

# Cosa è Python?

## Linguaggio

- Interpretato
- Interattivo
- Ad oggetti
- Incorpora
  - ▶ moduli
  - ▶ eccezioni
  - ▶ tipizzazione dinamica
  - ▶ tipi di dati dinamici di alto livello
  - ▶ classi
- Molto potente, sintassi chiara
- Portabile

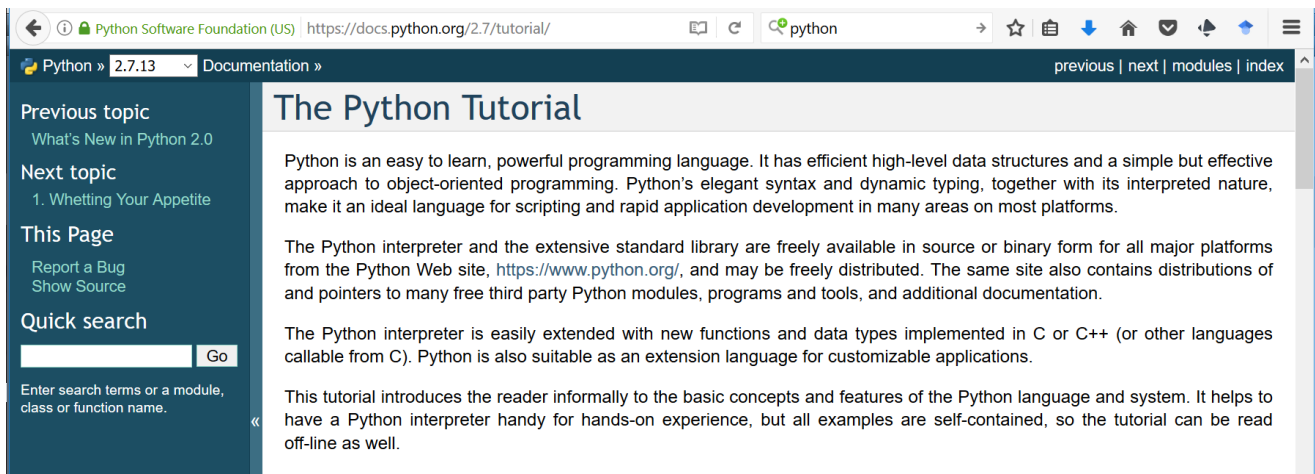
<https://docs.python.org/3/faq/general.html>

## Per cosa è utile

- Python è un linguaggio di programmazione general-purpose con un'ampia *standard library* per:
  - ▶ String processing (espressioni regolari, Unicode, differenze tra file)
  - ▶ Protocolli Internet (HTTP, FTP, SMTP, XML-RPC, POP, IMAP, programmazione CGI)
  - ▶ Ingegneria del software (unit testing, logging, profiling, parsing Python code)
  - ▶ Interfacce per sistemi operativi (system calls, filesystems, TCP/IP sockets)
- E soprattutto molte altre estensioni

