

File output

You will learn how to write information to a file as well as appending onto the end of a file.

James Tam

File I/O (Input/Output)

- File input
 - Reading information from a file into a program.
- File output
 - Writing information from a program out to a file.
 - Depending upon the mode (append vs. write):
 - the program will add to the end of existing information (append),
 - the program will overwrite an existing information (write).

James Tam

Opening A File For Writing

- First the file must be opened:
 - The 'open' function may be used but with 'w' (**write mode**) instead of read mode.
 - **Format:**
`open(<file name and applicable path>,"w")`
 - **Example:**
`aFile = open("example1_output.txt","w")`
- Effect:
 - If file already exists: previous text is overwritten.
 - If file doesn't exist: a new file is created.

James Tam

Writing Text To A File

- Next a command is needed to write text to the file:
 - One basic method is 'write'.
 - How this file write method differs from the print function:
 - The method argument is written as-is, no extra spaces, new lines.
 - The method argument can only be a string, other types must be converted using the `str` function.
 - (And of course the output is written to a file rather than appearing onscreen).
 - **Format:**
`<Name of file variable>.write(<String to write to file>)`
 - **Example:**
`aFile.write("line1\nline2")`

James Tam

Writing To A File: First Simple, Complete Example

- Name of the full online example:

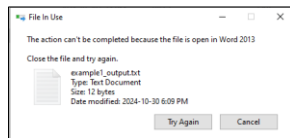
1file_output_basics.py

```
aFile = open("example1_output.txt", "w")
aFile.write("line1\nline2")
aFile.close()
```

James Tam

More On The Close Method

- When opening a file (read/write/append):
 - To prevent multiple processes from accessing the file operating systems may 'lock' the file after a process has opened it.
 - The lock will free up after your program ends.
 - But if the program terminates abnormally (e.g. crashes) then the file may remain locked.



- When writing to a file (write/append):
 - Information to be written does not immediately write to the file.
 - Instead the text is stored in a temporary memory buffer.
 - Closing a file will 'flush' the buffer to write any text remaining in the buffer.
 - If a program ends without flushing the buffer the text stored there may be lost!
 - And of course that memory allocated for the file is wasteful.
 - Morale of the story: **immediately close your files when you're done with them.**

James Tam

Writing Text To The End Of The File

- This can be done by opening the file in 'append' mode.
 - **Format:**
`open(<file name and applicable path>,"a")`
 - **Example:**
`aFile = open("example1_output.txt","a")`
- Effect:
 - If file already exists: adds text to the end of the file (thus 'append')
 - If file doesn't exist: creates a new file.

James Tam

Appending Text To A File

- **Name of the full online example:**
`2file_output_append.py`

`aFile = open("example1_output.txt","a")
aFile.write("line2B")
aFile.close()`

James Tam

Modes Of File Output: Write Vs. Append

- How does python 'handle' the difference between these cases?

```
aFile = open("example1_output.txt", "w")  
aFile = open("example1_output.txt", "a")
```
- How does python 'know' how to traverse line by line through the text in a file.

```
for line in aFile:
```
- Magic? No.
 - How does python 'know' how to access a list that is 'passed as a function argument' even if that list is local?

```
def fun(aList):  
    alist[0] = 12  
  
def start():  
    alist = [1,2]  
    fun(aList)
```

James Tam

Answers

- A list as 'parameter':
 - A list is not passed.
 - Instead a reference to the list (contains the address of the list in 'start') is passed to the function.
 - Inside of 'fun' the address is stored in a local variable that contains that address of the list in 'start'
- Opening a file:
 - A **file pointer** refers to the physical file on disk e.g.

```
aFile = open("example1_output.txt", "w")
```
 - Referring the file variable will act upon the file that has been opened.
- **Read or write:**
 - The file pointer is positioned at the **start** of the file.
- **Append:**
 - The file pointer is positioned at the **end** of the file.

James Tam

Write: Positioning The File Pointer

example1_output

line1



line2

James Tam

Append: Positioning The File Pointer

example1_output

line1

line2



James Tam

Repeatedly Writing To A File

- Unlike the case with file output where the condition to read from the file is based upon the characteristics of the file i.e.
Do-while(end of file not reached)
Continue reading from the input file
End-while
- Examples of external conditions where a file is repeatedly written to:
 - Writing out the contents of a list.
 - Reading information from an input file and writing modified information to an output file.

James Tam

Writing A List To A File

- **Name of the full online example:**
3file_repeated_file_output.py

```
myContacts = ["Stacey Hearn",
              "Jamie Smythe",
              "Jessie Ying",
              "Jessica Gravowski"]
aFile = open("contact_list.txt", "w")
maxContacts = len(myContacts)
i = 0
#Number of file writes depends upon the size of the list
while(i < maxContacts):
    print("Writing information for %s to file."
          %(myContacts[i]))
    aFile.write(myContacts[i] + "!\n")
    i += 1 #Same as i = i + 1
aFile.close()
```

James Tam

Reading From One File, Writing Out To Another

- Name of the full online example:
4file_repeated_file_input_output.py

```
NEWLINE = "\n"
inputFile = open("contact_list.txt","r")
outputFile = open("contact_list_UPDATED.txt","w")
i = 1
#Number of file writes depends upon the input file.
for line in inputFile:
    line = line.rstrip(NEWLINE) #Remove end of line
    modifiedLine = "Contact #" + str(i) + " " + line
    print("Original line: %s,
          Modified line: %s" %(line,modifiedLine))
    outputFile.write(modifiedLine+NEWLINE)
    i += 1
inputFile.close()
outputFile.close()
```

James Tam

File Opening Modes

Mode	File must exist?	Creates file if missing	Can read from file?	Can write to file	File pointer position
r	Yes	No	Yes	No	Start
w	No (overwrites/truncates)	Yes	No	Yes	Start
a	No	Yes	No	Yes	End
r+	Yes	No	Yes	Yes	Start
w+	No (overwrites/truncates)	Yes	Yes	Yes	Start
a+	No (overwrites/truncates)	Yes	Yes	Yes	End

'r' is the default and it will be the mode applied if the 2nd argument is omitted
e.g. aFile = open("data.txt") #Open for reading only

James Tam

After This Section You Should Now Know

- How to open a file for writing as well as appending.
- The difference between write vs. append as well as the consequences of modifying an existing file vs. file output when the file does not yet exist.
- How the `write` method performs file output.
- Why files must be explicitly closed after a program has opened them.
- How a file pointer is used during file input and output.
- Two example conditions for writing to a file.

James Tam

Copyright Notification

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

slide 18

James Tam