

[FIM] FONDAMENTI DI INFORMATICA per medicina e chirurgia high tech

P00: Python Intro

Dott. Giorgio De Magistris

demagistris@diag.uniroma1.it

Prof. Christian Napoli

c.napoli@uniroma1.it

CORSO DI LAUREA IN MEDICINA E CHIRURGIA HIGH TECH



SAPIENZA
UNIVERSITÀ DI ROMA

I3S

FACOLTÀ DI INGEGNERIA DELL'INFORMAZIONE, INFORMATICA E STATISTICA

DIAG

DIPARTIMENTO DI INGEGNERIA INFORMATICA, AUTOMATICA E GESTIONALE

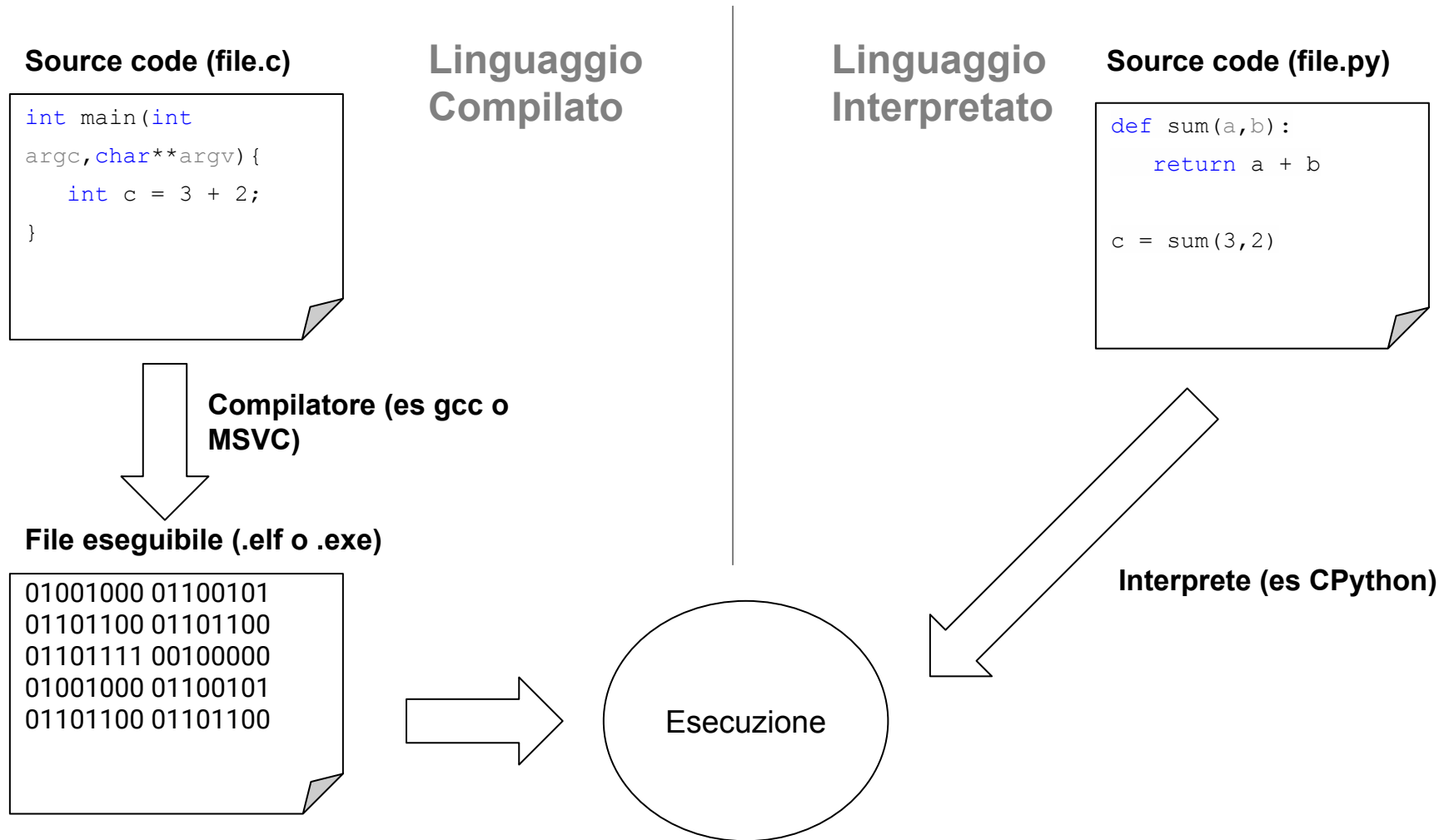
TUTTI I DIRITTI RELATIVI AL PRESENTE MATERIALE DIDATTICO ED AL SUO CONTENUTO SONO RISERVATI A SAPIENZA E AI SUOI AUTORI (O DOCENTI CHE LO HANNO PRODOTTO). È CONSENTITO L'USO PERSONALE DELLO STESSO DA PARTE DELLO STUDENTE A FINI DI STUDIO. NE È VIETATA NEL MODO PIÙ ASSOLUTO LA DIFFUSIONE, DUPLICAZIONE, CESSIONE, TRASMISSIONE, DISTRIBUZIONE A TERZI O AL PUBBLICO PENA LE SANZIONI APPLICABILI PER LEGGE

