Python Basics

General Info

- In case of any doubts or help, you can contact me at rafael.fontana@polito.it

- **DO NOT BE AFRAID TO USE GOOGLE!** It knows everything!

- **Strongly recommend** to use a LINUX based OS, instead of Windows

How to launch a Python script

Launching a Python script is quite easy. These are the steps to follow:

1. Open a new terminal (command line for Windows)
2. Navigate to the folder, `cd name of the folder` to open a folder, *dir* to list the files in the folder (`ls` for MacOS and Linux)
3. Type `python name_of_the_file.py` (for example `python myScript.py`)

For the rest of the course, we will always launch our Python scripts in this way

How to write a simple class

```python
#This is the class
class Student:
        def __init__(self,name,surname):
                self.name=name
                self.surname=surname

        def show(self)
                print(f"Hi, i'm {self.name} {self.surname}")

#This is the main of our program
if __name__=="__main__":
        name="Lionel"
        surname="Messi"
        studentA= Student(name,surname)
        studentA.show()
```

Things to notice

- Every class needs to have an '__init__' method
- The attribute of the class start with 'self.'
- Each method of the class needs to have 'self' as first parameter
- We can easily print using f-string which will automatically be filled with the variables values

Ask user Input

The function to ask inputs from user in Python is simply 'input()'. So for example this code:

```python
number=int(input('Please write a number:'))
print(f"{number+10}")
```

will ask to the user to write a number in the terminal and will print it back, after adding 10 to it.

```
Terminal                                        ● ● ●

Please write a number: 5
15
```

Exercise 1

Develop a `Student` class similar to the one in the previous example. The class should be able to receive as input the name, the surname and also the year of birth of the student. In addition, it must have a method to return the current age of the student. The input should be provided by the user (NOT hardcoded in the script).

```python
if __name__=="__main__":
        name=...
        surname=...
        birthYear=...
        studentA= Student(name,surname,birthYear)
        studentA.show()
        studentA.age()
        studentA.save()
```

How to work on files

Reading and writing files in Python is really easy. The functions you need are the following:
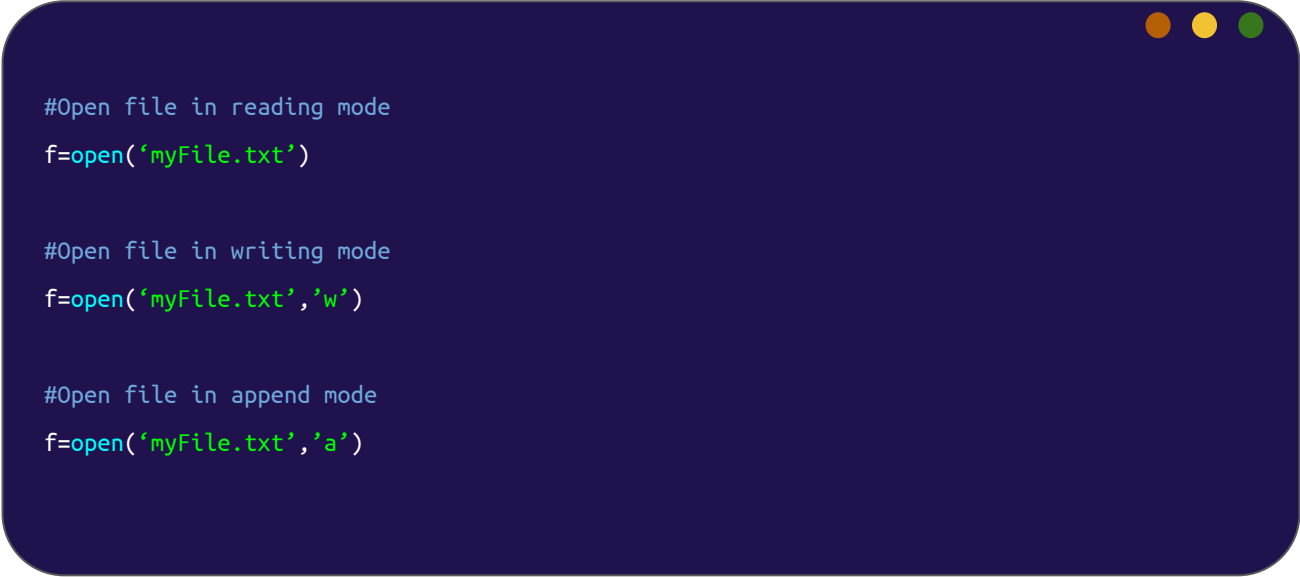
```
open(...)
read(...)
write(...)
close(...)
```

