LAB1_Tips

SURVIVAL TIPS

Run a Python program

Once you have set up your environment, let's run your first "Hello world" program together.
Open Pycharm and in the editor section you are ready to type your first Python program.

Type:

```
print('This is my first program. \n Hello Wolrd!!')
```

Save the file (e.g. First_program.py) in the folder you prefer and **set the correct working directory**. This step is really important since it allows the Ipython console to work in the correct directory. In the end press onto **Run button** and your program will be executed.

Apart from the Python IDE, another way to execute your script is to run it from the terminal. Open your terminal (for MacOS and Linux users) or the Ubuntu 18.04 application (for Windows users).

- Activate Bioinfo_labs environment

  `conda activate Bioinfo_labs` (for some versions use 'source activate Bioinfo_labs')
- Move to the directory in which First_program.py has been saved.

  e.g.

  ```
  cd /home/marta/BIOINFORMATICS/LAB_1/
  ```

  (If you are using the Ubuntu 18.04 application, usually your Documents folder is placed in something like /mnt/c/Users/marta/Documents)
- Call the program

  ```
  python First_program.py
  ```

Pass arguments from command line

The Python **sys** module provides access to any command-line arguments via the `sys.argv`.
`sys.argv[0]` contains the name of the script. Try:
In pippo.py file write

```
import sys
print ('Number of arguments: %s arguments.' % len(sys.argv))
print ('Argv0: %s' % sys.argv[0])
print ('Argv1: %s' % sys.argv[1])
```

Now, let's execute it!
python pippo.py PRIMOARGOMENTO SECONDOARGOMENTO

Insert comments in a python script

In python the comments are preceded by the symbol #. **Always remember not to be stingy with comments!** If you read again your code in a month, you might not remember what some lines perform anymore. So, always be kind, add clever comments!!
E.g.

```
# This is my first comment, python interpreter will ignore this line
```

Random library

This module implements pseudo-random number generators for various distributions. Please, take a look to random documentation here https://docs.python.org/3/library/random.html especially random.choice() and random.choices()

```
import random
# random number between 0 and 20
a = random.randint(0, 21)
print (a)
```

File Handling

How to create a file

```
f= open("pippo.txt","w+")
for i in range(10):
      f.write("This is line %d\r\n" % (i+1))
f.close()
```

We declared the variable f to open a file named textfile.txt. Open takes 2 arguments, the file that we want to open and a string that represents the kinds of permission or operation we want to do on the file (here there is "w" letter in our argument, which indicates write and the plus sign that means it will create a file if it does not exist in library). The available option beside "w" are "r" for read and "a" for append and plus sign means if it is not there then create it.
Then we have a for loop that runs over a range of 10 numbers and uses the **write** function to enter data into the file. f.close() will close the instance of the file pippo.txt stored.

You can also append a new text to the already existing file or the new file. Eg.

```
f=open("pippo.txt", "a+")
for i in range(2):
      f.write("Appended line %d\r\n" % (i+1))
f.close()
```

Once again if you could see a plus sign in the code, it indicates that it will create a new file if it does not exist. But in our case we already have the file, so we are not required to create a new file.

How to Read a File

Not only you can create .txt file from Python but you can also call .txt file in a "read mode"(r).

```
f=open("pippo.txt", "r")
if f.mode == 'r':
      contents = f.read()
f.close()
```

We use the mode function in the code to check that the file is in open mode. If yes, use f.read to read file data and store it in variable content.
You can also work with file objects using the with statement to open a file. One bonus of using this method is that any files opened will be closed automatically after you are done. This leaves less to worry about during cleanup.
e.g.

```
with open(“testfile.txt”) as file:
      data = file.read() # do something with data
```

You'll also notice that in the above example we didn't use the "**file.close()**" method because the with statement will automatically call that for us upon execution.

How to Read a File line by line

You can also read your .txt file line by line if your data is too big to read. **Readlines** method reads all the lines of a file and put them in a list (with '\n' character at the end of each element of the list).

```
f=open("pippo.txt", "r")
f1 = f.readlines() # f1 is a list!!
for i in f1:
      print(i)
```

Another useful method is **readline**. It reads a single line of the file each time the method is called on the file. With this method you can avoid storing all the lines of the file in a list and directly calculate what you need one line at a time.

e.g.

```
f = open(filename)
line = f.readline()
while line:
      print (line) # or do something with the line
      line = f.readline() # this read the next line
 f.close()
```

Useful methods for strings

The string type has many useful methods. Please have a look at the documentation (here https://docs.python.org/2/library/string.html) to get a general idea of the methods available.

E.g.

```
mystring = 'This_is_my_string'
mystring.split('_')
mystring.count('is')
mystring.replace('_', ' ')
```