Python

- Uno dei linguaggi di programmazione più usati al mondo
- tipizzazione dinamica
- orientato agli oggetti
- linguaggio interpretato
- ampia libreria interna
- moltissimi moduli esterni



Linguaggio Compilato vs Linguaggio Interpretato



Linguaggio Compilato vs Linguaggio Interpretato

Linguaggio Compilato

- Il compilatore traduce il codice direttamente in linguaggio macchina

✓ In genere è più efficiente

□ L'eseguibile generato dal compilatore è specifico per la piattaforma

Linguaggio Interpretato

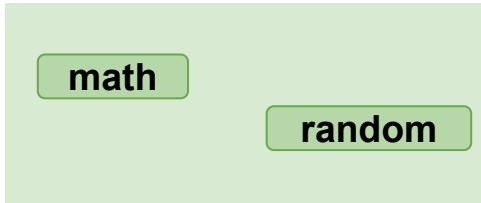
- Non c'è un compilatore
- L'interprete è un programma che esegue direttamente il codice sorgente (convertito in bytecode) un'istruzione alla volta

✓ Di solito è meno efficiente

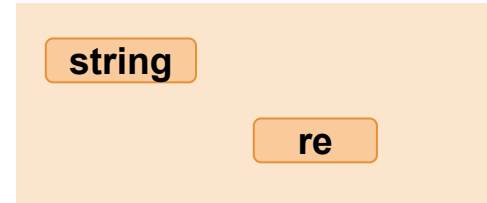
□ Cross-platform, lo stesso codice può essere eseguito su più piattaforme, a patto che esista un'implementazione dell'interprete per la piattaforma in questione