<https://docs.python.org/3/faq/general.html>

# Libri?!?



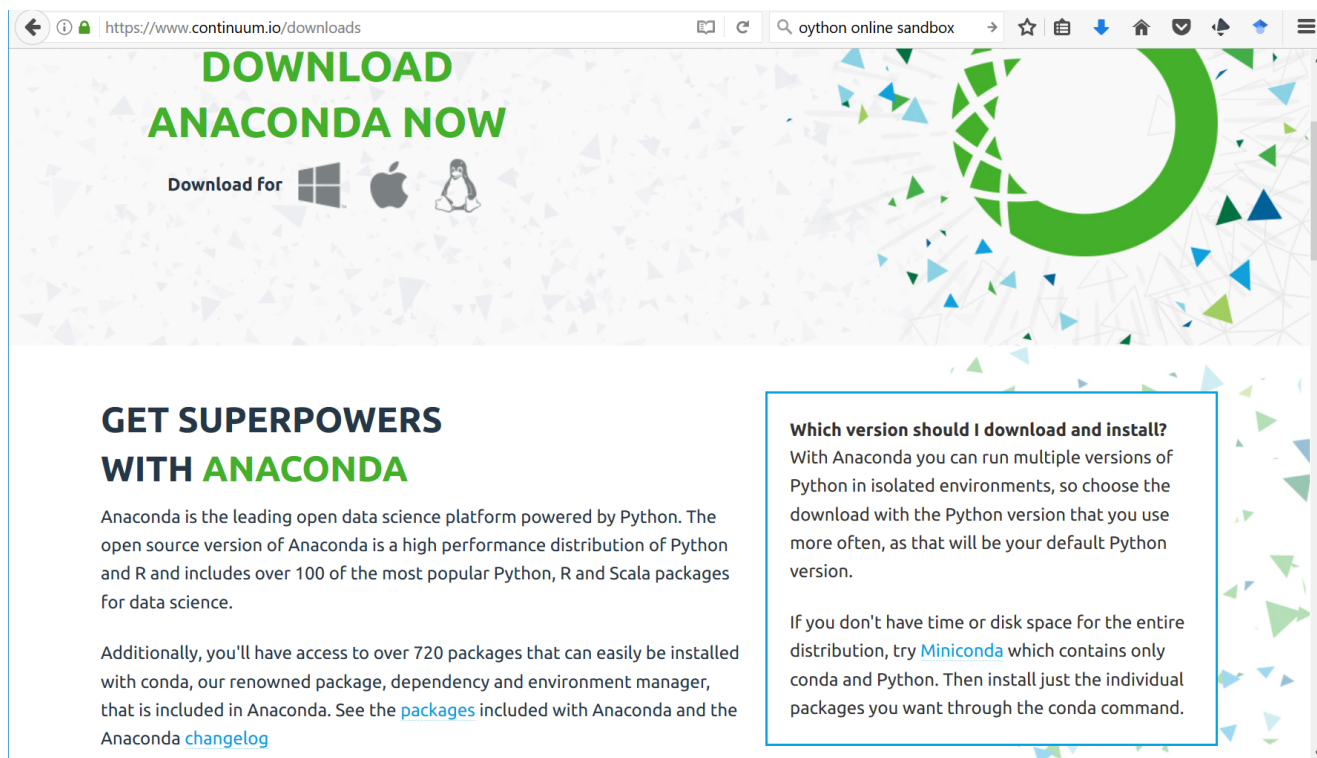
- <https://docs.python.org/3.7/tutorial/>
- 2.\* vs 3.\*

## Alternative per sviluppo SW

- Installare Python (<https://www.python.org/>)
  - ▶ Usare la shell
  - ▶ eseguire un programma .py
- Usare Python in una Console online
  - ▶ <http://www.python.org>
  - ▶ <https://www.tutorialspoint.com/python/>
- Installare IPython (Jupyter - <http://jupyter.org/>)
  - ▶ Shell interattiva con completamento con tab, history...
- Usare IPython online  
([https://www.tutorialspoint.com/ipython\\_terminal\\_online.php](https://www.tutorialspoint.com/ipython_terminal_online.php))
- Jupyter Notebook: Web-based interactive computational environment
  - ▶ Usare un Notebook online (<https://try.jupyter.org/>)
- Installare una piattaforma.  
Esempio Anaconda (distribuzione di Python con più di 100 package, NumPy, Pandas, SciPy, Matplotlib, Jupyter ...)

# Install Anaconda

<https://www.continuum.io/downloads>



## The Shell

- Si invoca python dalla linea di comando
- Utile per matematica di base, per provare idee
- Non si scrivono programmi completi nell'interprete
- Non si può salvare ciò che si scrive

### Interprete

```
>>>print("Hello, World!")
Hello, World!
>>>var = 9+2
>>>var*11
121
```

## Jupyter Notebook

- Web-based interactive computational environment for creating IPython notebooks
- An IPython notebook is a JSON document containing an ordered list of input/output cells which can contain code, text, mathematics, plots and rich media
- IPython notebooks can be converted to a number of open standard output formats (HTML, presentation slides, LaTeX, PDF, ReStructuredText, Markdown, Python)

- ▶ 'Download As' in the web interface
- ▶ nbconvert in a shell

```
jupyter nbconvert Test.ipynb --to latex
```

