Python File Open

File handling is an important part of any 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)

To open a file for reading it is enough to specify the name of the file:

In [4]:
```
f = open("demofile1.txt")

# The code above is the same as:

f = open("demofile1.txt", "rt")
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[4], line 1
----> 1 f = open("demofile1.txt")
      3 # The code above is the same as:
      5 f = open("demofile1.txt", "rt")

File ~/Developer/Pycharm/LearnPython/venv/lib/python3.9/site-packages/IPython/core/interactiveshell.py:286, in _modified_open(file, *args, **kwargs)
    279 if file in {0, 1, 2}:
    280     raise ValueError(
    281         f"IPython won't let you open fd={file} by default "
    282         "as it is likely to crash IPython. If you know what you are doing, "
    283         "you can use builtins' open."
    284     )
--> 286 return io_open(file, *args, **kwargs)

FileNotFoundError: [Errno 2] No such file or directory: 'demofile1.txt'
```

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

> **Note:** Make sure the file exists, or else you will get an error.

In [5]:
```
# Trying with a file that exists

f = open("demofile.txt")
```

Arsalan

In [ ]:
```

```

Open a file on the Server

Assume we have demofile.txt, located in the same folder as Python:

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:

In [1]:
```
f = open("demofile.txt", "r")
print(f.read())
```

```
Hello! Welcome to demofile.txt
This file is for testing purposes.
Good Luck!
```

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

In [2]:
```
f = open("/Users/arsalan/Developer/Pycharm/LearnPython/01. Python/welcome
print(f.read())
```

```
Hello World!
```

Read Only Parts of the File

By default the **read()** method returns the whole text, but you can also specify how many characters you want to return:

In [3]:
```
# Return the 5 first characters of the file:

f = open("demofile.txt", "r")
print(f.read(5))
```

```
Hello
```

Read Lines

You can return one line by using the **readline()** method:

In [4]:
```
# Read one line of the file:

f = open("demofile.txt", "r")
print(f.readline())
```

```
Hello! Welcome to demofile.txt
```

By calling **readline()** two times, you can read the two first lines:

In [5]:
```
# Read two lines of the file:

f = open("demofile.txt", "r")
print(f.readline())
print(f.readline())
```

```
Hello! Welcome to demofile.txt

This file is for testing purposes.
```

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

In [6]:
```
# Loop through the file line by line:

f = open("demofile.txt", "r")
for x in f:
  print(x)
```

```
Hello! Welcome to demofile.txt

This file is for testing purposes.

Good Luck!
```

Close Files

It is a good practice to always close the file when you are done with it.

In [7]:
```
# Close the file when you are finish with it:

f = open("demofile.txt", "r")
print(f.readline())
f.close()
```

```
Hello! Welcome to demofile.txt
```

> **Note:** You should always close your files, in some cases, due to buffering, changes made to a file may not show until you close the file.

Arsalan

In [ ]:
```

```

Python File Write

Write to an Existing File

To write to an existing file, you must add a parameter to the **open()** function:

"a" - Append - will append to the end of the file **"w"** - Write - will overwrite any existing content

In [1]:
```
# Open the file "demofile2.txt" and append content to the file:

f = open("demofile2.txt", "a")
f.write("Now the file has more content!")
f.close()

#open and read the file after the appending:
f = open("demofile2.txt", "r")
print(f.read())
f.close()
```

```
Now the file has more content!
```

In [2]:
```
# Open the file "demofile3.txt" and overwrite the content:

f = open("demofile3.txt", "w")
f.write("Woops! I have deleted the content!")
f.close()

#open and read the file after the appending:
f = open("demofile3.txt", "r")
print(f.read())
f.close()
```

```
Woops! I have deleted the content!
```

> **Note:** the "w" method will overwrite the entire file.

Create a New File

To create a new file in Python, use the **open()** method, with one of the following parameters:

"x" - Create - will create a file, returns an error if the file exist **"a"** - Append - will create a file if the specified file does not exist **"w"** - Write - will create a file if the specified file does not exist

In [3]:
```
# Create a file called "myfile.txt":

f = open("myfile.txt", "x")
```

> **Note:** A new empty file is created!

In [4]:
```
# Create a new file if it does not exist:

f = open("myfile2.txt", "w")
```

Arsalan

In [ ]:

Python Delete File

Delete a File

To delete a file, you must import the OS module, and run its **os.remove()** function:

In [4]:
```
import os

# Open the file in write mode and write to it
f = open("Sample.txt", "w")
f.write("Hello World")
f.close()

# Reopen the file in read mode and read the content
f = open("Sample.txt", "r")
print(f.readline())
f.close()

# Remove the file after reading
os.remove("Sample.txt")
```

```
Hello World
```

Check if File exist

To avoid getting an error, you might want to check if the file exists before you try to delete it:

In [5]:
```
import os

# Open the file in write mode and write to it
f = open("Sample.txt", "w")
f.write("Hello World")
f.close()

# Reopen the file in read mode and read the content
f = open("Sample.txt", "r")
print(f.readline())
f.close()

# Remove the file after reading
if os.path.exists("Sample.txt"):
    os.remove("Sample.txt")
else:
    print("The file does not exist")
```

```
Hello World
```

Delete Folder

To delete an entire folder, use the **os.rmdir()** method:

In [6]:

```python
import os

# Create a folder
os.mkdir("myfolder")
print("Folder created")

# Remove the folder
os.rmdir("myfolder")
print("Folder deleted")
```

```
Folder created
Folder deleted
```

> **Note:** You can only remove empty folders.

Arsalan

In [ ]: