

Uso di GitHub

- Si consiglia di iniziare ad usare GitHub per scaricare e rimanere aggiornati sul materiale usato a lezione.
- Si può usare l'applicativo GitHub Desktop (<https://desktop.github.com/>) anche senza crearsi un account su GitHub.
- Se però vi create un account riceverete delle notifiche ogni volta che modifichiamo qualche file.

In aula B2 o C3, da shell, una volta il comando:

```
$ git clone git@github.com:mathcoding/programming.git
```

Viene creata una cartella “programming” con tutto il materiale. Le volte successive, basterà andare nella cartella e scrivere

```
$ cd programming
```

```
$ git pull
```

Uso personale di GitHub

- Se vi create un account, potete iniziarlo ad usare come se fosse un “Dropbox” pensato apposta per chi programma
- Potete creare un Vostro progetto con tutti gli esercizi che provate a svolgere
- Se lavorate da casa e poi venite in una delle aule, vi ritrovate sempre l’ultima versione degli esercizi che avete svolto (sono sincronizzati tramite Git)
- Per caricare i vostri file sul GitHub dovete usare i comandi:

```
$ git add soluzione_es_3.py
```

```
$ git commit -m "commento vostro" ./soluzione_es_3.py
```

```
$ git push
```

Oppure potete usare l’applicativo GitHub Desktop

Obiettivo: Prima vostra libreria

L'obiettivo degli esercizi dal 3.3 al 3.9 è di iniziare a scrivere una vostra libreria che permette di gestire delle **sequenze di dati**.

Chiameremo la libreria: **pairslis**

Ad un certo punto potrete importare la vostra libreria:

```
[1]: import pairlists
```

Oppure:

```
[2]: from pairlists import *
```

Obiettivo: Prima vostra libreria

Consiglio: lavorate direttamente con **Spyder3**

Tutte le funzioni che implementerete, dovranno essere salvate in un file chiamato **`"pairslist.py"`**

Per poter importare la libreria dovete avere la shell di python che si trova nella directory in cui avete salvato il file **`"pairslist.py"`**

Usare i comandi:

pwd : mostra la dir corrente vista dalla shell di python

ls : mostra la lista di file nella dir corrente

cd : cambia la dir corrente, andando in quella indicata (..) per salire di uno)

Obiettivo: Prima vostra libreria

```
IPython console
Console 1/A
Python 3.5.3 [Anaconda custom (64-bit)] (default, Feb 22 2017, 21:28:42) [MSC v.1900 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 5.1.0 -- An enhanced Interactive Python.
?          -> Introduction and overview of IPython's features.
%quickref  -> Quick reference.
help       -> Python's own help system.
object?    -> Details about 'object', use 'object??' for extra details.

Restarting kernel...

In [1]: pwd
Out[1]: 'C:\\Users\\gualandi'

In [2]: cd e:/GitHub/programming/scripts/
e:\\GitHub\\programming\\scripts

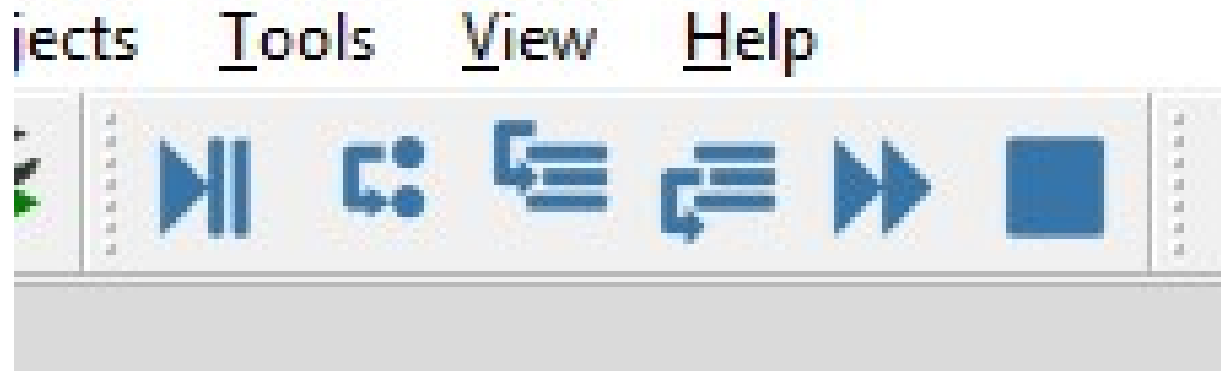
In [3]: ls
Volume in drive E is DATA
Volume Serial Number is D299-FD79

Directory of e:\\GitHub\\programming\\scripts

27/10/2017  08:57    <DIR>          .
27/10/2017  08:57    <DIR>          ..
27/10/2017  08:57    <DIR>          __pycache__
27/10/2017  08:59                666 pairslist.py
17/10/2017  19:10                2,242 Soluzioni_es_1.py
25/10/2017  11:18                4,841 Soluzioni_es_2.py
              3 File(s)              7,749 bytes
              3 Dir(s)  280,613,253,120 bytes free

In [4]: |
```

