Python Programming

Lesson 1 – Sequence and file operations

Lesson 1 - Outline

- File Types in Python
- Basic File Handling Operations
- Other File Handling Operations
- Other Usages

File Types in Python

File Types in Python

- The Python program considers two types of files
  - Binary file
  - Text file

File Types in Python

- **Binary file**
  - **Document files:** .pdf, .docx, .xlsx etc.
  - **Image files:** .png, .jpg, .gif, .bmp etc.
  - **Video files:** .mp4, .3gp, .mkv, .avi etc.
  - **Audio files:** .mp3, .wav, .mka, .aac etc.
  - **Database files:** .mdb, .accde, .frm, .sqlite etc.
  - **Archive files:** .zip, .rar, .iso, .7z etc.
  - **Executable files:** .exe, .dll, .class etc.

File Types in Python

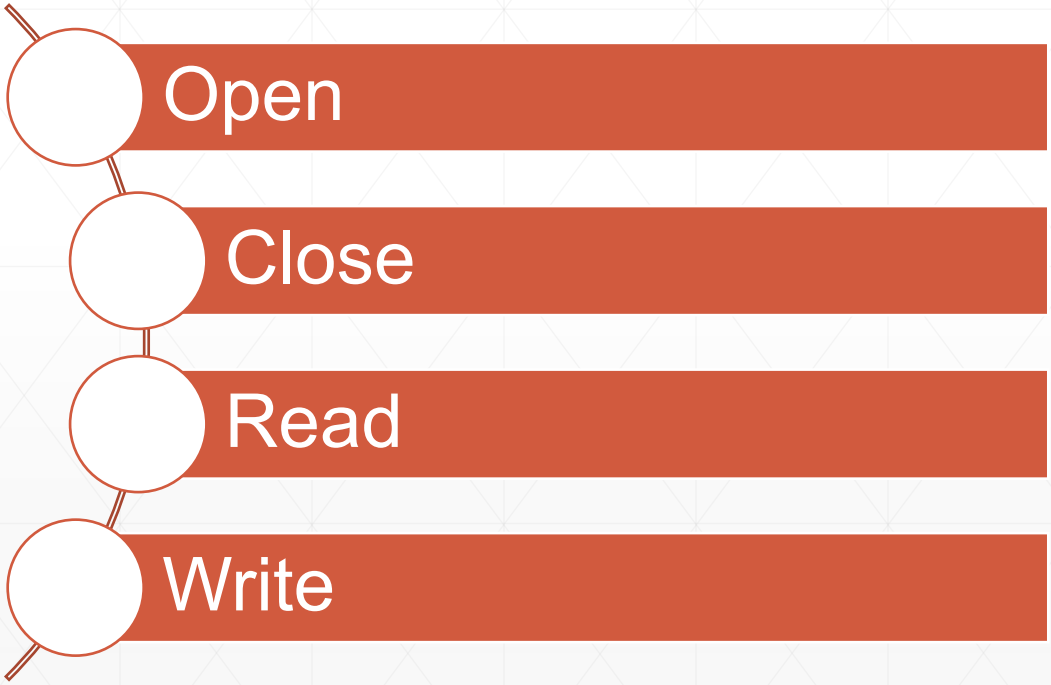
- **Text file**
  - **Web standards:** html, XML, CSS, JSON etc.
  - **Source code:** c, app, js, py, java etc.
  - **Documents:** txt, tex, RTF etc.
  - **Tabular data:** csv, tsv etc.
  - **Configuration:** ini, cfg, reg etc.

Basic File Handling Operations

Basic File Handling Operations <By Human>

Basic File Handling Operations <By Python>

- Basic File Handling Operations in Python



Open

To open a file using the built-in open() function.

Basic File Handling Operations - Open a file

- **open**
- To open a file in read or write mode use the built-in **open()** function
- This function returns a file object, called a handle which can be used to read or modify the file

Basic File Handling Operations - Open a file

- **open**
- **Syntax**

file_object = open("filename", "mode")

1) file_object is the variable to add the file object
2) filename is the file name (optional with filepath)
3) mode is optional because a default value of **'r'** will be assumed if it is omitted

Basic File Handling Operations - Open a file

file_object = open("filename", "mode")

Mode to open a text file:

- **'r' – Read Mode:** Read mode is used only to read data from the file.
- **'w' – Write Mode:** This mode is used when you want to write data into the file or modify it.
  (Remember write mode overwrites the data present in the file.)
- **'a' – Append Mode:** Append mode is used to append data to the file.
  (Remember data will be appended at the end of the file pointer.)
- **'r+' – Read or Write Mode:** This mode is used when we want to write or read the data from the same file.
- **'a+' – Append or Read Mode:** This mode is used when we want to read data from the file or append the data into the same file.

