

# Introduzione ai laboratori e Installazione dell'Ambiente di Lavoro

## Contents

- 1.1. Installazione di Python
- 1.2. Programmare in Python
- 1.3. Google Colab

Questo corso include ore di laboratorio in cui affronteremo manipolazioni pratiche di analisi dei dati. Tutti i laboratori verranno condotti utilizzando il linguaggio Python e il notebook Jupyter (tramite un'installazione locale o Google Colab). Inizieremo vedendo come installare l'ambiente di lavoro.

## 1.1. Installazione di Python

Nel corso, utilizzeremo il linguaggio Python. Scegliamo Python in quanto un linguaggio moderno, diffuso, multi-piattaforma, portabile, facile da usare ed elegante e dotato di una modalità interattiva. Questa ultima caratteristica risulterà molto utile per sperimentare con il codice e apprendere bene il funzionamento del linguaggio e delle librerie.

Python è un linguaggio fortemente modulare: alcune funzionalità di base sono incluse nel linguaggio, mentre molte altre sono fornite da pacchetti di terze parti, gestite mediante un vero e proprio gestore dei pacchetti chiamato `pip`. Per avere un ambiente di lavoro pronto per l'utilizzo in applicazioni che richiedono l'analisi dei

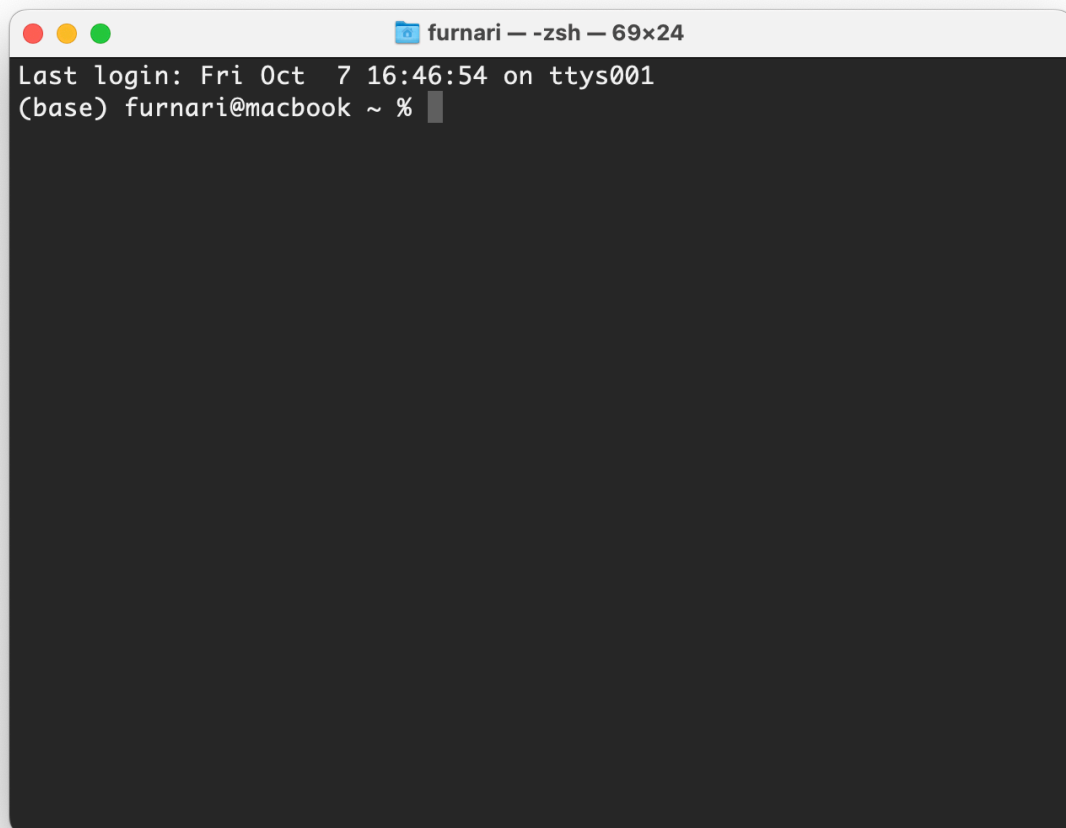
dati, utilizzeremo una distribuzione chiamata [Anaconda](#). Va notato che in passato esistevano due branch di Python: il 2.x e il 3.x. Oggi il 2.x non è più utilizzato. La distribuzione di Anaconda utilizzata nel momento in cui si scrive installerà Python 3.9.