Python Standard Library

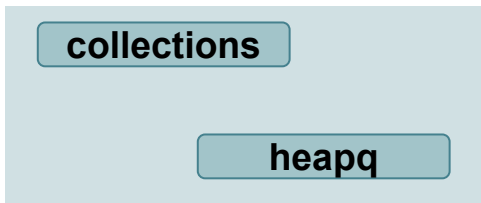
Matematica



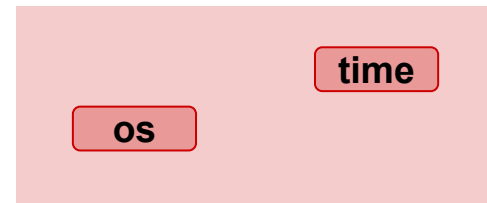
Manipolazione Testo



Collezioni



Primitive Sistema Operativo

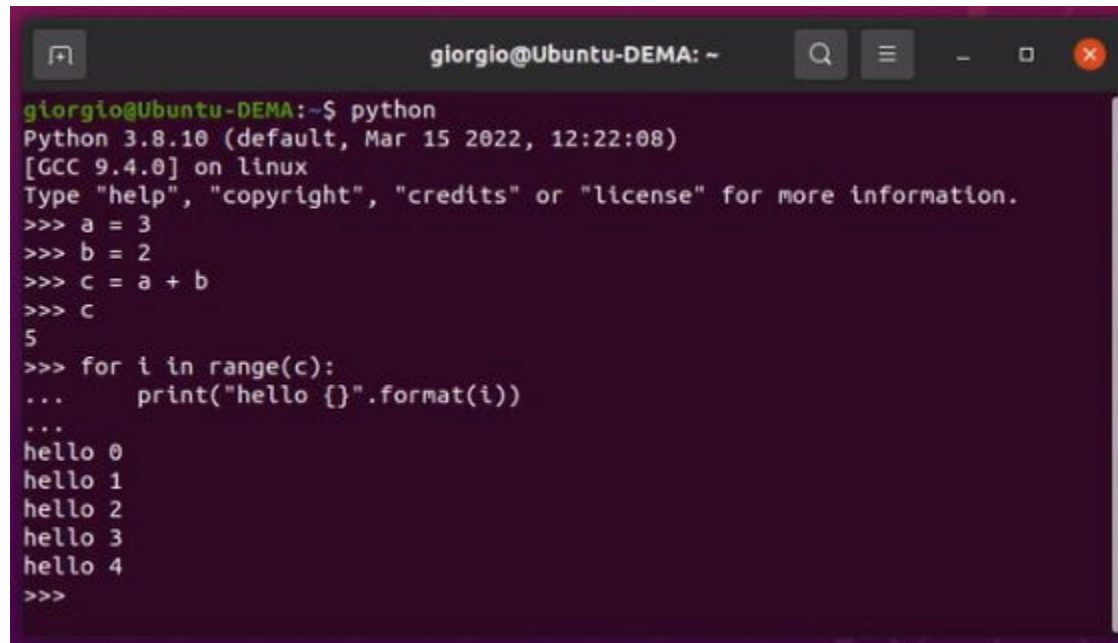


Python External Libraries



Interprete Python

- L'interprete Python può eseguire comandi in modo interattivo

A screenshot of a terminal window titled 'giorgio@Ubuntu-DEMA: ~'. The terminal shows the execution of the 'python' command, which starts the Python 3.8.10 interpreter. The user enters several commands: 'a = 3', 'b = 2', 'c = a + b', and 'c', which outputs '5'. Then, a loop is executed: 'for i in range(c):' followed by 'print("hello {}".format(i))' on subsequent lines, which outputs 'hello 0', 'hello 1', 'hello 2', 'hello 3', and 'hello 4'. The prompt '>>>' is visible at the end of the last line.

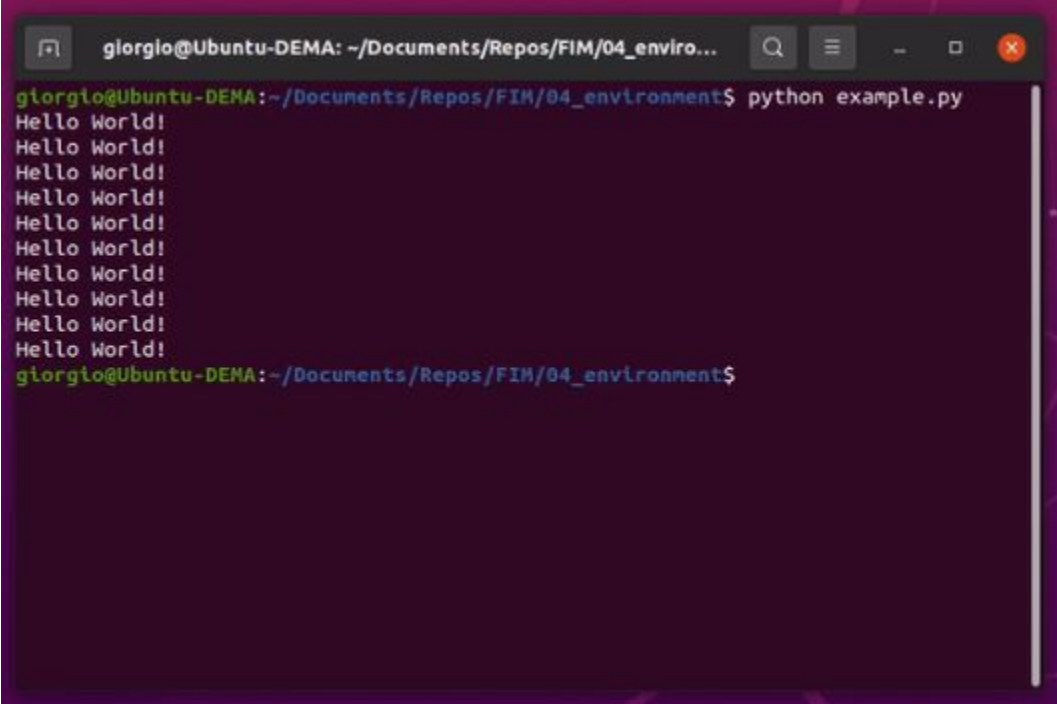
```
giorgio@Ubuntu-DEMA:~$ python
Python 3.8.10 (default, Mar 15 2022, 12:22:08)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> a = 3
>>> b = 2
>>> c = a + b
>>> c
5
>>> for i in range(c):
...     print("hello {}".format(i))
...
hello 0
hello 1
hello 2
hello 3
hello 4
>>>
```

Interprete Python

- Oppure può eseguire un programma salvato in un file con estensione **.py**

example.py

```
if __name__=="__main__":  
    for i in range(10):  
        print("Hello World!")
```