Basic File Handling Operations - Open a file

file_object = open("filename", "mode")

<u>Mode to open a binary file:</u>

- **'wb' –** Open a file for write only mode in the binary format.
- **'rb' –** Open a file for the read-only mode in the binary format.
- **'ab' –** Open a file for appending only mode in the binary format.
- **'rb+' –** Open a file for read and write only mode in the binary format.
- **'ab+' –** Open a file for appending and read-only mode in the binary format.

Basic File Handling Operations - Open a file

- **open**
- **Example 1**

```
fob = open("C:/Python/test.txt", "r+")
```

- **Explanation**

Opening the file test.txt in a read or write.

Close

To close a file using the built-in close() function.

Basic File Handling Operations - Close a file

- **close**

- To close a file, we must first open the file.
- Important to close a file after open because the written data will not be saved into the file without closing it.
- Two ways in which the files can be closed
  1. **close()**
  2. **with open("test.txt", "r") as f:**
     *(Remarks: When the file is open using **with**, all the cleanup is automatically performed by Python. i.e. no close() is required.)*

Basic File Handling Operations - Close a file

- **close()**
  Close an opened file.
- **Example 2**

```
fob = open("C:/Python/test_write.txt", "w")
fob.write("Testing1!")
fob.close()
```

- **Explanation**

Closing the file after writing ‘Testing1!’ to the ‘test_write.txt’ file.
(Try to execute without closing the file)

Basic File Handling Operations - Close a file

- **close()**
  Close an opened file using with statement.

- **Example 3**

```
with open("C:/Python/test_write.txt", "w") as fob:
        fob.write("Testing2!")
```

- **Explanation**

Auto close the file after writing ‘Testing2!’ to the ‘test_write.txt’ file by using with statement during open.

Read

To open a file using the built-in read() / readline() / readlines() functions.

Basic File Handling Operations - Read a file

- **read**
- To read a file, we have to open the file in **read mode (i.e. ‘r’)**
- Three ways in which the files can be read
  1. **read([n])**
  2. **readline([n])**
  3. **readlines()**

  where **n** is the number of bytes to be read

Basic File Handling Operations - Read a file

- **read([n])**
  Read the given no. of bytes and return string. It may read less if EOF is hit.

- **Example 4**

```
fob = open("C:/Python/test.txt", "r")
print(fob.read(4))
fob.close()
```

- **Explanation**

Opening the file test.txt in a read-only mode and are reading only the first 4 characters of the file.

Basic File Handling Operations - Read a file

- **read([n])**
  Read the given no. of bytes and return string. It may read less if EOF is hit.

- **Example 5**

```
fob = open("C:/Python/test.txt", "r")
print(fob.read())
fob.close()
```

- **Explanation**

Opening the file test.txt in a read-only mode and are reading all the content in the file.

Basic File Handling Operations - Read a file

- **readline([n])**
  Read an entire line (trailing with a new line char) from the file and return string.

- **Example 6**

```
fob = open("C:/Python/test.txt", "r")
print(fob.readline(3))
fob.close()
```

- **Explanation**

Opening the file test.txt in a read-only mode and are reading first 3 characters of the entire line.

Basic File Handling Operations - Read a file

- **readline([n])**
  Reads an entire line (trailing with a new line char) from the file and return string.

- **Example 7**

```
fob = open("C:/Python/test.txt", "r")
print(fob.readline())
print(fob.readline())
fob.close()
```

- **Explanation**

Opening the file test.txt in a read-only mode and are reading first two lines.

Basic File Handling Operations - Read a file

- **readlines()**
  Read all the lines include the newline characters and return a list of lines read from the file.

- **Example 8**

```
fob = open("C:/Python/test.txt", "r")
print(fob.readlines())
fob.close()
```

- **Explanation**

Opening the file test.txt in a read-only mode and are reading all the lines from the file.

Basic File Handling Operations - Read a file

- **How to read all lines using for loop?**

- **Example 9**

```
fob = open("C:/Python/test.txt", "r")
for line in fob:
      print(line)
fob.close()
```

- **Explanation**

Opening the file test.txt in a read-only mode and are reading all the lines from the file.

Basic File Handling Operations - Read a file

- **How to read a specific line?**

- **Example 10**

```
line_number = 2
fob = open("C:/Python/test.txt", 'r')
currentline = 1
for line in fob:
        if(currentline == line_number):
                print(line)
                break
        currentline = currentline + 1
fob.close()
```

- **Explanation**

Opening the file test.txt in a read-only mode and are reading line number 2 from the file.