Open a file

open()

This function returns an object of the type file

```
#Open file in reading mode
f=open('myFile.txt')


#Open file in writing mode
f=open('myFile.txt','w')


#Open file in append mode
f=open('myFile.txt','a')
```
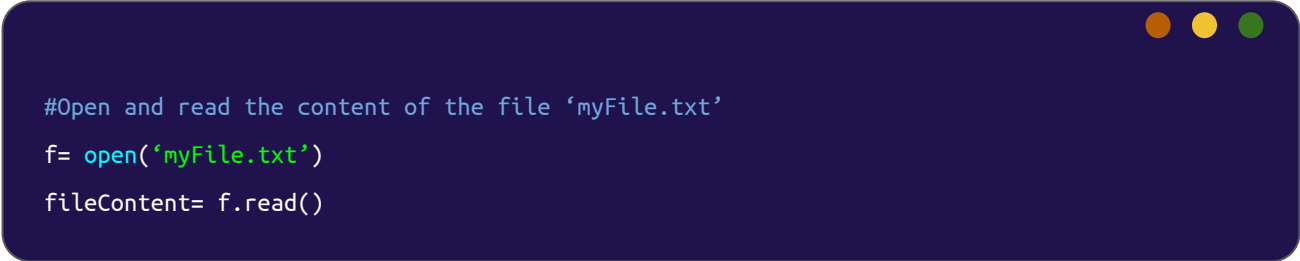
Read a file

read()

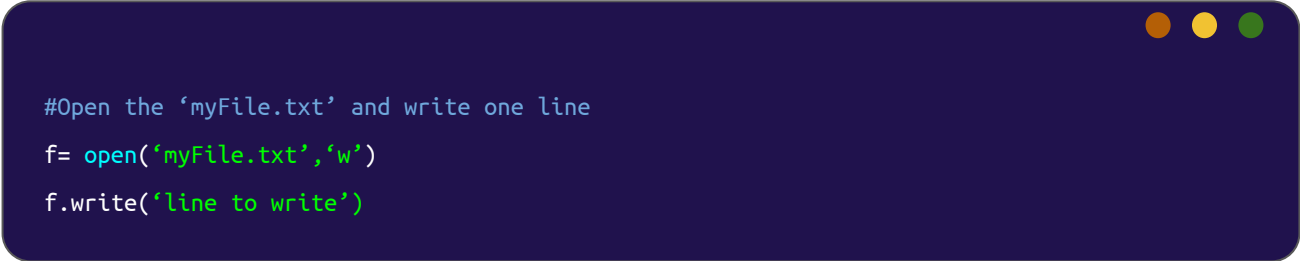This function returns the whole content of a file as a string

```python
#Open and read the content of the file 'myFile.txt'
f= open('myFile.txt')
fileContent= f.read()
```

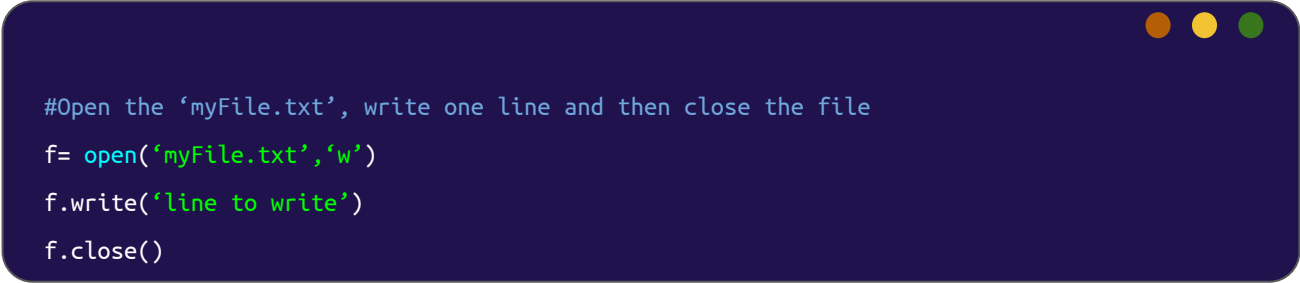# Write a file

write()

