Day – 7 (part)

**Python File I/O**

**And Modules**

**7.4 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 conten
* ***"r"*** used to read from file

*Example* 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())

*Example* 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())

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

**7.5 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 Create a file called "myfile.txt":

f = open("myfile.txt", "x")

Result: a new empty file is created!

Example Create a new file if it does not exist:

f = open("myfile.txt", "w")

**7.6 Python Modules**

* Module: Consider a module to be the same as a ***code library***. A file containing a ***set of functions*** you want to ***include*** in your ***application***.
* Create a Module: To create a module just save the code you want in a file with the file extension ***.py***:

*Example*: Save this code in a file named mymodule.py

def greeting(name):  
  print("Hello, " + name)

* Use a Module: Now we can use the module we just created, by using the ***import*** statement:

*Example* : Import the module named mymodule, and call the greeting function:

import mymodule  
  
mymodule.greeting("Jonathan")

* Note: When using a function from a module, use the syntax: **module\_name.function\_name**. i.e. use "**.**" dot operator.
* Variables in Module: The module can contain ***functions***, as already described, but also ***variables*** of all ***types*** (arrays, dictionaries, objects etc):

Example Save this code in the file mymodule.py

person1 = {  
  "name": "John",  
  "age": 36,  
  "country": "Norway"  
}

Example Import the module named mymodule, and access the ***person1*** dictionary:

import mymodule  
  
a = mymodule.person1["age"]  
print(a)

* Naming a Module: You can name the module file whatever you like, but it must have the file extension ***.py***
* Re-naming a Module: You can create an alias when you *import* a module, by using the ***as*** keyword:

Example Create an alias for mymodule called mx:

import mymodule as mx  
  
a = mx.person1["age"]  
print(a)

* Built-in Modules: There are several built-in modules in Python, which you can import whenever you like.
* *list all the function names Using the* ***dir()*** *Function:* There is a built-in function to list all the function names (or variable names) in a module. The ***dir()*** function:

Example List all the defined names belonging to the platform module:

import platform  
  
x = dir(platform)  
print(x)

* Note: The ***dir()*** function can be used on *all modules*, also the ones you *create* *yourself*.

**7.7 IMPORT From Module**

You can choose to import only parts from a module, by using the **from** keyword.

Example The module named **mymodule** has one function and one dictionary:

def greeting(name):  
  print("Hello, " + name)  
  
person1 = {  
  "name": "John",  
  "age": 36,  
  "country": "Norway"  
}

Example Import only the person1 dictionary from the module (no "**.**" is required to access *module-variables*):

from mymodule import person1  
  
print (person1["age"])