## Install Jupyter

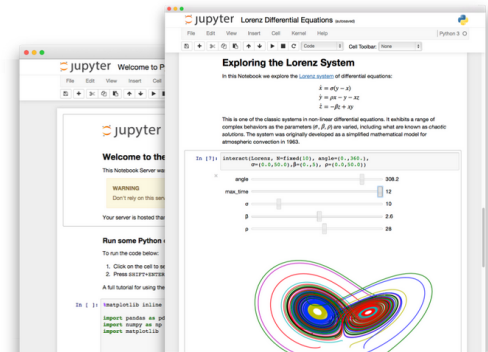
The screenshot shows the Jupyter website's 'Installing Jupyter' page. At the top, there is a navigation bar with links: Install, About, Resources, Documentation, NBViewer, Widgets, Blog, and Donate. The main heading is 'Installing Jupyter', followed by a subheading 'Get up and running with the Jupyter Notebook on your computer within minutes! Follow the instructions below.' The page content includes a section for 'Prerequisite: Python' which states that Python 3.3 or greater, or Python 2.7, is required. It then recommends using the Anaconda distribution for installation. Below this, there is a section for 'Installing Jupyter using Anaconda and conda' which also recommends Anaconda and provides a list of steps: 1. Download Anaconda (latest Python 3 version, currently Python 3.5). 2. Install the version of Anaconda which you downloaded, following the instructions on the download page. 3. Congratulations, you have installed Jupyter Notebook. To run the notebook: 

```
jupyter notebook
```

Incluso in Anaconda

poi: jupyter notebook

# Jupyter

[Install](#)[About](#)[Resources](#)[Documentation](#)[NBViewer](#)[Widgets](#)[Blog](#)[Donate](#)

## The Jupyter Notebook

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and explanatory text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, machine learning and much more.



Language of choice



Share notebooks



Interactive widgets



Big data integration

February 12, 2019

9/30

# Jupyter



Hosted by Rackspace

[Files](#) [Running](#) [Clusters](#)

Select items to perform actions on them.

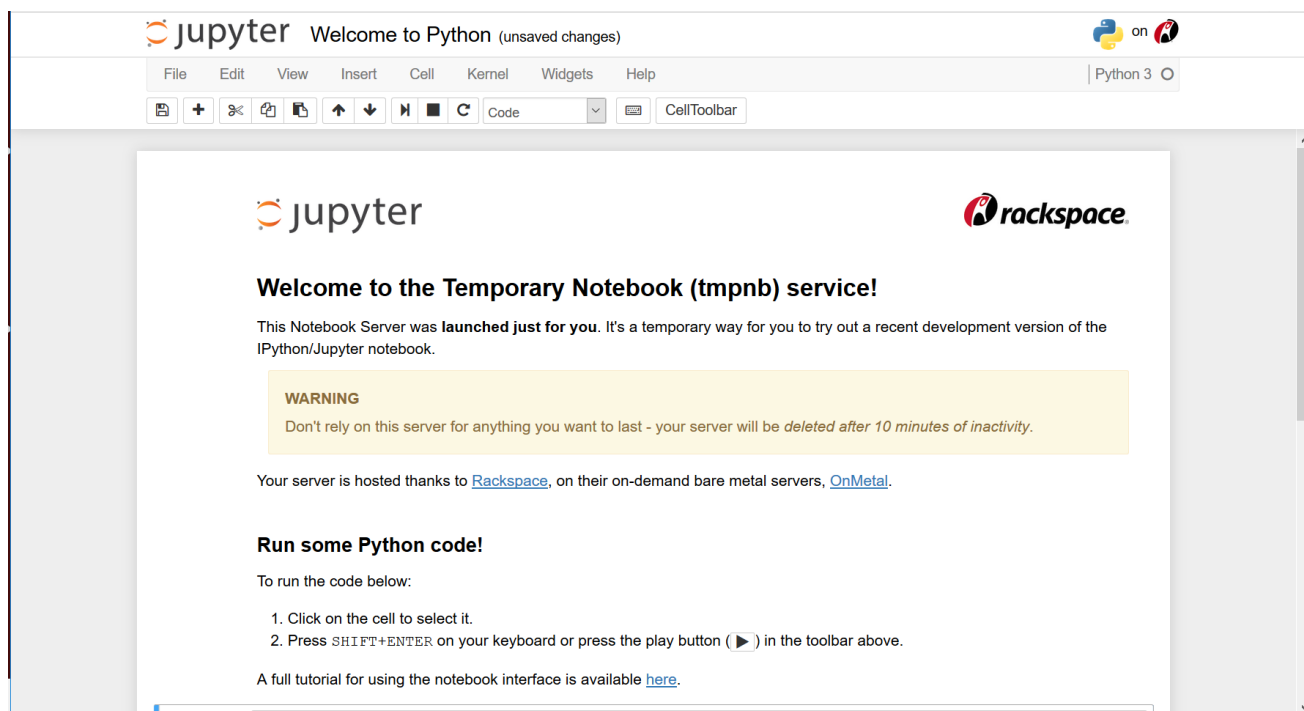
[Upload](#) [New](#)