Write
(in write mode)

To write a file using the built-in write() / writelines() functions.

Basic File Handling Operations - Write a file in write mode

- **write**

- To write into a file in Python, we need to open it in **write mode (i.e. 'w')** or append mode (i.e. 'a')
- Two ways in which the files can be written *(i.e. write a string or a sequence of bytes(for binary files))*
  1. **write(string)**
  2. **writelines(sequence)**

Basic File Handling Operations - Write a file in write mode

- **write(string)**
  Write a string to the file and do not return any value.

- **Example 11**

```
fob = open("C:/Python/test_write.txt", "w")
fob.write("Enjoy Learning Python!")
fob.close()
```

- **Explanation**

Writing the string ‘Enjoy Learning Python!’ into the ‘test_write.txt’ file. *(Erase the existing data in the file)*

Basic File Handling Operations - Write a file in write mode

- **write(string)**
  Write a string to the file and do not return any value.

- **Example 12**

```
fob = open("C:/Python/test_write.txt", "w")
fob.write("Enjoy Learning Python!")
fob.write("Sure!")
fob.close()
```

- **Explanation**

Writing the string ‘Enjoy Learning Python!’ and ‘Sure!’ into the ‘test_write.txt’ file. *(Erase the existing data in the file)*

Basic File Handling Operations - Write a file in write mode

- **write(string)**
  Write a string to the file and do not return any value.

- **Example 13**

```
fob = open("C:/Python/test_write.txt", "w")
fob.write("Enjoy Learning Python!\n")
fob.write("Sure!")
fob.close()
```

- **Explanation**

Writing the string ‘Enjoy Learning Python!’ and ‘Sure!’ in separate line into the ‘test_write.txt’ file. *(Erase the existing data in the file)*

Basic File Handling Operations - Write a file in write mode

- **writelines(sequence)**
  Write a sequence of string (e.g. a list of string) to the file.

- **Example 14**

```
districts = ["Kwun Tong\n", "Sha Tin\n", "Wong Tai Sin"]
fob = open("C:/Python/test_write.txt", "w")
fob.writelines(districts)
fob.close()
```

- **Explanation**

Writing the list of data into the ‘test_write.txt’ file.

Write
(in append mode)

To append a file using the built-in write() / writelines() functions.

Basic File Handling Operations - Write a file in append mode

- **write**

- To write into a file in Python, we need to open it in write mode (i.e. ‘w’) or **append mode (i.e. ‘a’)**
- Two ways in which the files can be appended *(i.e. append a string or a sequence of bytes(for binary files))*
  1. **write(string)**
  2. **writelines(sequence)**

Basic File Handling Operations - Write a file in append mode

- **write(string)**
  Append a string to the file.

- **Example 15**

```
fob = open("C:/Python/test_append.txt", " a")
fob.write("I Love Python!")
fob.close()
```

- **Explanation**

Appending the string 'I Love Python!' to the 'test_append.txt' file.
(Append the string to the end of the existing data in the file)
You may execute the program a few times to see the result.

Basic File Handling Operations - Write a file in append mode

- **writelines(sequence)**
  Write a sequence of string (e.g. a list of string) to the file.
- **Example 16**

```
words = ["\nGood", "\nGreat", "\nNice"]
fob = open("C:/Python/test_append.txt", "a")
fob.writelines(words)
fob.close()
```