This function writes on a file

```
#Open the 'myFile.txt' and write one line
f= open('myFile.txt','w')
f.write('line to write')
```

Close a file

close()

This function closes the file

```
#Open the 'myFile.txt', write one line and then close the file
f= open('myFile.txt','w')
f.write('line to write')
f.close()
```

Exercise 2

Modify the class you wrote for the previous exercise by adding a method that writes the *Student* information on a *.txt* file

```python
if __name__=="__main__":
        name=...
        surname=...
        birthYear=...
        studentA= Student(name,surname,birthYear)
        studentA.show()
        studentA.age()
        studentA.save()
```

student.txt

Lionel,Messi,1987

If-elif-else

The easiest way to show how *if-else-if* works is by making an example. Let's see a script that asks to the user two numbers and tells if the first is greater, smaller, or equal to the second

```python
if __name__=="__main__":
        #Receive two number as input
        a=int(input('Write a number\n'))
        b=int(input('Write another number\n'))
        #First case a>b
        if a>b:
                print(f"{a} is greater than {b}")
        #Second case a<b
        elif a<b:
                print(f"{a} is smaller than {b}")
        #All the other case
        else:
                print(f"{a} equals {b}")
```

Exercise 3

Modify the class you wrote for the previous exercise by adding two boolean attributes (i.e. true/false) named *bachelor* and *master*, which should be provided by the user as input. Add also two methods to assess whether the student is enrolled on a bachelor or a master degree.

```python
class  Student:
    ...
    def isBachelor(self):
    ...
    def isMaster(self):
    ...


if __name__=="__main__":
    ...
    studentA.isBachelor()
    studentA.isMaster()
```

List, for-loop, while-loop

As you've seen in the slides, in Python the `array` type is called `list` and it has some particular functions related to it:

- *list[i]*
  Return the i-th element of the list

- *list.append(x)*
  Add an item to the end of the list

- *len(list)*
  Return the number of element in the list

- *list.remove(x)*
  Remove the first item from the list whose value is *x*. An error will occur in case there is no such item

- *list.pop(index)*
  Remove the item at the given position in the list and returns it. If no index is specified, *a.pop()* removes and returns the last item in the list

List, for-loop, while-loop

Additional functions

- *list.index(x)*
  Return the index in the list of the first item whose value is x. It is an error if there is no such item.

- *list.count(x)*
  Return the number of times x appears in the list.

- *list.insert(i,x)*
  Insert an item at a given position. The first argument is the index of the element before which to insert

⚠️ **Remember that lists in Python start at 0 !!** ⚠️

How to use lists?

The easiest way to show how the *for-loop* and the *while-loop* works is by making an example. So let's see a script that creates a list with some number and calculates their sum and their product:

For-loop

```python
if __name__=="__main__":
    numbers=[1,2,3,4,5]
    list_len=len(numbers)
    ##For-loop for the sum
    #set sum to 0
    sum_of_numbers=0
    for i in range(list_len):
        #takes the previous value of the sum and add the number at the place i
        sum_of_numbers=sum_of_numbers+numbers[i]
        #the more "pythonic" way to do it is
        #sum_of_numbers+=numbers[i]
    print(f"The sum of the numbers in the list is {sum_of_numbers}")
```

