Cosa è Python?

Linguaggio

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

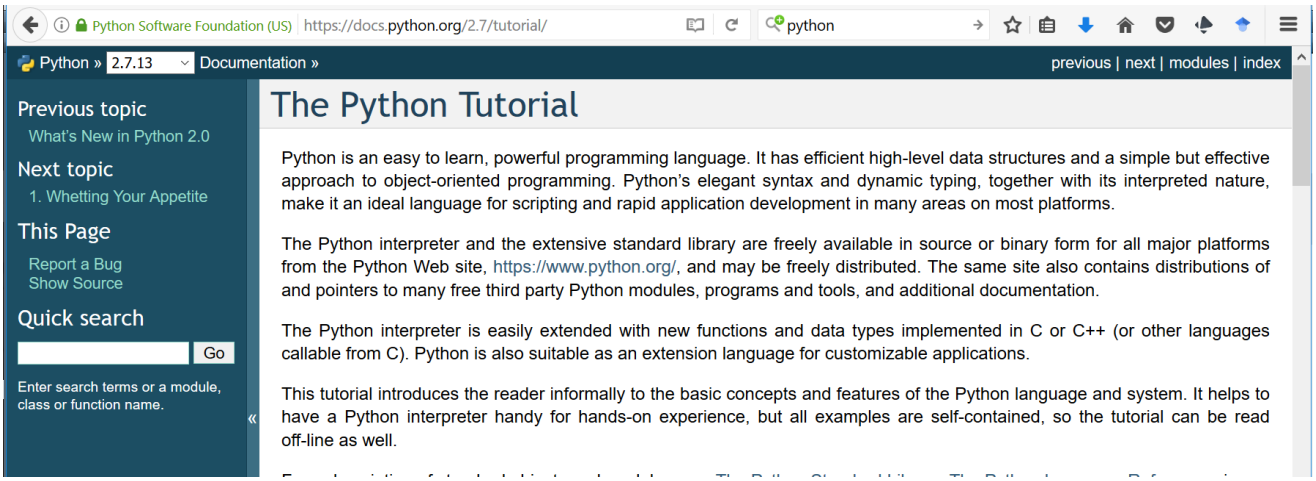
https://docs.python.org/3/faq/general.html

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?!?



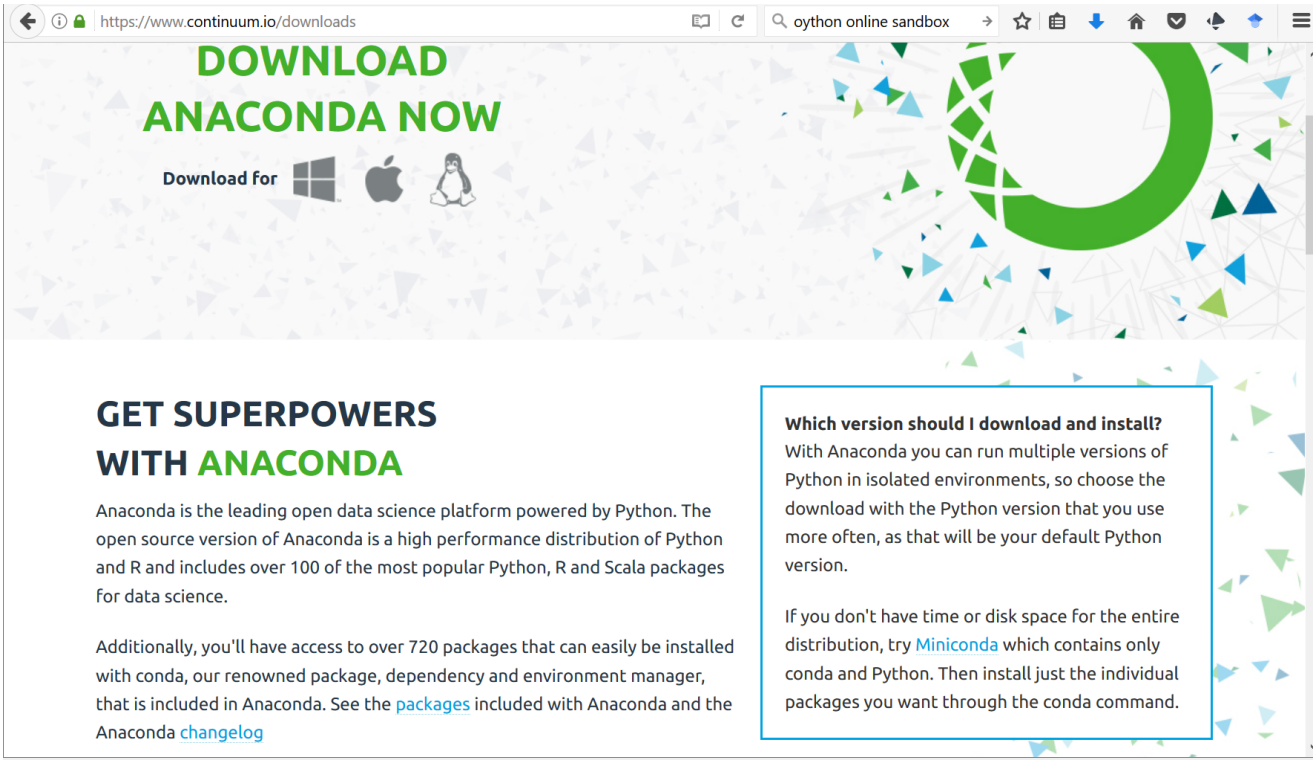
- https://docs.python.org/3.7/tutorial/
- 2.* vs 3.*