- ☐
- ☐ communities
- ☐ datasets
- ☐ featured
- ☐ Welcome Julia - Intro to Gadfly.ipynb
- ☐ Welcome R - demo.ipynb
- ☐ Welcome to Haskell.ipynb
- ☐ Welcome to Python.ipynb
- ☐ Welcome to Spark with Python.ipynb
- ☐ Welcome to Spark with Scala.ipynb

February 12, 2019

10/30

# Jupyter



The screenshot shows the Jupyter Notebook interface with the title "Welcome to Python (unsaved changes)". The top menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. The toolbar contains icons for saving, creating new cells, and running code. The main content area displays the Jupyter logo, the Rackspace logo, and a welcome message to the Temporary Notebook (tmpnb) service. It includes a warning that the server will be deleted after 10 minutes of inactivity and instructions on how to run Python code.

**jupyter** Welcome to Python (unsaved changes) Python 3

**jupyter** **rackspace**

**Welcome to the Temporary Notebook (tmpnb) service!**

This Notebook Server was **launched just for you**. It's a temporary way for you to try out a recent development version of the IPython/Jupyter notebook.

**WARNING**  
Don't rely on this server for anything you want to last - your server will be deleted after 10 minutes of inactivity.

Your server is hosted thanks to [Rackspace](#), on their on-demand bare metal servers, [OnMetal](#).

**Run some Python code!**

To run the code below:

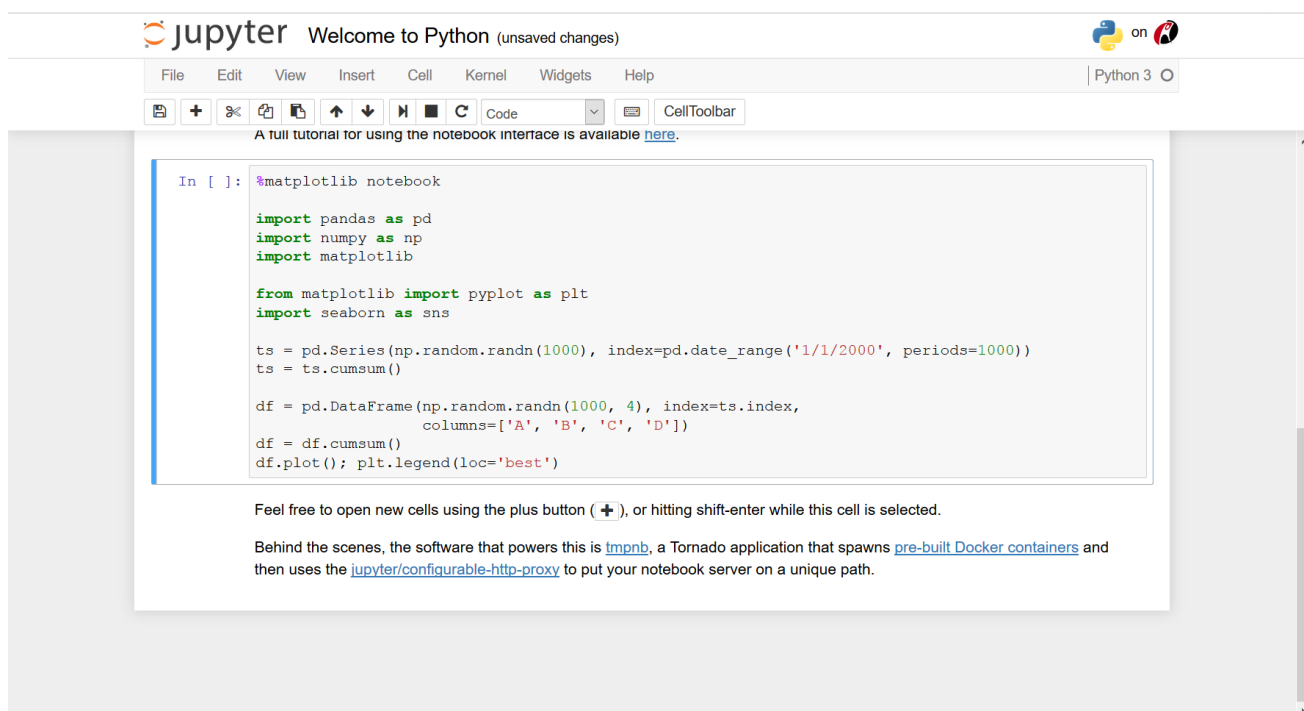
1. Click on the cell to select it.
2. Press **SHIFT+ENTER** on your keyboard or press the play button (▶) in the toolbar above.