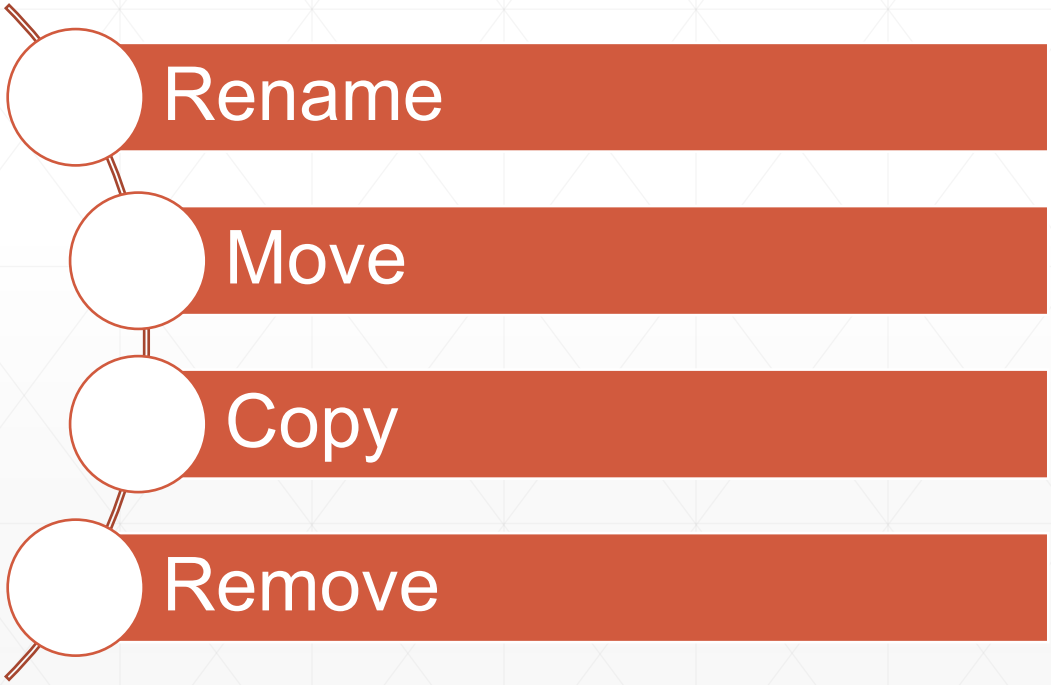
- **Explanation**

Appending the list of string to the 'test_append.txt' file. *(Append the string to the end of the existing data in the file)*

Other File Handling Operations

Other File Handling Operations

- Other File Handling Operations in Python



Rename

To rename a file using the built-in rename() function.

Other File Handling Operations - Rename a file

- **rename**
- The “os” module has the built-in methods using which we can perform the renaming on the file.
- **Syntax**

import os
os.rename(current_file_name, new_file_name)

Other File Handling Operations - Rename a file

- **os.rename(current_file_name, new_file_name)**
  Rename a filename to a new filename.
- **Example 17**

```
import os
os.rename("test.txt", "test1.txt")
```

- **Explanation**

Rename the file ‘test.txt’ to ‘test1.txt’.

Other File Handling Operations - Rename a file

- **os.rename(current_file_name, new_file_name)**
  Rename a filename to a new filename.
- **Example 18**

```
import os
os.rename("C:/Python/test1.txt", "C:/Python/test.txt")
```

- **Explanation**

Rename the file ‘test1.txt’ to ‘test.txt’ with the full path.

Move

To move a file using the built-in rename() / move() functions.

Other File Handling Operations - Move a file

- **move**
- The “os” and “shutil” module has the built-in methods using which we can perform the move a file.
- **Syntax 1**

import os
os.rename(current_file_name_with_source_path, new_file_name_with_destination_path)

- **Syntax 2**

import shutil
shutil.move(“current_file_name_with_source_path , new_file_name_with_destination_path”)

Other File Handling Operations - Move a file

- **os.rename(current_file_name_with_source_path, new_file_name_with_destination_path)**

- **Example 19**

```
import os
os.rename("C:/Python/test.txt", "C:/Python/test/test.txt")
```

- **Explanation**

Move the test.txt from source path to destination path.

Other File Handling Operations - Move a file

- **shutil.move(“current_file_name_with_source_path , new_file_name_with_destination_path”)**

- **Example 20**

```
import shutil
shutil.move("C:/Python/test/test.txt", "C:/Python/test.txt")
```

- **Explanation**

Move the ‘test.txt’ from source path to destination path.

Copyfile

To copy a file using the built-in copyfile() function.

Other File Handling Operations - Copy a file

- **copyfile**
- The “shutil” module has the built-in methods using which we can perform the copy a file.
- **Syntax**

import shutil
shutil.copyfile(current_file_name_with_source_path, new_file_name_with_destination_path)

Other File Handling Operations - Copy a file

- **shutil.copyfile(current_file_name_with_source_path, new_file_name_with_destination_path)**

- **Example 21**

```
import shutil
shutil.copyfile("C:/Python/test.txt", "C:/Python/test1.txt")
```

- **Explanation**

Copy the ‘test.txt’ from source path to ‘test1.txt’ at destination path.

Remove

To remove a file using the built-in remove() function.

Other File Handling Operations - Remove a file

- **remove**
- The “os” module has the built-in methods using which we can perform the remove (i.e. delete) a file.
- **Syntax**

import os
os.remove(file_name)

- **Example 22**

```
import os
os.remove("C:/Python/test1.txt")
```

- **Explanation**

Remove the ‘test1.txt’.

Other Usages

Other Usages

- How to …
  - get the filename?
  - get the file size?
  - use relative path?
  - file existence?
  - check word count in a file?

Thank you