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)
- 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

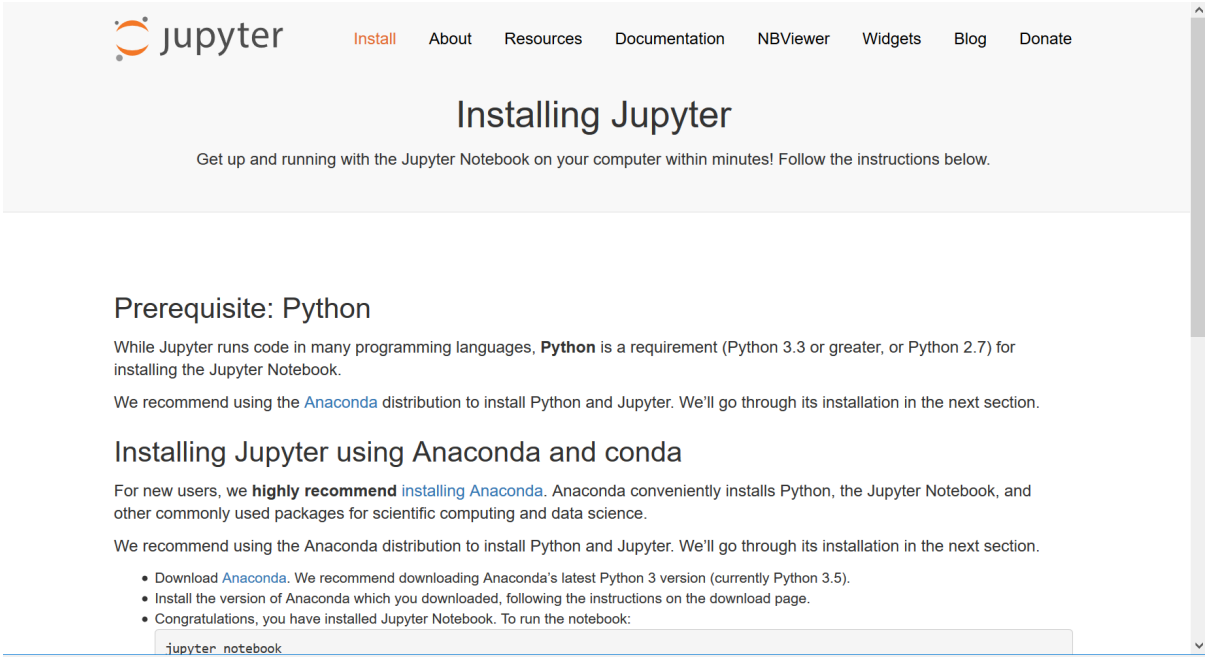
```
>>>print("Hello, World!")
Hello, World!
>>>var = 9+2
>>>var*11
121
```

Jupyter Notebook

- Ambiente computazionale interattivo basato sul Web per creare notebook IPython
- Un notebook IPython è un documento JSON contenente una lista ordinata di celle di input/output che possono contenere codice, testo, math, plot e altri media
- I notebooks possono essere convertiti in vari formati standard di uscita (HTML, presentation slides, LaTeX, PDF, ReStructuredText, Markdown, Python)
  - 'Download As' in the web interface
  - nbconvert in a shell

