FILE HANDLING

FILE OPEN

- Python has several functions for creating, reading, updating, and deleting files.
- 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:

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

The code above is the same as:

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

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

NOTE : By default, the file will be located in the same folder as Python:

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

Example

```
f = open("D:\VNSGU\MCA 4\MCA 4 2020-21\demofile1.txt")
```

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

Example

```
print(f.read())
```

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:

Example

Return the 5 first characters of the file:

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

READ LINES

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

Example

Read one line of the file:

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

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

Example

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

CLOSE FILES

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

Example

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

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

Example

```
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())
```

Example

```
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())
```

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

Example

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

DELETE FILE

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

Example

```
import os
os.remove("demofile.txt")
```

CHECK IF FILE EXIST

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

Example

```
import os
if os.path.exists("demofile.txt"):
  os.remove("demofile.txt")
else:
  print("The file does not exist")
```

FILE HANDLING WITH EXCEPTION HANDLING

```
try:
    fh = open("testfile", "w")
    fh.write("This is my test file for exception
handling!!")

except IOError:
    print "Error: can\'t find file or read data"

else:
    print "Written content in the file successfully"
    fh.close()
```

```
try:
   fh = open("testfile", "r")
   fh.write("This is my test file for exception handling!!")

except IOError:
   print "Error: can\'t find file or read data"

else:
   print "Written content in the file successfully"
```

DELETE FOLDER

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

Example

```
import os
os.rmdir("myfolder")
```