GOVERNMENT OF WESTERN AUSTRALIA

TAFE WA South Metropolitan

File Input and Output

(Session 10)

Overview

- Introduction - Python Input and Output (I/O)
- Different File Modes
- Opening File in Python
- Reading Files in Python
- Overwriting Existing Content
- Writing to a Binary File
- Reading Binary File

Introduction to Python I/O

- Python has a built-in function open() and several methods for reading, writing, updating and deleting files
- Syntax: file = **open**(*filename*, *mode*)
- The **open()** function returns a handle (i.e., a file object) to manipulate the specified file
- *filename* – is a string (i.e., a named location of a file)
- Files:
  - Text files (by default) – you can open it with a simple text editor (readable)
  - Binary files (video files, database files, image files, etc.) – sequence of bytes

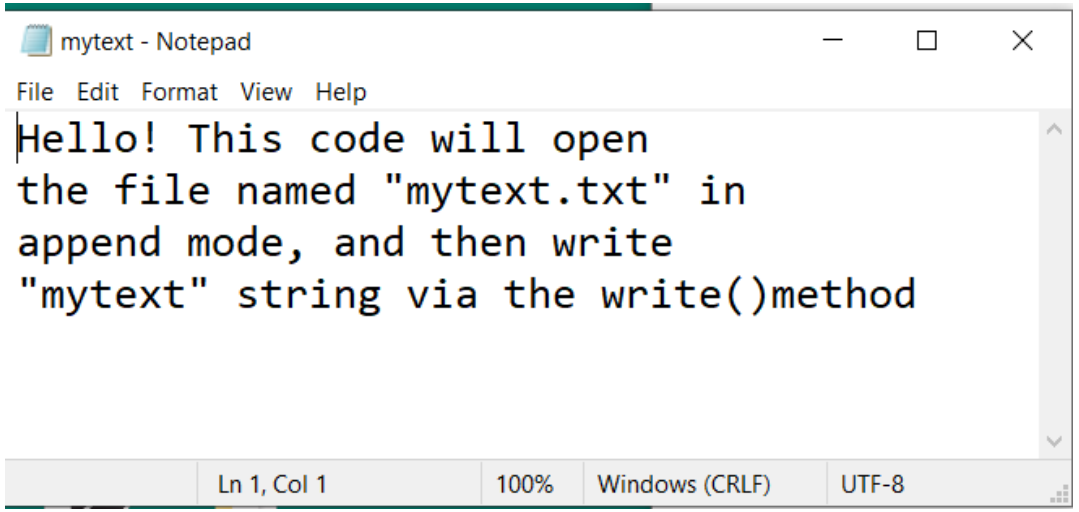
Introduction to Python I/O - Modes

- The **open()** function uses the following modes:
- "**r**" – read (default), opens a file for reading
- "**a**" – append, opens a file for appending, the file is created if does not exist
- "**w**" – write, opens a file for writing, the file is created if does not exist
- "**x**" – create the specified file only if does not exist, otherwise, returns an error (FileExistsError)

How to append text to a file?

```
mytext = """Hello! This code will open
the file named "mytext.txt" in
append mode, and then write
"mytext" string via the write() method"""

# Open a file with access mode 'a'
file = open("mytext.txt", 'a')
# Append 'mytext' at the end of file
file.write(mytext)
# Close the file
file.close()
```



Reading a File

- Assuming that you have “mytext.txt” file located in the same folder as your python code file (e.g., session10.py)

```
# this program will open the file named
# "mytext.txt" using the built-in open() function
# Open a file with access mode 'r'
file = open("mytext.txt", 'r')
# use the read () method
print(file.read())
# Close the file
file.close()
```

```
Hello! This code will open
the file named "mytext.txt" in
append mode, and then write
"mytext" string via the write()method
>>>
```

Read Only Parts of the File

```
# this program will open the file named
# "mytext.txt" with default access mode 'r'
file = open("mytext.txt")     # without mode specified
# print the first 6 characters of the file
print(file.read(6))
# read the rest of the line
file.readline()
# print the second line, note that file handle can be seen as a sequence of strings
print(file.readline(), end="")
# close the file
file.close()
```

```
Hello!
the file named "mytext.txt" in
```

Iterating Over Lines in a File

- A line of a file is a sequence of characters
- Note that a special character called the **newline** (\n) indicates when a line ends
- For loop can be used to iterate over the line of the file

```
# this program will open the file named
# "mytext.txt" and loop through the whole file
file = open("mytext.txt", "r")
# use a for loop to iterate through the file line by line
for i in file:
    print(i, end="")  # each line is a string
# Close the file
file.close()
```

```
Hello! This code will open
the file named "mytext.txt" in
append mode, and then write
"mytext" string via the write()method
```

Activity 1: write a program to counts lines in a file

- Open the file named “mytext.txt” for reading
- Use a for loop to read each line
- Count the lines
- Print out the number of lines

```
file = open("mytext.txt", "?")   # replace ?  with your code
count = 0
...    # your code
...    # your code
print("Line Count:", ?)      # replace ? with you code
# Close the file
file.close()
```

```
Line Count: 4
```

Write to a File in Python

- To write to an existing file:
  - use the “w” mode to *overwrite* the existing content of file
  - use the “a” mode to *append* to the end of file
- To create a new file:
  - use the “w” mode to *open file for writing* (if the specified file does not exist it will create a new file)
  - use the “a” mode to *append to the end of file* (if the specified file does not exist it will create a new file)
  - use the “x” mode *for exclusive creation* (if the specified file exists it will return an error)

Overwriting Existing Content

```
file = open("mytext.txt", "w")
file.write("Now the file has new content!")
file.write("\nThe write method will overwrite any existing content")
file.close()

#open and read the file after the writing:
file = open("mytext.txt", "r")
print(file.read())
```

```
Now the file has new content!
The write method will overwrite any existing content
```

Activity2: append content to a file

- Open the file named “mytext.txt” in append mode **‘a’**
- Using the write() function append to the end of file the following text:
  - mystring = “You can append a new text to the existing file”

```
mystring = "\nYou can append a new text to the existing file"
file = open("mytext.txt", “?") # replace ? with your code
file.write(?)   # replace ? with your code
file.close()
#open and read the file after the appending
file = open("mytext.txt", “?") # replace ? with your code
print(file.read())
```

```
Now the file has new content!
The write method will overwrite any existing content
You can append a new text to the existing file
```

Activity3: writing multiple lines

- create the file “file.txt” in writing mode **‘w’**
- Use the **writelines**() function to write the following lines:
  - “Text File.”, “Writing Multiple Lines.”, “The newline character must be provided as a part of string”

```
lines=["Text File.\n",
      "Writing multiple lines.\n",
      "The newline character must be provided as a part of string.\n"]
file=open("file.txt", “?")    # write your code instead of ?
file.writelines(?)     # write your code instead of ?
file.close()
```

```
file = open("file.txt", "r")
print(file.read())
file.close()
```

```
Text File.
Writing multiple lines.
The newline character must be provided as a part of string.
```

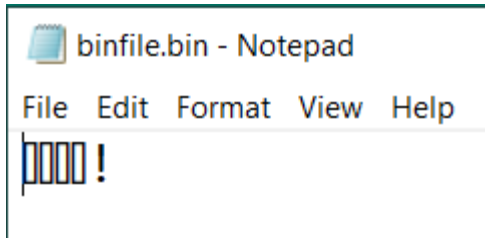
Writing to a Binary File

- To write a file in binary format, use the '**wb**' mode format.
- Binary files are not human-readable
- Example:

```
# open a file in a binary format
# mode is writing
file =open("binfile.bin","wb")
my_list =[1, 7, 18, 27, 31, 33]
# use the built-in function bytearray()
# to get a byte representation
bit_sequence = bytearray(my_list)
file.write(bit_sequence)
file.close()
```

Reading Binary File

```
# open a file in a binary format
# mode is reading, rb
file =open("binfile.bin","rb")
# read a binary file
print(file.read())
# close file
file.close()
```




```
b'\x01\x07\x12\x1b\x1f!'
```

Questions?