

File Basics, Reading From A File In Python

You will learn the basics of different file formats and different ways of reading from a text file.

James Tam

Storing Information: What You Were Previously Taught

- Information is retained in memory in the form of variables: simple (int, float, bool) or composite/"data structures" (string, list, dictionary, tuple, set).
- This information is stored in memory (RAM) and is only accessible (at most) while the program runs.

James Tam

Storing Information: What You Will Learn

- Information is retained on a storage device (hard drive, portable device etc.) or even remotely (on another device via an Internet connection).
 - Sending data over the Internet can be done by using the “JSON string” format (a python dictionary) – as needed you be provided additional details.
 - Storing information in a file.
 - **Text files:** must be encoded as ASCII or Unicode (the Unicode values are the same as UTF-8).
 - Recall: ASCII only represents values using 8 bits, UTF-8 is used for representing more than just the basic ASCII characters and uses more bits.
 - **Binary files:** “files that aren’t text” but use bits to encode the information (everything else e.g. images, videos, MS-Office files etc.)

James Tam

Sources Of Information: Python Programs

- Hard coded values (unnamed or named constants)
 - E.g.

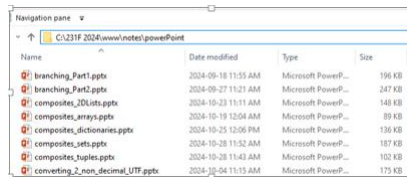
```
age = 12
taxOwed = price * GST
```
- User entered
 - E.g.

```
age = float(input("Type in your age in years: "))
```
- Read in from a file
 - Details come later in this section of notes.

James Tam

Review: Folders/Directories

- Files are contained in a folder/directory.
- Windows



A screenshot of a Windows File Explorer window. The address bar shows the path 'C:\231F 2024\www\notes\powerPoint'. The file list contains the following items:

Name	Date modified	Type	Size
branching_Part1.pptx	2024-09-18 11:55 AM	Microsoft PowerPoint	196 KB
branching_Part2.pptx	2024-09-27 11:21 AM	Microsoft PowerPoint	247 KB
composites_2DLits.pptx	2024-10-23 11:11 AM	Microsoft PowerPoint	148 KB
composites_arrays.pptx	2024-10-19 12:04 AM	Microsoft PowerPoint	89 KB
composites_dictionaries.pptx	2024-10-25 12:06 PM	Microsoft PowerPoint	139 KB
composites_sets.pptx	2024-10-28 11:52 AM	Microsoft PowerPoint	187 KB
composites_tuples.pptx	2024-10-28 11:43 AM	Microsoft PowerPoint	102 KB
converting_2_non_decimal_UTF.pptx	2024-10-04 11:15 AM	Microsoft PowerPoint	175 KB

- Linux/UNIX
 - Example: /home/profs/tam

James Tam

Checking Location

- Before opening a file with a python program you can first determine if the file exists at the specified location using the 'os' library/module/package.

- **Name of the full online example:**

```
1file_checking_information.py
import os
```

An alternative to a hard-coded path C:\231 F 2024..) may involve getting the location from the user.

```
FILE_PATH = "C:/231F 2024/www/notes/powerPoint/"
FILE_NAME = "input_file.txt"

if(os.path.exists(FILE_PATH+FILE_NAME)):
    print("File %s exists in the location %s"
          %(FILE_NAME,FILE_PATH))
else:
    print("File %s not found in location %s"
          %(FILE_NAME,FILE_PATH))
```

James Tam

Location Not Specified

- If the path is not specified for a program then the program opening a file will look in the directory/folder which contains the program that is trying to access the file.
 - If the program is moved then the location will reflect the current location.
- The following examples assume that the input file is in the same folder as the program.