Uso del debugger in Spyder3



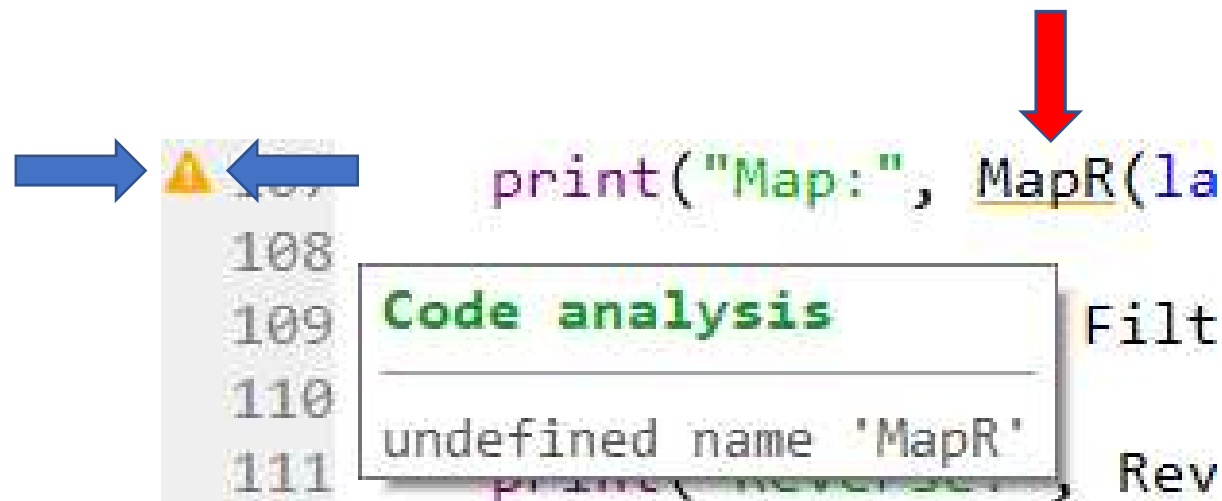
1. La prima icona fa partire il debugger
2. La seconda icona chiede di “eseguire” la riga corrente, evidenziata sulla sinistra nella finestra text editor
3. La terza icona chiede di mostrare il codice interno alla funzione che sta per essere chiamata
4. La quarta per uscire dalla funzione corrente
5. La quinta per eseguire tutta la parte mancante di codice
6. La sesta, il quadrato, per fermare l’esecuzione del debugger

NOTA: Ricordarsi sempre che “print()” è il debugger fondamentale!

Abituarsi ai “messaggi” di Spyder



```
25 def Tail(As)
26     """ Restituisce
27     return As[1:]
```



```
107 print('Map:', MapR(la... Filt... Rev...
108
109
110
111
```

Code analysis

undefined name 'MapR'