A full tutorial for using the notebook interface is available [here](#).

February 12, 2019

11/30

# Jupyter



The screenshot shows the Jupyter Notebook interface with the title "Welcome to Python (unsaved changes)". The top menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. The toolbar contains icons for saving, creating new cells, and running code. The main content area displays the Jupyter logo, the Rackspace logo, and a welcome message to the Temporary Notebook (tmpnb) service. It includes a warning that the server will be deleted after 10 minutes of inactivity and instructions on how to run Python code. Below the welcome message, there is a code cell containing Python code for data analysis and visualization.

**jupyter** Welcome to Python (unsaved changes) Python 3

**jupyter** **rackspace**

**Welcome to the Temporary Notebook (tmpnb) service!**

This Notebook Server was **launched just for you**. It's a temporary way for you to try out a recent development version of the IPython/Jupyter notebook.

**WARNING**  
Don't rely on this server for anything you want to last - your server will be deleted after 10 minutes of inactivity.

Your server is hosted thanks to [Rackspace](#), on their on-demand bare metal servers, [OnMetal](#).

**Run some Python code!**

To run the code below:

1. Click on the cell to select it.
2. Press **SHIFT+ENTER** on your keyboard or press the play button (▶) in the toolbar above.

A full tutorial for using the notebook interface is available [here](#).

```
In [ ]: %matplotlib notebook

import pandas as pd
import numpy as np
import matplotlib

from matplotlib import pyplot as plt
import seaborn as sns

ts = pd.Series(np.random.randn(1000), index=pd.date_range('1/1/2000', periods=1000))
ts = ts.cumsum()

df = pd.DataFrame(np.random.randn(1000, 4), index=ts.index,
                  columns=['A', 'B', 'C', 'D'])
df = df.cumsum()
df.plot(); plt.legend(loc='best')
```

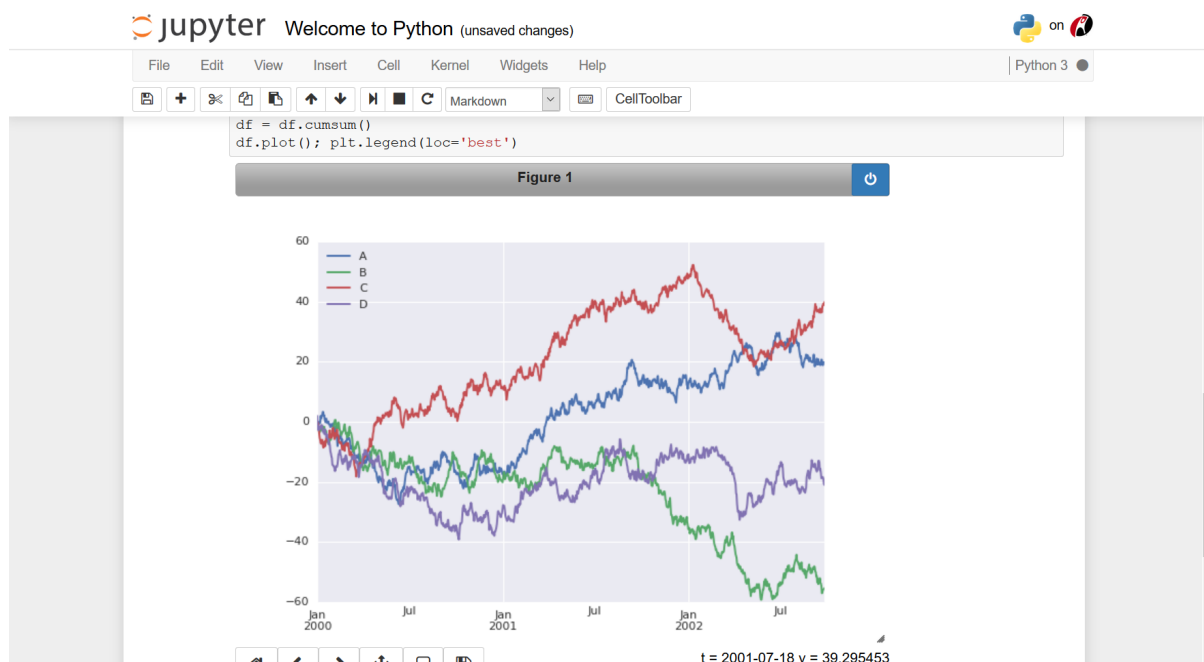
Feel free to open new cells using the plus button (+), or hitting shift-enter while this cell is selected.