Scarichiamo la distribuzione per la nostra piattaforma dal seguente link:

<https://www.anaconda.com/products/distribution> e installiamola. Una volta fatto ciò, dovremmo poter accedere a un terminale con l'accesso agli eseguibili messi a disposizione da Anaconda. In particolare:

- Su windows, sarà disponibile il programma "Anaconda Prompt";
- Su MacOS e su Linux gli eseguibili dovrebbero già trovarsi nel path dell'utente e dunque dovrebbe essere sufficiente aprire un semplice terminale.

Se anaconda è stato correttamente installato, apparirà l'indicazione `(base)` prima del prompt dei comandi, come mostrato di seguito:



```
furnari — zsh — 69x24
Last login: Fri Oct 7 16:46:54 on ttys001
(base) furnari@macbook ~ %
```

`(base)` indica l'`environment` (di default è l'environnement di base) in cui stiamo operando. Anaconda infatti permette di definire e utilizzare diversi environment indipendenti tra di loro in cui è possibile installare pacchetti diversi (anche in termini di versioni).

## 1.2. Programmare in Python

Con Anaconda verranno installati diversi strumenti. Tra questi:

- L'interprete python;
- La shell interattiva ipython;
- L'IDE Spyder;
- Jupyter Notebook.

### 1.2.1. Interprete Python

L'interprete Python permette di eseguire un programma Python mediante un comando del genere:

```
python programma.py
```

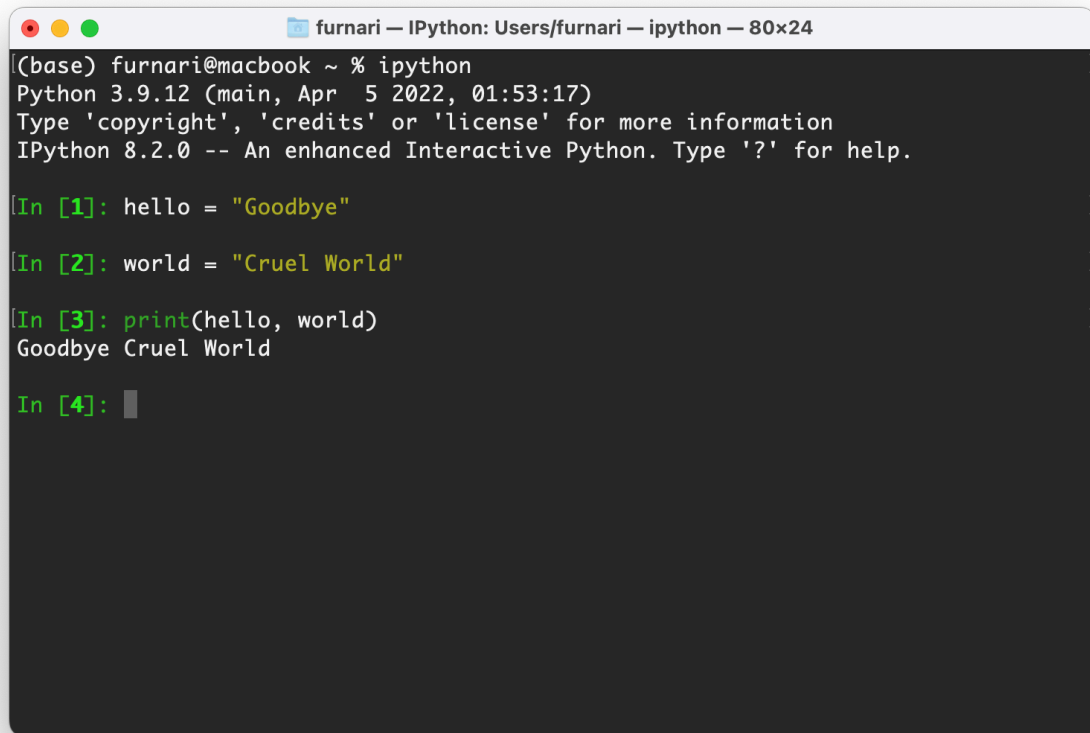
Dove `programma.py` è il file di testo sorgente del programma. Alla prima esecuzione, Python compilerà il programma e genererà un file bytecode `programma.pyc`.