James Tam

First Example: Basic File Read

- **Name of the full online example:**

1file_checking_information.py

```
file_name = input("Type in the complete file name including  
extension (e.g. input.txt): ")
```

```
#aFile: open file and connects program with file  
aFile = open(file_name, "r") #r= read mode
```

```
#Reads a single line  
aLine = aFile.readline()  
print(aLine)  
aFile.close() #Frees the file for access by other programs.  
print("-")
```

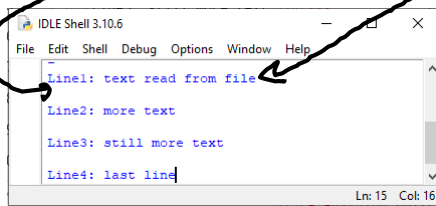
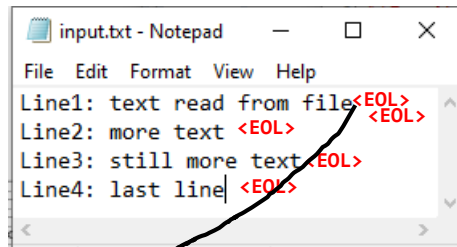
```
#Reads line by line from file from beginning to end.  
for line in aFile:  
    #readline returns the information as a string  
    print(line)
```

James Tam

Previous Example: Double Spaced Output?

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

Recall: `print()`
displays the
specified
information AND
adds an `<EOL>`



James Tam

Removing One Or Both Newlines

- **Approach 1** (you have already been taught): **prevent print from adding an extra newline** after 'line':

```
for line in aFile:  
    print(line, end="")
```
- **Approach 2** (use a built in string method): **strip (remove) any white space characters (space, tab, new line) at the end of the string read from file**:

```
for line in aFile:  
    line = line.rstrip() #rstrip is method of strings  
    print(line, end="")
```

James Tam

An Example Illustrating The String Strip Methods

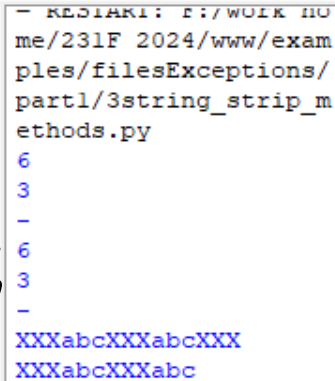
- Name of the full online example:

3string_strip_methods.py

```
aString = "abc \t\n"  
print(len(aString))  
aString = aString.rstrip()  
print(len(aString))
```

```
aString = "  abc"  
print(len(aString))  
aString = aString.lstrip()  
print(len(aString))
```

```
aString = "XXXabcXXXabcXXX"  
print(aString)  
aString = aString.rstrip("X")  
print(aString)
```

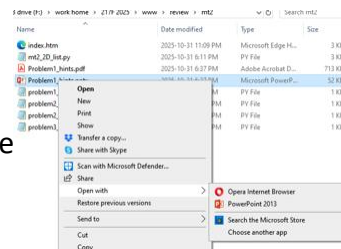


```
- RESIARI: F:/WORK HO  
me/231F 2024/www/exam  
ples/filesExceptions/  
part1/3string_strip_m  
ethods.py  
6  
3  
-  
6  
3  
-  
XXXabcXXXabcXXX  
XXXabcXXXabc
```

James Tam

File Name Extensions/Suffixes

- Usually hidden by the operating system to protect the user (from “doing something bad”).
- The extension/suffix is the part of the file name **after the period** e.g. files_part1_basics_files_file_input.pptx
- Different operating systems use this part of the file name for important purposes.
 - Example, MS-Windows: a file name extension identifies the file type which in turn is associated with a program (which will open the file and allow viewing or even editing – the latter if appropriate).
 - Python programs .py
 - Java programs .java
 - Image files .jpg, .jpeg, .png etc.



James Tam

Other Ways Of Reading From A File

- Name of the full online example:

4file_alternative_reads.py

```
- Read all lines from the file into one continuous string
someText = aFile.read()
print(someText)
print("# char %d" %(len(someText)))

- Read first 'n' (e.g. n=3) characters from the file into a string
someText = aFile.read(3)
print(someText)
print("# char %d of text" %(len(someText)), type(someText))

- Read all lines from the file into a 1D list (line = list element)
someText = aFile.readlines() #aFile.read() same but reads as string
print(someText)
print("# list has %d elements, of type=" %(len(someText)),
      type(someText))

#Determine number characters read ("how far")
aFile.tell() #returns # char read, includes 'appended'
```

James Tam

WHILE-Loop Implementation

- `readline()`: returns an empty string if nothing can be read.
- Pseudo code solution
 - Read line from file
 - If(line is empty) then
 - indicate that file is empty
 - End if
 - Rem: non-empty file**
 - Do while(line read from file is not empty)
 - process line (display contents onscreen)
 - Read another line from file
 - End while

James Tam

WHILE-Loop File Input Example

- Name of the full online example:

5file_read_while_implementation.py

```
EMPTY = ""
file_name = input("Type in the complete file name including
extension (e.g. input.txt): ")
aFile = open(file_name, "r")
#Empty file means readline returns an empty string
line = aFile.readline()
if(line == EMPTY):
    print("File input.txt is empty")
#Continue reading until an empty line read in from file
while(line != EMPTY):
    print(line, end="")
    line = aFile.readline()
aFile.close()
```

James Tam

WHILE-Loop File Input Example

- Name of the full online example:

6file_read_for_implementation_with_empty_file_check.py

```
aFile = open(file_name, "r")
#1st modification over the original FOR-loop example (#2)
line = aFile.readline()
if(line == EMPTY):
    print("File input.txt is empty")
else:
    #2nd modification: process the line just read from file
    #before reading the remaining lines
    print(line, end="")
    for line in aFile:
        print(line, end="")
```

James Tam

Specific Ways To Access A File

- To access a specific location you can 'seek'.
 - **Whence**, you can start the seek at these locations:
 - 0=start of file
 - 1=current location of file pointer
 - 2=end of file
 - **Offset**, the number of characters to move the file pointer off the starting location (if you start the seek from the end it must be a negative number to seek 'backwards' otherwise it will *likely* result in a run-time error as [JT thinks] python won't 'wrap' and read past the end of file to the start).
- **Format:**
`<File variable name>.seek(<offset>, <whence - start location to 'seek'>)`
- **Example:**
`aFile.seek(10,0) #Move 10 characters from start of file.`
`aFile.seek(7,1) #Move 7 characters from current location of file pointer.`
`aFile.seek(-8,2) #Move back 8 characters from the end of file.`

James Tam

Seek: An Application (Zhao/Hudson)

- Consider the file "inputFile.txt".
Each row consists of a 2 character long student ID, and a 4-character long student name.
- Retrieve the ID and name of the third student.

inputFile.txt	
ID	NAME
01	Jack
02	Mack
03	Pham
...	

James Tam

Files - Reading From a File – Example (Zhoa/Hudson)

Program.py

```
filePath = "inputFile.txt"
fileHandler = open(filePath)

for counter in range(3): #skip the first 3 lines
    fileHandler.readline()

s_id = fileHandler.read(2)
fileHandler.read(1)      #skip a space
s_name = fileHandler.read(4)
print("ID: %s, Name: %s" % (s_id, s_name))
```

inputFile.txt

```
ID NAME
01 Jack
02 Mack
03 Pham
...
```

Console output:

```
Student ID: 03, Name: Pham
```



UNIVERSITY OF
CALGARY

Copyright © 2024. Do not distribute outside of the CPSC 231 Fall 2024 class.

After This Section You Should Now Know

- The difference between storing information in RAM (as variables or constants) vs. in storage (in the form of a text file)
- The difference between text vs. binary files.
- How to check for the existence of a file in a particular location.
- File input basics:
 - How to open/close a file.
 - Different ways of reading from a file using different file methods e.g. `readline`.
 - What is a file name extension/suffix and their importance.
 - Reading line by line from a file using a `while` vs. `for` loop.
 - How to check/react to cases of an empty input file.

James Tam

JT's Implementation Of Previous

- Count the number of lines read.
- Since the format of the file is fixed (number of characters in each line, separator is a space character):
 - Read line by line up to the third line.
 - Use a string function/method to split the string
 - Alternatively you could also use an index to access a substring

```
data.txt - Notepad
File Edit Format View Help
ID NAME
01 Jack
02 Mack
03

Python Shell 3.10.6
File Edit Shell Debug Options Window
Python 3.10.6 (tags/v3.10.6:9
AMD64) on win32
Type "help", "copyright", "cr
>>>
===== RESTART: F:
02 desired name: Mack
>>>
```

```
file.py - F:/work home/217F 2025/junk/file.py (3.10.6)
File Edit Format Run Options Window Help
1 count = 1 #inside loop body one line is already read
2 aFile = open("data.txt","r")
3 for line in aFile:
4     if count==3:
5         lineToGet = line
6         count = count + 1
7
8 id, nameToGet = lineToGet.split() #can specify separator, default is a space
9 print(id, 'desired name:', nameToGet)
10
```

Copyright Notification

- Unless otherwise indicated, all images in this presentation were provided courtesy of James Tam.