Of Objects (4)

```
def display(clients):
    MAX = len(clients)
    print()
    print("CLIENT LIST")
    print("-----")
    for i in range (0,MAX,1):
        print("Client #%"d" %(i+1))
        print(clients[i])
        print("~~~~~")

def main():
    clients = readClientInformationFromFile()
    display(clients)

main()
```

James Tam

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

James Tam

Exceptions: Example

- Name of the online example: file_exception.py

```
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)
```

James Tam

Exceptions: 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)
```

James Tam

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

James Tam