### 1.2.2. Shell interattiva ipython

La shell interattiva è una versione più "evoluta" dell'interprete Python che permette di:

- Interpretare comandi;
- Eseguire programmi Python;
- Analizzare il contenuto delle variabili del workspace;

Possiamo lanciare la shell interattiva con il comando `ipython`:

A screenshot of a terminal window titled "furnari — IPython: Users/furnari — ipython — 80x24". The terminal shows the command prompt "(base) furnari@macbook ~ %" followed by the command "ipython". The output shows the IPython shell version 8.2.0 and the Python version 3.9.12. The user then enters four lines of code: "hello = 'Goodbye'", "world = 'Cruel World'", "print(hello, world)", and "In [4]:". The output of the print statement is "Goodbye Cruel World".

```
(base) furnari@macbook ~ % ipython
Python 3.9.12 (main, Apr  5 2022, 01:53:17)
Type 'copyright', 'credits' or 'license' for more information
IPython 8.2.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: hello = "Goodbye"

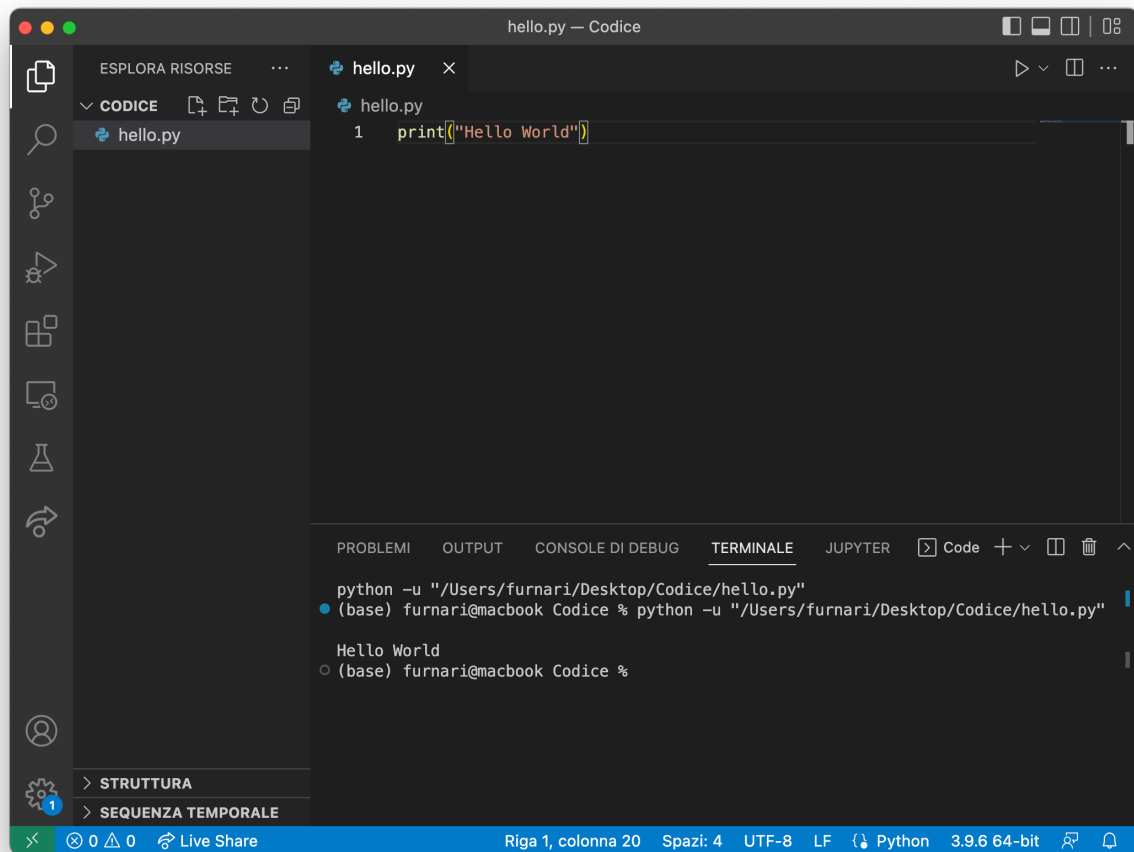
In [2]: world = "Cruel World"

In [3]: print(hello, world)
Goodbye Cruel World

In [4]:
```