Behind the scenes, the software that powers this is [tmpnb](#), a Tornado application that spawns [pre-built Docker containers](#) and then uses the [jupyter/configurable-http-proxy](#) to put your notebook server on a unique path.

February 12, 2019

12/30

# Jupyter



## Markdown

- Un modo per scrivere contenuto nel Web
- Scritto in "plaintext", caratteri normali con alcuni caratteri speciali
- Usato per commenti in GitHub
- Learning curve poco ripida (si impara in 10 minuti)
- Poche cose, in modo semplice (corsivo, neretto, headers, liste ...)
  - ▶ Corsivo: (`_`). Esempio `_corsivo_`
  - ▶ Neretto (`**`). Esempio `**neretto**`
  - ▶ Header (`#`): Esempi (`# Header One`)... (`### Header Three`).
  - ▶ Inline link: link text tra `[ ]` link tra parentesi `( )`  
Esempio: `[Visit GitHub] (www.github.com)`
  - ▶ Immagine come link: `![TestoAlternativo](http://xxx.jpg)`
  - ▶ Liste: `*` prima di ogni item
  - ▶ Oppure numeri

# Python

- # Un commento
- Python è "space sensitive"
- I blocchi di codice sono definiti dall'indentazione  
Le linee dopo un : devono essere indentate

## Esempio

```
for i in (1,2,3,4):  
    print (i),
```

```
1 2 3 4
```

# Stringhe

- Semplice stringa "hello world"
- Concatenazione: "hello"+" world" → "hello world"
- Ripetizione: "hello"\*3 → "hello hello hello "
- Indicizzazione: "world"[3] → "l"
  - ▶ Nota: liste python sono zero-offset
- Cercare: "o" in "hello" → True



# Numeri

- Notazione matematica di base:  $1.4$ ,  $2+2$ ,  $2^{**}10$ ,  $1e10$ 
  - ▶ Nota: Divisione intera è con floor:  $2/3 \rightarrow 0$ ,  $2./3 \rightarrow .6667$
- Funzioni matematiche richiedono `import math`
  - ▶ `import math`
  - ▶ `math.sqrt(4) \rightarrow 2.0`
  - ▶ `from math import *`
  - ▶ `sqrt(4) \rightarrow 2.0`

# Variabili e Liste

## Variabili Dynamically-Typed

- `x = 5`
- `x = 3.14`
- `x = 'text'`
- `x = '3.14'`

## Liste Dynamically-Typed

- `numeri = [0,1,2,3,4,5]`
- `parole = ['algoritmi','strutture']`
- `combinato = [12,23,['testa','croce']] + parole`

## Operatori su Liste

- `words.append('dati') → ['algoritmi','strutture','dati']`
- `words.insert(1,'e') → ['algoritmi','e', 'strutture','dati']`
- `words.reverse() → ['dati','strutture','e','algoritmi']`
- `words.remove('strutture') → ['dati','e','algoritmi']`

## Dizionari o Hash Tables, Associative Arrays, Lookup Tables

- `dictionary = {'indefatigable':'untiring',  
                  'intrepid':'fearlessness','dissemble':'simulate'}`
- `constants = {'pi':3.1415, 'e':2.7182, 'phi':1.6180}`
- `com_dict = {1:[1,2,3],2:[1,0,3],3:[0,4,5]}`

## Operazioni su dizionari

- `com_dict.keys()` → `[1,2,3]`
- `com_dict.values()` → `[[1, 2, 3], [1, 0, 3], [0, 4, 5]]`

