Python Course

Class 3 : Structures and files

Presented by : Sofiane MEDJKOUNE
Email : sofiane.medjkoune@protonmail.com

New structure, the dictionary

- Python dictionary is an unordered collection of items. While other compound data types have only value as an element, a dictionary has a **key: value** pair.
- If you need to pair an information stored with its value, the dictionary is your structure. Take last course's exercice about BMI, we could've used the dictionary to store the name of the person as a **key** and his personal informations as a **value.**

- In python, to create a dictionary you can use the following syntax :
  - my_variable_id = dict()
  - my_variable_id = {} #Just like with lists right ?
  - You can also initialize your dictionary with values like we've seen with the lists, remember the syntax **key:value** . ages_dict = **{'sofiane': 23, 'Lyes': 12}**
  - Or ages_dict = **dict({'sofiane': 23, 'Lyes': 12})**

Dictionary modification

- To add values to your dictionary, simple syntaxe : ***my_dict[key] = value***
- Your key can be of any type (String, integer,…)
- You can update a value the following way :

```
my_dict[‘Lyes’] = 12 #Oops i feel like i exaggerated
my_dict[‘Lyes’] = 18 #Now my key: value pair has been updated
```

- To access a the **value** of a **key,** all we have to do is use the variable : ***my_dict[key]***, for example :

```
name = “Lyes”
print(“the age of “,name,” is : “,my_dict[name])
```

- To the get all the **keys** as a **list**, it’s easy, just use : your_dict.keys()
- To the get all the **values** as a **list**, well you guessed it : your_dict.values()

Parse through your dict !

- You’ll obviously need to parse through all of your dictionary, here’s what to do :

```
for key in my_dict :
    print(my_dict[key])
```

- You may need the **keys** and **values** directly, here’s the solution for you :

```
for key, value in my_dict.items():
    print(key)
    print(value)
```

- There’s many other ways to use your dictionaries depending on your needs, don’t forget the check the official documentation for more useful methods : https://docs.python.org/3/tutorial/datastructures.html#dictionaries

Files, what and why ?

- Until now we didn’t save any information. Basically if you stop your program or turn off your computer while your program is still running, you won’t save any information.
- Files are a way to store your data within your computer and retrieve them anytime you want.
- Files on your computer have a name and extension for example : omc_members.csv (CSV for Comma-separated values, this is a format that’s we’ll be using a lot during this course) You can find others extensions here : https://www.computerhope.com/issues/ch001789.htm

Open a file

- In your computer, phone or tablet, if you want to read a book for example you need to ***open*** that file, and sames goes for programming in general, if you wanna read a file or write in it you need to ***open*** it, don't forget to ***close it*** when you finish you work.
- To open and close a file all you have to is :

```
omc_members = open(path, opening_mode)
omc_members.close()
```

  - The path to your file can either be an ***absolute*** or ***relative*** path. https://www.coffeecup.com/help/articles/absolute-vs-relative-pathslinks/
  - The opening modes are ***'r'*** to read, ***'b'*** to write (be careful this mode erases the content of the file) and ***'a'***, to add content at the end of the file.

Read your file and write on it

When you read your file you can either read : omc_file = open(‘omc.csv’, ‘r’)

- The whole content at onces

  omc_members = omc_file***.read()***

- Line by line :

  omc_members = omc_file.***readLines()*** # a list of all the lines of the file

  ```
  for member in omc_members :
      print(member)
  ```

To write something in your file, it’s also simple : omc_file = open(omc.csv, ‘a’) ***#If i opened with ‘w’ it would’ve erased the whole content***

omc_file.***write***(‘yes i’m putting a new line’)

Don’t forget to close your file : omc_file.close()

Additional informations

- When you open your file, an error may occur, this is why we can use the *with* statement :

```
with open(path, open_mode) as my_file:
    # my bloc of instructions
```

- The write method we've seen lets us write strings only, two solutions :
  - Convert our variables to strings.
  - use the pickle module https://docs.python.org/3.1/library/pickle.html

Split and join

- When we’re manipulating strings (in files for example) we may need the informations separately (in CSV format for example, informations are separated by a ‘,’), to do so, we have the ***split*** function. This functions **splits a string and returns a list.**

```
list_of_words = my_string.split(separator)
 #This separator can be anything it could be a blank in a casual text file or a coma in your csv files.
```

- If you need to add a person in the omc.csv file, you have to use the ***join*** method, that works in the opposite way of split. This function **joins the content of a list to make a string.**

```
personnal_informations = [‘sofiane’,23,’sofiane.medjkoune@protonmail.com’]
‘,’.join(personnal_informations)
```

Practice !

Let’s work with the previous BMI exercise, only this time we’ll write the personal informations in a csv file, read it, for each person calculate the BMI and add it to our csv file