### 1.2.3. IDE

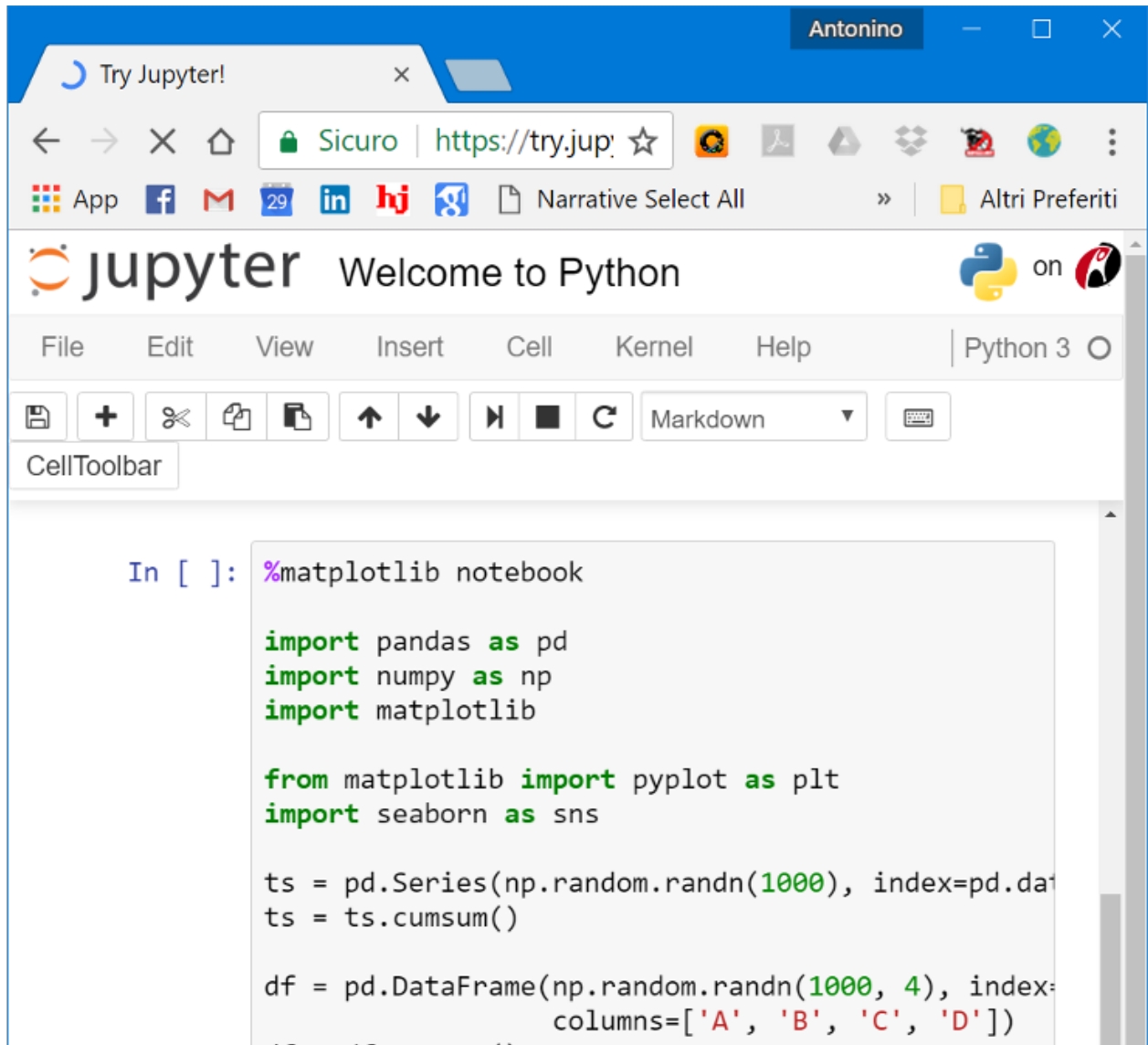
Un IDE generalmente integra una shell ipython e diversi strumenti per il debugging. É un ottimo strumento per progetti di medie o grandi dimensioni. Un ide molto usato per la programmazione in Python è [Visual Studio Code](https://code.visualstudio.com/).



## 1.2.4. Notebook Jupyter

È possibile avviare jupyter mediante il comando:

```
jupyter notebook
```



Jupyter permette di creare (mediante interfaccia web) dei “notebook”, ovvero degli archivi contenenti:

- Testo formattato;
- Il codice da eseguire;
- Le immagini ottenuti come risultati dell’esecuzione del codice.

Si tratta di uno strumento molto potente, in quanto permette di generare dei veri e propri report delle sperimentazioni.