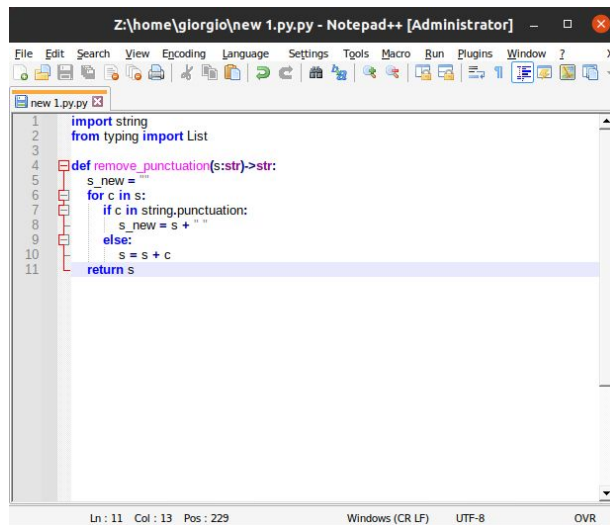
A terminal window with a dark purple background. The title bar shows the user 'glorgio' on a machine named 'Ubuntu-DEMA' at the directory '~/Documents/Repos/FIM/04_enviro...'. The prompt is 'glorgio@Ubuntu-DEMA:~/Documents/Repos/FIM/04_environment\$'. The command 'python example.py' has been entered and executed, resulting in ten lines of 'Hello World!' output. The prompt is now 'glorgio@Ubuntu-DEMA:~/Documents/Repos/FIM/04_environment\$' again.

```
glorgio@Ubuntu-DEMA:~/Documents/Repos/FIM/04_enviro...  
glorgio@Ubuntu-DEMA:~/Documents/Repos/FIM/04_environment$ python example.py  
Hello World!  
Hello World!  
Hello World!  
Hello World!  
Hello World!  
Hello World!  
Hello World!  
Hello World!  
Hello World!  
Hello World!  
glorgio@Ubuntu-DEMA:~/Documents/Repos/FIM/04_environment$
```


Integrated Development Environment

Text Editor

- Code Editing
- Syntax Highlighting



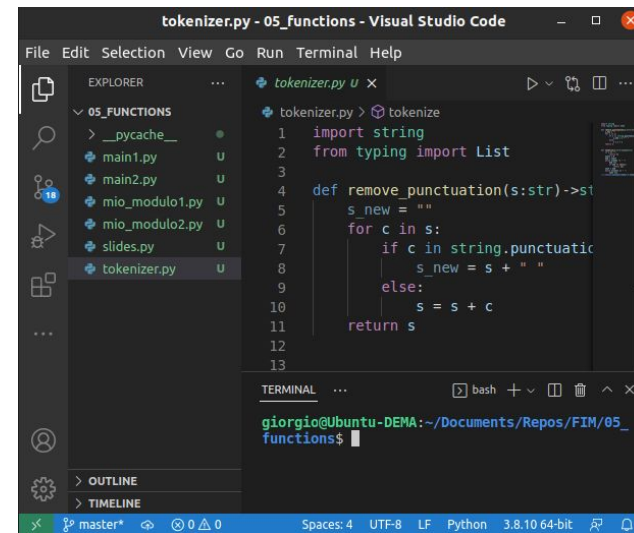
A screenshot of the Notepad++ text editor. The title bar reads "Z:\home\giorgio\new 1.py.py - Notepad++ [Administrator]". The menu bar includes File, Edit, Search, View, Encoding, Language, Settings, Tools, Macro, Run, Plugins, Window, and Help. The toolbar contains various icons for file operations and editing. The code editor shows a Python script with the following content:

```
1 import string
2 from typing import List
3
4 def remove_punctuation(s:str)->str:
5     s_new = ""
6     for c in s:
7         if c in string.punctuation:
8             s_new = s + " "
9         else:
10            s = s + c
11    return s
```

The status bar at the bottom indicates "Ln : 11 Col : 13 Pos : 229", "Windows (CR LF)", "UTF-8", and "OVR".

Integrated Development Environment (IDE)

- Code Editing
- Syntax Highlighting
- Integrated Terminal
- Integrated Debugger
- Versioning Control



A screenshot of the Visual Studio Code IDE. The title bar reads "tokenizer.py - 05_functions - Visual Studio Code". The menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The Explorer sidebar on the left shows a file tree with "05_FUNCTIONS" expanded, containing files like __pycache__, main1.py, main2.py, mio_modulo1.py, mio_modulo2.py, slides.py, and tokenizer.py. The main editor shows the content of "tokenizer.py" with the following code:

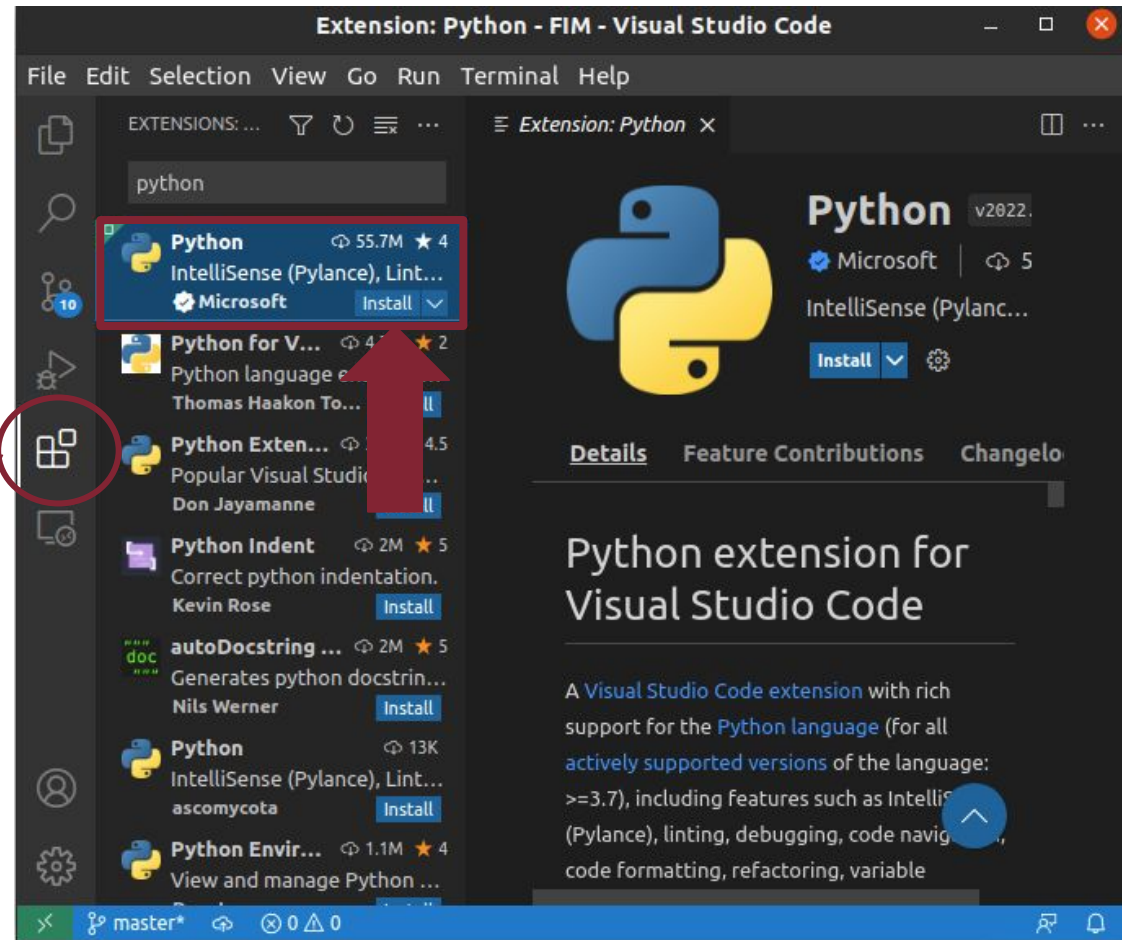
```
1 import string
2 from typing import List
3
4 def remove_punctuation(s:str)->str:
5     s_new = ""
6     for c in s:
7         if c in string.punctuation:
8             s_new = s + " "
9         else:
10            s = s + c
11    return s
```

The TERMINAL panel at the bottom shows a bash shell with the prompt "giorgio@Ubuntu-DEMA:~/Documents/Repos/FIM/05_functions\$". The status bar at the bottom indicates "master*", "0 0 0", "Spaces: 4", "UTF-8", "LF", "Python", "3.8.10 64-bit", and a search icon.