While-loop

```python
if __name__=="__main__":
    numbers=[1,2,3,4,5]
    list_len=len(numbers)
    #set counter to 0 an product to 1
    i=0
    product_of_numbers=1
    while i<list_len:
        #Multiply the old value for the number at place i
        product_of_numbers=product_of_numbers*numbers[i]
        #the more "pythonic" way to do it is
        #product_of_numbers*=numbers[i]
        #Update counter
        i=i+1
    print(f"The product of the numbers in the list in {product_of_numbers}")
```

For-loop in the "pythonic" way

There is another method to write the for-loop that is: `for item in list-name:`

Using this method, the previous example becomes:

```python
if __name__=="__main__":
    numbers=[1,2,3,4,5]
    sum_of_numbers=0
    for item in numbers:
        sum_of_numbers+= item
```

By doing this we make the code more readable. However, we lost the information regarding the index of the item in the list. In case the information about the index is necessary, we have to use the "old way" or `enumerate()` (more details [here](#))

Exercise 4

Modify the class you wrote for previous exercise by adding a method to read a file that contains the vote of a student. If the file is written as follows:

```
studentVotes.txt
24,27,18,30,21
```

you will be able to obtain the separated list of vote by splitting the content in this way:

```python
fileContent= open('studentVotes.txt').read()
votesList=fileContent.split(',')
```

However, in this way `votesList` will be a list of string. How can we convert it into a list of numbers (int or float) ?

After you find a way to to that, add a method to the class to find the average, the maximum and the minimum of the votes

Dictionaries

A really useful Python datatype are *Dictionaries* . Dictionaries are collections of couples of key/value. *Key* are of type `string` and they are unique for each dict. *Value* can be of whatever type: `int`, `float`, `string`, `list` or even other `dict`. Let's see an example of `dict` and what function can be used on a dict.

```python
config={
        "lastUpdate": "2023-10-18-17:42",
        "devicesList": [
                {
                        "deviceID": "BOX1_A_FAN_1",
                        "availableResources": "Fan",
                        "IP": "192.168.1.254",
                        "port": 49160,
                        "endPoints": {
                                "REST": "http://192.168.1.254:49160/greenhouse/box_1/actuators/fan",
                                "MQTT": "greenhouse/box_1/actuators/fan"
                        },
                        "insertTimestamp": 1595709994.0579066
                }
        ]
}
```

Exercise 5

Modify the class you developed previously: add a method to create a dictionary with the attributes of the student and to print them

```python
class  Student:
        ...
        def asDictionary(self):
        ...


if __name__=="__main__":
        ...
        studentA.asDictionary()
```

Terminal

{"name":"Lionel","surname":"Messi","birthYear":1987,"votes":[24,27,18,30,21]}

JSON files

⚠️ **Follow this slide only after the lesson on the data format** ⚠️

For all the course we will use `.json` files. A JSON is essentially a dictionary saved on a file, it is easy for humans to read and write. It is easy for machines to parse and generate. Python has a module (i.e. library) called `json` that contains all the functions we need to read and write `.json` files.

A JSON file looks exactly like a dictionary:

```
student.json

{
        "name": "Lionel",
        "surname": "Messi",
        "birthYear": 1987,
        "votes": [24,27,18,30,21]
}
```

JSON module functions 1/2

All the functions we will use are contained in the Python json module (in case of using it, you need to import it!)

- *json.load(fp)*
  The output is a `dictionary` filled with the content from the file pointer fp (fp is the result of the function *open("<name_of_the_file>.json"))*. What we will usually write is:

  ```
  dictionaryName=json.load(open("<name of_the_file>.json"))
  ```

- *json.dump(d,fp)*
  It is used to write the dictionary d on the file pointer fp. What we usually write could be:

  ```
  json.dump(dictionaryName,open("<name of_the_file>.json","w"))
  ```

JSON module functions 2/2

- *json.loads(myString)*
  Return a dictionary obtained by converting the string `myString`. We will use this function later in the course

- *json.dumps(d)*
  Return a string by converting the dictionary `d`. This function will be used later in the course too

Exercise 6

Modify the code of exercise 4 and 5 to be able to work with a json file similar to the one below

```
student.json

{
        "name": "Lionel",
        "surname": "Messi",
        "birthYear": 1987,
        "votes": [24,27,18,30,21]
}
```