Python – Files

- File handling is an important part for many applications including web application.
- Python has several functions for creating, reading, updating, and deleting files.

File Handling

- The key function for working with files in Python is the `open()` function.
- The `open()` function takes two parameters; *filename*, and *mode*.
- There are four different methods (modes) for opening a file:
  - `"r"` - Read - Default value. Opens a file for reading, error if the file does not exist.
  - `"a"` - Append - Opens a file for appending, creates the file if it does not exist.
  - `"w"` - Write - Opens a file for writing, creates the file if it does not exist.
  - `"x"` - Create - Creates the specified file, returns an error if the file exists.

In addition you can specify if the file should be handled as binary or text mode

`"t"` - Text - Default value. Text mode

`"b"` - Binary - Binary mode (e.g. images)

Syntax

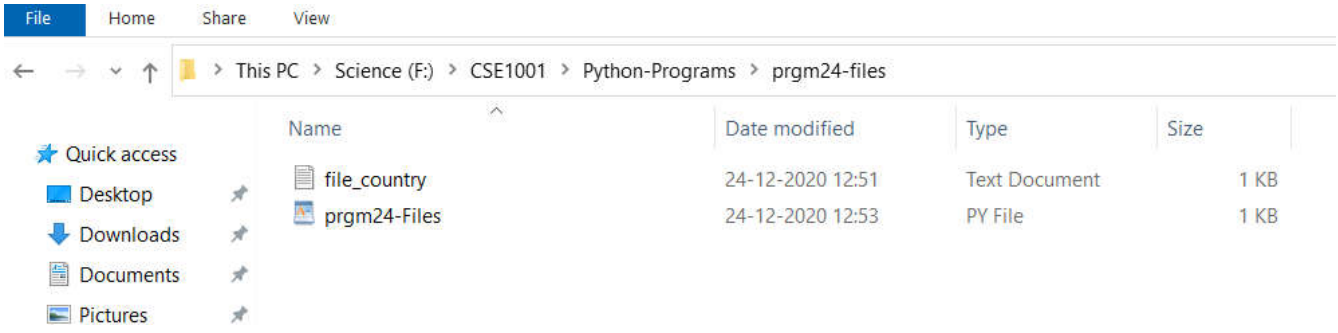
- To open a file for reading it is enough to specify the name of the file:
- Eg: `f = open("file_country.txt ")`

- The code above is the same as:
  `f = open("file_country.txt ", "rt")`

- Because `"r"` for read, and `"t"` for text are the default values, you do not need to specify them.

Python File Open

Note: We have created a file with name "`file_country.txt`" in the same folder where the Python file processing file is stored as shown below.



- To open the file, use the built-in `open()` function.

- The `open()` function returns a file object, which has a `read()` method for reading the content of the file:

Example

```
print("\n**************************************\n")
print("\n")
f = open("file_country.txt")
print(f.read())
```

```
India
America
Srilanka
China
Italy
```

Location of a File - If the file is located in a different location, you will have to specify the file path, like this:

Note: We have created a file with name "file_country.txt". Here both the "file_country.txt" and the Python file processing file are stored in different locations.

Example: Open a file on a different location.

```
print("\n**************************************************\n")
print("\n")
f = open("courses.txt")
print(f.read())
```

```
Traceback (most recent call last):
  File "prgm24-Files.py", line 10, in <module>
    f = open("courses.txt")
FileNotFoundError: [Errno 2] No such file or directory: 'courses.txt'
```

Example: Open a file on a different location.

```
print("\n**************************************************\n")
print("\n")
f = open("G:\\SP_Files\courses.txt")
print(f.read())
```

```
Physics
Artificial Intelligence
Mathemetics
Biology
Electronics Circuits
Engineering Graphics
```

Read Only Parts of the File

- By default the `read()` method returns the whole text.
- We can also specify how many characters we want to return.

Example - Return the first 3 characters of the file

```
print("\n**********************************\n")
print("\n")
f = open("file_country.txt")
print(f.read(3))
```

```
Ind
```

Example - Return few characters of the file

```
print("\n**********************************\n")
print("\n")
f = open("file_country.txt")
print(f.read(4))
print(f.read(3))
print(f.read(4))
f.close()
```

```
Indi
a
A
meri
```

Output

```
India
America
Srilanka
China
Italy
```

Reference: Total contents of the file "file_country.txt"

Read Lines

- We can return one line by using the `readline()` method:

Example: Read first line of the file

```
print("\n**********************************\n")
print("\n")
f = open("file_country.txt")
print(f.readline())
f.close()
```

```
India
```

Example: Read first two lines of the file

```
print("\n**********************************\n")
print("\n")
f = open("file_country.txt")
print(f.readline())
print(f.readline())
print(f.readline())
f.close()
```

```
India

America

Srilanka
```

Example: looping through the lines of the file

- By looping through the lines of the file, we can read the whole file, line by line:

Example: Loop through the file line by line

```
print("\n**********************************\n")
print("\n")
f = open("file_country.txt")
for x in f:
  print(x)
f.close()
```

```
India

America

Srilanka

China

Italy
```

Closing Files

- It is a good practice to always close the file when you are done with it.
- In some cases, due to buffering, changes made to a file may not show until you close the file.

Example - Close the file when you are finish with it:

```
print("\n**********************************\n")
print("\n")
f = open("file_country.txt")
for x in f:
  print(x)
f.close()
```

```
India

America

Srilanka

China

Italy
```

Example – Accessing the file contents after closing the file

```
print("\n**********************************\n")
print("\n")
f = open("file_country.txt")
print(f.readline())
print(f.readline())
f.close()
print(f.readline())
```

```
India

America

Traceback (most recent call last):
  File "prgm24-Files.py", line 72, in <module>
    print(f.readline())
ValueError: I/O operation on closed file.
```

Write to an existing file

- To write to an existing file, we must add a parameter to the `open()` function:
  - "w" - Write - will overwrite any existing content
  - "a" - Append - will append to the end of the file

‘Write Mode’

Example: Writing to a file using “w” mode

```
print("\n**********************************\n")
print("\n")
f = open("file_country.txt")
for x in f:
  print(x)
f.close()

f = open("file_country.txt", "w")
f.write("India is our country. *******")
f.close()

f = open("file_country.txt")
print(f.read())
f.close()
```

```
India is our country. *******
India is our country. *******
```

‘Append Mode’

Example: Writing to a file using “a” mode

```
print("\n**********************************\n")
print("\n")
f = open("file_numbers.txt")
for x in f:
  print(x)
f.close()

f = open("file_numbers.txt", "a")
f.write(" 6 7 8 9 10")
f.close()

f = open("file_numbers.txt")
print(f.read())
f.close()
```

After n^{th} execution

```
1 2 3 4 5
1 2 3 4 5 6 7 8 9 10
```

After $(n+1)^{th}$ execution

```
1 2 3 4 5 6 7 8 9 10
1 2 3 4 5 6 7 8 9 10 6 7 8 9 10
```

Activity:

1. Create a file (say numbers_file.txt). Write a python code to achieve the following.
    a. Read any 10 numbers from users and insert them into numbers_file.txt.
    b. Access the elements from **numbers_file.txt** and select the even numbers. Then write the even numbers into another one file called **even_numbers_file.txt**