Environment Setup

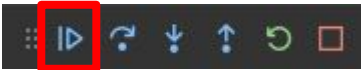
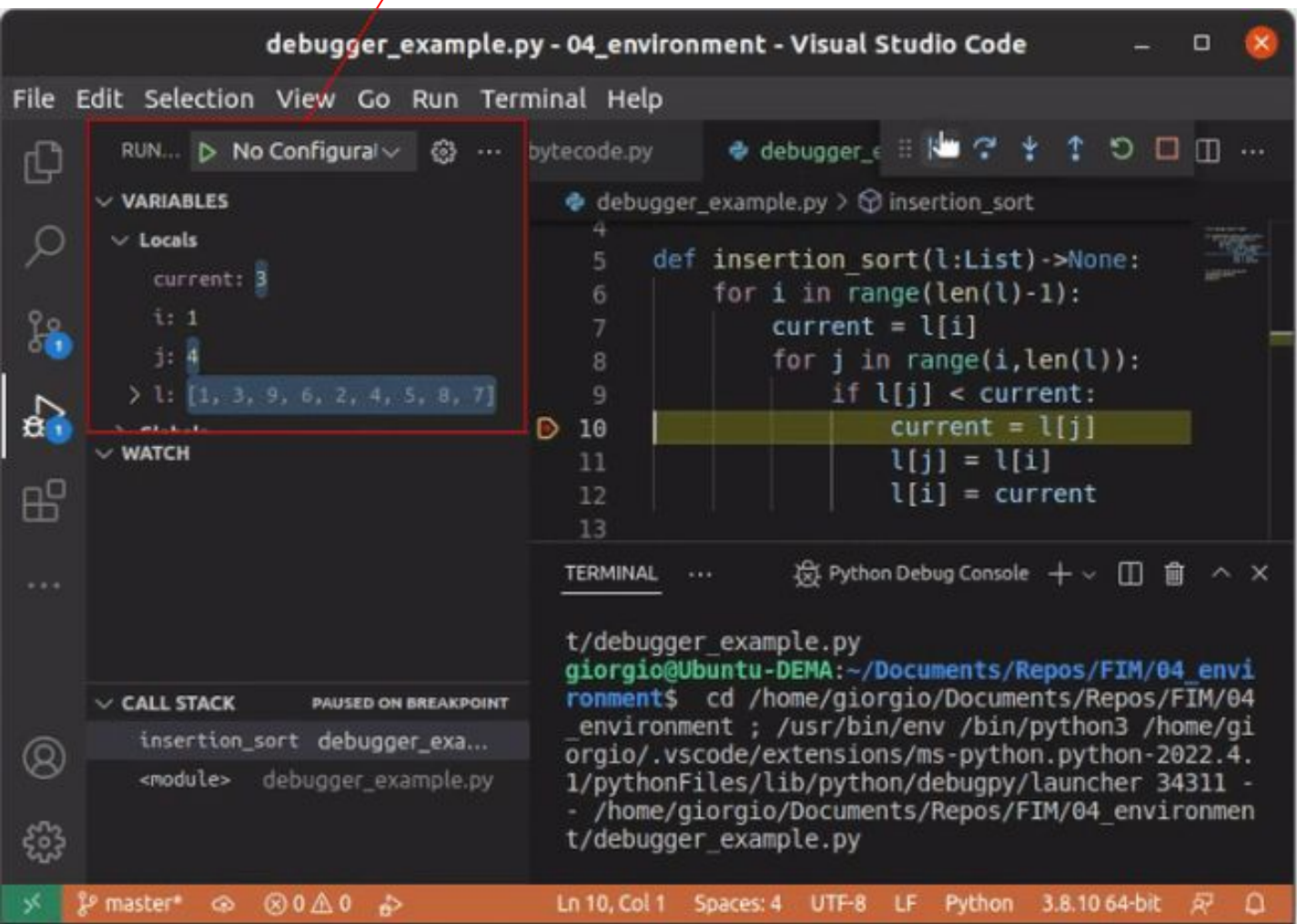
- Download the [Python Interpreter](#)
- Download [Visual Studio Code IDE](#)
- Install Python Extension in VS Code
- Enjoy Programming :)

Extensions
Marketplace



Debugger

Variables Explorer



Esegui fino al prossimo breakpoint



Salta alla riga successiva



Entra nella funzione



Esci dalla funzione (viene eseguita comunque)



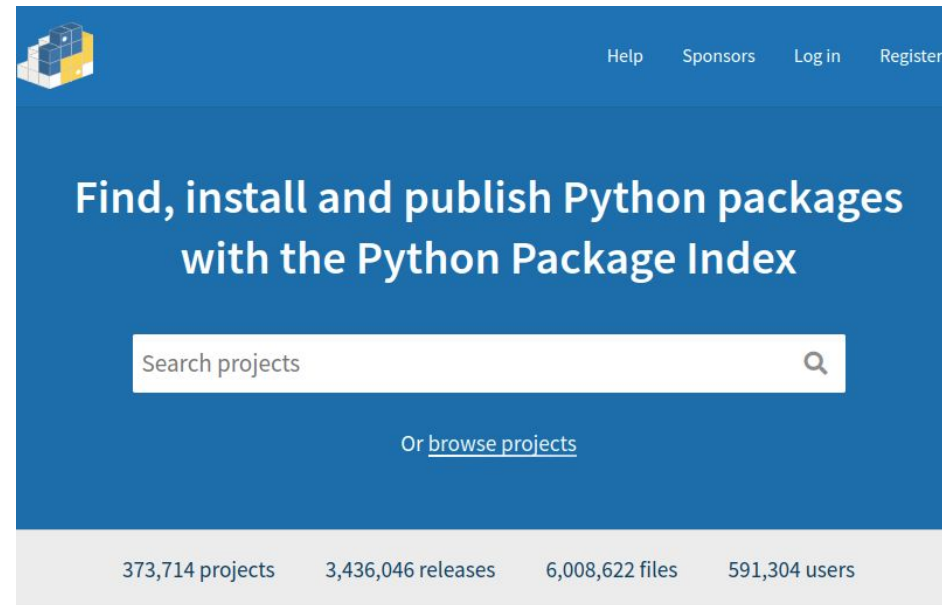
Ricomincia



Ferma


Python Package Manager (pip)