### Accedere agli elementi

- `com_dict[3]` → `[0,4,5]`
- `constants['phi']` → `1.6180`

## If, While, and For

### If...

```
if condition:
    statements
elif condition:
    statements
else condition:
    statements
```

**Occhio:** Python indenta i blocchi!

### While...

```
while condition:
    statements
```

### For...

```
for var in sequence:
    statements
```

## Esempi

### If...

```
if (x > 0):
    print("Positivo")
elif (x < 0):
    print("Negativo")
else:
    print("Zero")
```

### While...

```
while (1):
    print("Sempre vero")
```

### For...

```
for i in range(5):
    print(i)
```

## Numpy

- Numpy è la libreria principale per calcolo scientifico in Python
- Gestisce efficientemente array multidimensionali
- All'inizio del codice:

```
import numpy as np
```

- Array:

- ▶ Un array numpy è una matrice di valori (tutti dello stesso tipo)
- ▶ Indicizzati con una tupla di valori non negativi
- ▶ Numero di dimensioni: **rank** dell'array
- ▶ **Shape**: tupla con le dimensioni

- Tutorial numpy:

```
docs.scipy.org/doc/numpy-dev/user/quickstart.html
```

## Array

```
import numpy as np
a = np.array([1, 2, 3])          # Crea array con rank 1 (vettore)
print(type(a))                  # => "<type 'numpy.ndarray'>"
print(a.shape)                  # => "(3,)"

print(a[0], a[1], a[2]) #      => "1 2 3"
a[0] = 5 # Cambia un elemento dell'array
print(a) #                      => "[5, 2, 3]"

b = np.array([[1,2,3],[4,5,6]]) # Crea array di rank 2
print(b.shape)                  #      => "(2,3)"
print(b[0,0], b[0,1], b[1,0])  #      => "1 2 4"
```

**Nota:** se si usa python 3 cambiare

```
print a          con          print (a)
```

## Creazione array

```
import numpy as np
a = np.zeros((2,2))             # Crea array di zero
print(a)                        #      => "[[ 0.  0.]
#                               [ 0.  0.]]"

b = np.ones((1,2))             # Crea array con 1
print(b)                        #      => "[[ 1.  1.]]"

c = np.full((2,2), 7)          # Crea array costante
print(c)                        #      => "[[ 7.  7.]
#                               [ 7.  7.]]"

d = np.eye(2)                   # Crea matrice identita 2x2
print(d)                        #      => "[[ 1.  0.]
# [                               0.  1.]]"

e = np.random.random((2,2))    # Array con valori casuali
print(e)
```

## Indicizzazione array

```
import numpy as np
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])# Crea array
# [[ 1  2  3  4]
#  [ 5  6  7  8]
#  [ 9 10 11 12]]

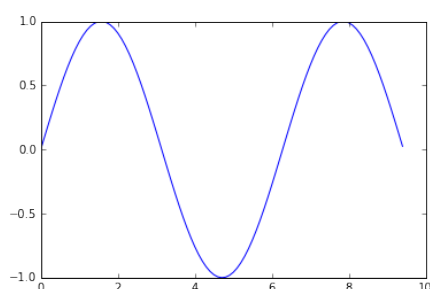
b = a[:2, 1:3]# Usa slicing per avere il sottoarray con le prim
# e colonne 1 e 2 ottiene b(shape (2, 2)):
# [[2 3]
#  [6 7]]

# Uno slice di un array e' una sua vista
print a[0, 1] # => "2"
b[0, 0] = 77 # b[0, 0] stessi dati di a[0, 1]
print a[0, 1] # => "77"
```

## Plot in matplotlib

```
import numpy as np
import matplotlib.pyplot as plt

# Calcola le coordinate x e y per punti della funzione seno
x = np.arange(0, 3 * np.pi, 0.1)
y = np.sin(x)
# Disegna i punti
plt.plot(x, y)
plt.show() # Visualizza il plot
```



```
import numpy as np
import matplotlib.pyplot as plt

# Seno e coseno
x = np.arange(0, 3 * np.pi, 0.1)
y_sin = np.sin(x)
y_cos = np.cos(x)

# usiamo matplotlib
plt.plot(x, y_sin)
plt.plot(x, y_cos)
plt.xlabel('x axis label')
plt.ylabel('y axis label')
plt.title('Sine and Cosine')
plt.legend(['Sine', 'Cosine'])
plt.show()
```