## 1.2.5. IDE vs Notebook

C’è da chiedersi quando sia utile usare uno degli strumenti rispetto agli altri.

Gli IDE sono ottimi per:

- medi o grossi progetti, con diversi moduli e classi;
- computazione interattiva (ad esempio elaborare un video in tempo reale).

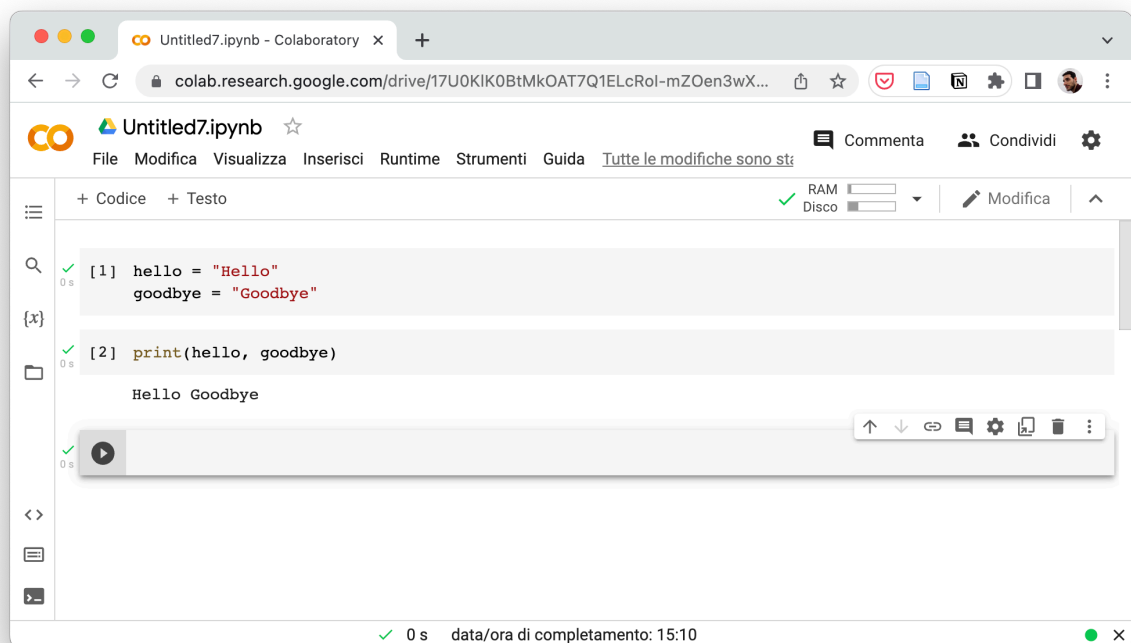
I Notebook di Jupyter sono ottimi per:

- documentare i processi di analisi dei dati sperimentare nuove idee;
- scrivere documentazione e tutorials.

## 1.3. Google Colab

In alternativa alla installazione di un ambiente di lavoro sul proprio computer, è possibile utilizzare un servizio gratuito messo a disposizione da Google chiamato "Colab": <https://colab.research.google.com/>

Il servizio mette a disposizione dei notebook in Python (simili a quelli di Jupyter) con supporto GPU.



In Google Colab, troveremo i principali pacchetti installati, ma possiamo installare

dei pacchetti aggiuntivi mediante comandi del tipo:

```
pip install sklearn
```

## 1.3.1. Risorse limitate

Google Colab è un ottimo strumento per sperimentare o fare piccoli esperimenti. Tuttavia, tenete a mente che le risorse sono limitate e l'accesso alla GPU potrebbe essere revocato dopo alcune ore di computazione o se il computer è inattivo per diverso tempo. Pertanto, potrebbe essere difficile usarlo per progetti medio-grandi.

## 1.3.2. Persistenza dei dati

Ad ogni esecuzione di un notebook, Google Colab mette a disposizione uno spazio temporaneo su cui è possibile scrivere e dal quale è possibile leggere dei dati. Ogni volta che il notebook è chiuso e riaperto, lo spazio viene liberato e il contenuto eliminato. Se si vuole avere accesso a uno spazio persistente, è possibile montare una specifica cartella del proprio Google Drive eseguendo una cella con il seguente codice:

```
from google.colab import drive
drive.mount('/content/drive/path/to/folder')
```

Dove `/path/to/folder` è il path alla cartella all'interno del drive.

Previous

< [Lecture Notes on  
Fundamental of Data Analysis](#)

Next >

[2. Introduzione a Python](#)