- Permette di installare i packages esterni
- I package devono essere presenti nel Python Package Index
- Per installare un package usare il comando:
 - `$ pip install package_name`
-
- Per esempio per installare jupyter:
 - `$ pip install numpy`
- Per fare una lista dei package installati:
 - `$ pip list`



Python Data Model

- Ogni entità che esiste nel programma è un oggetto
- Ogni oggetto è identificato da un indirizzo in memoria, che si può consultare utilizzando la funzione `id(obj)`
- Ogni oggetto ha un tipo (anch'esso un oggetto) accessibile chiamando la funzione `type(obj)`

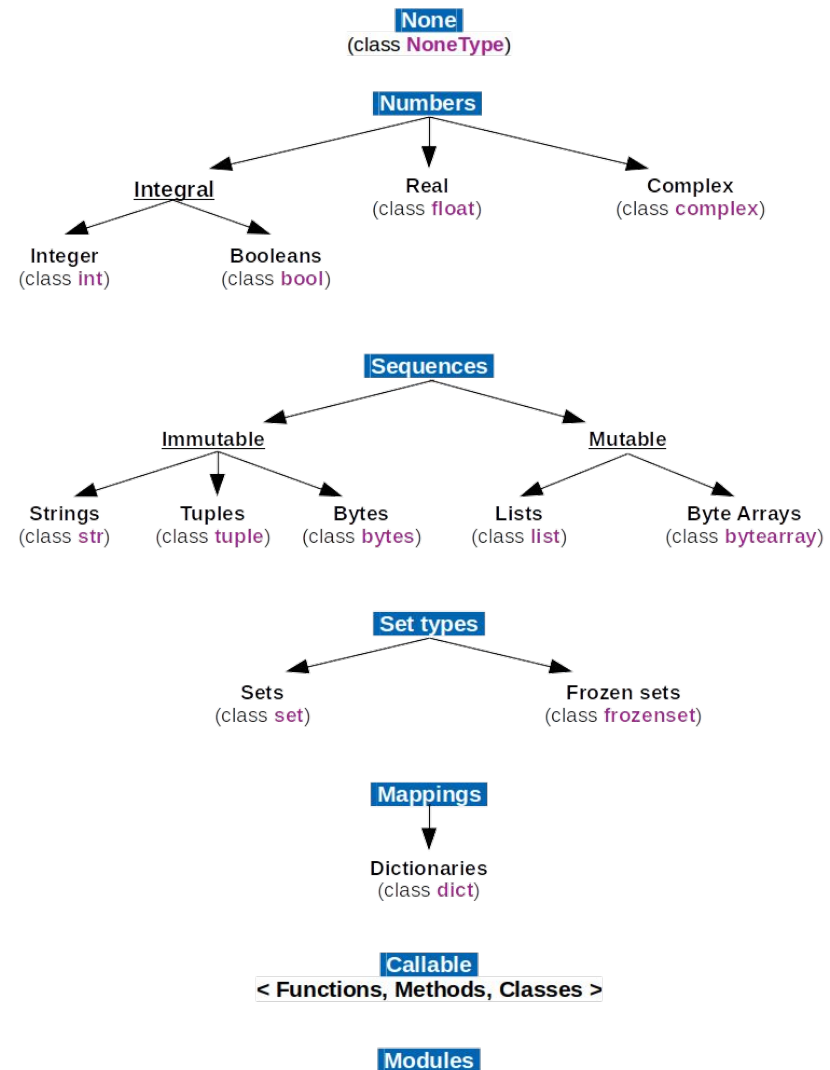


```
IDLE Shell 3.8.10
File Edit Shell Debug Options Window Help
Python 3.8.10 (default, Mar 15 2022, 12:22:08)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>> a = 1
>>> type(a)
<class 'int'>
>>> b = 2
>>> type(b)
<class 'int'>
>>> hex(id(a))
'0x955e20'
>>> hex(id(b))
'0x955e40'
>>> hex(id(type(a)))
'0x90b400'
>>> hex(id(type(b)))
'0x90b400'
>>> c = 1
>>> hex(id(c))
'0x955e20'
>>> a = []
>>> b = []
>>> hex(id(a))
'0x7f09cb8181c0'
>>> hex(id(b))
'0x7f09ca0e5480'
>>> |
```

