

# 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:**<sup>1</sup>

*<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: ")
```

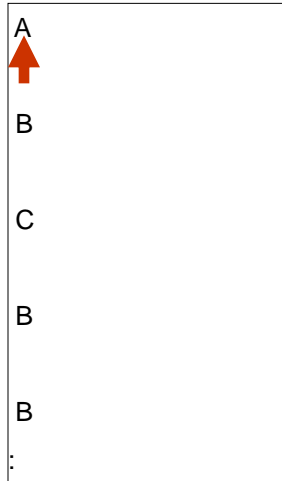
```
inputFile = open (filename, "r")
```

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

James Tam

## B. Positioning The File Pointer

letters.txt



A diagram representing a file named 'letters.txt'. The file contains the following lines of text: 'A', 'B', 'C', 'B', 'B', and a vertical ellipsis '⋮'. A red arrow points to the first line, 'A', indicating the current position of the file pointer.

James Tam

## 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)
```

James Tam

## 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()
```

James Tam

## 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)
```

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<sup>1</sup>:**

```
<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")
```

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

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
- 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.")
```

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

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: 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)
```

James Tam



## **Exceptions: File Example (2)**

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

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