Ln: 27 Col: 4

Type Hierarchy

- In Python esistono oggetti Mutabili e oggetti Immutabili
- Gli oggetti mutabili possono cambiare il loro contenuto (list,set,dict)
- Gli oggetti immutabili non possono cambiare il loro contenuto (Numbers, str,tuple)



Garbage Collector

- Se assegno un oggetto ad una variabile essa contiene un riferimento all'oggetto
- L'interprete Python mantiene una tabella per contare il numero di riferimenti per ogni oggetto
- Quando un oggetto non è più referenziato (diventa inaccessibile) la sua memoria viene liberata dal Garbage Collector

Object ID	Refs Counter
0x7fea47569680	1



```
a = [1,2,3,4]
print(hex(id(a))) # prints 0x7fea47569680

b = a

c = [1,2]

c[0] = a

b = None


c[0] = 1

a = 3
```

Garbage Collector

- Se assegno un oggetto ad una variabile essa contiene un riferimento all'oggetto
- L'interprete Python mantiene una tabella per contare il numero di riferimenti per ogni oggetto
- Quando un oggetto non è più referenziato (diventa inaccessibile) la sua memoria viene liberata dal Garbage Collector

Object ID	Refs Counter
0x7fea47569680	2



```
a = [1,2,3,4]
print(hex(id(a))) # prints 0x7fea47569680

b = a

c = [1,2]

c[0] = a

b = None


c[0] = 1

a = 3
```


Garbage Collector

- Se assegno un oggetto ad una variabile essa contiene un riferimento all'oggetto
- L'interprete Python mantiene una tabella per contare il numero di riferimenti per ogni oggetto
- Quando un oggetto non è più referenziato (diventa inaccessibile) la sua memoria viene liberata dal Garbage Collector

Object ID	Refs Counter
0x7fea47569680	3



```
a = [1,2,3,4]
print(hex(id(a))) # prints 0x7fea47569680

b = a

c = [1,2]

c[0] = a

b = None


c[0] = 1

a = 3
```

Garbage Collector

- Se assegno un oggetto ad una variabile essa contiene un riferimento all'oggetto
- L'interprete Python mantiene una tabella per contare il numero di riferimenti per ogni oggetto
- Quando un oggetto non è più referenziato (diventa inaccessibile) la sua memoria viene liberata dal Garbage Collector

Object ID	Refs Counter
0x7fea47569680	2



```
a = [1,2,3,4]
print(hex(id(a))) # prints 0x7fea47569680

b = a

c = [1,2]

c[0] = a

b = None


c[0] = 1

a = 3
```

Garbage Collector

- Se assegno un oggetto ad una variabile essa contiene un riferimento all'oggetto
- L'interprete Python mantiene una tabella per contare il numero di riferimenti per ogni oggetto
- Quando un oggetto non è più referenziato (diventa inaccessibile) la sua memoria viene liberata dal Garbage Collector

Object ID	Refs Counter
0x7fea47569680	1



```
a = [1,2,3,4]
print(hex(id(a))) # prints 0x7fea47569680

b = a

c = [1,2]

c[0] = a

b = None

c[0] = 1

a = 3
```

Garbage Collector

- Se assegno un oggetto ad una variabile essa contiene un riferimento all'oggetto
- L'interprete Python mantiene una tabella per contare il numero di riferimenti per ogni oggetto
- Quando un oggetto non è più referenziato (diventa inaccessibile) la sua memoria viene liberata dal Garbage Collector

Object ID	Refs Counter
0x7fea47569680	0

```
a = [1,2,3,4]
print(hex(id(a)))    # prints 0x7fea47569680

b = a

c = [1,2]

c[0] = a

b = None

c[0] = 1

a = 3
```

The object is garbage collected !!



Slides distribuite con Licenza Creative Commons (CC BY-NC-ND 4.0) Attribuzione - Non commerciale - Non opere derivate 4.0 Internazionale

PUOI CONDIVIDERLE ALLE SEGUENTI CONDIZIONI

(riprodurre, distribuire, comunicare o esporre in pubblico, rappresentare, eseguire e recitare questo materiale con qualsiasi mezzo e formato)

Attribuzione*

Devi riconoscere una menzione di paternità adeguata, fornire un link alla licenza e indicare se sono state effettuate delle modifiche. Puoi fare ciò in qualsiasi maniera ragionevole possibile, ma non con modalità tali da suggerire che il licenziante avalli te o il tuo utilizzo del materiale.

Non Commerciale

Non puoi utilizzare il materiale per scopi commerciali.

Non opere derivate

Se remixi, trasformi il materiale o ti basi su di esso, non puoi distribuire il materiale così modificato.

Divieto di restrizioni aggiuntive

Non puoi applicare termini legali o misure tecnologiche che impongano ad altri soggetti dei vincoli giuridici a questa licenza