    `jupyther nbconvert Test.ipynb -to latex`

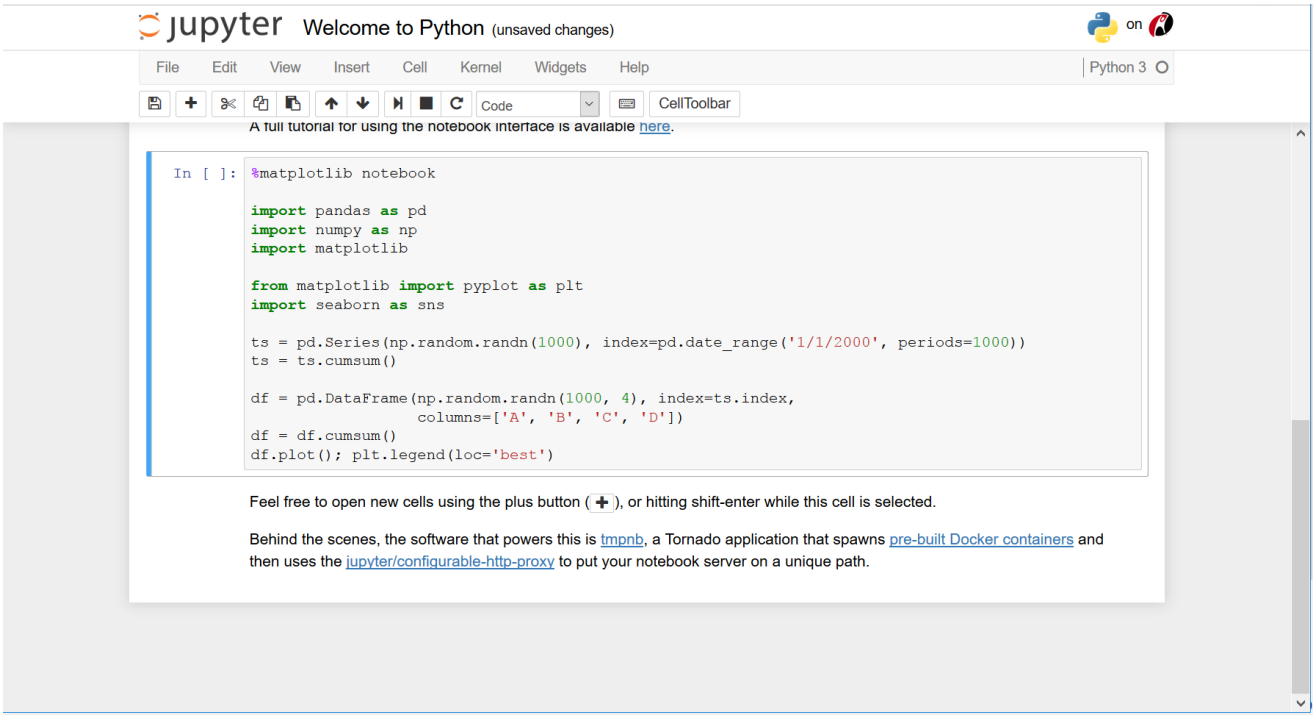
Istallare Jupyter



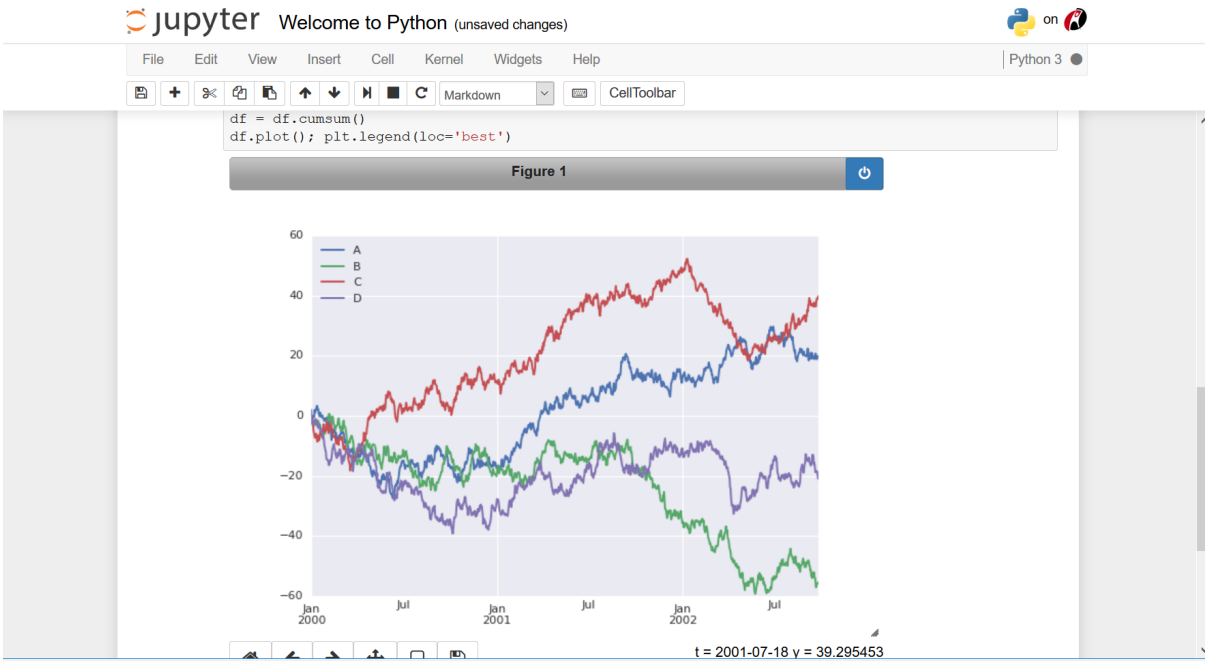
Incluso in Anaconda

poi: `jupyter notebook`

Jupyter



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" $\rightarrow$ "hello world"
- Ripetizione: "hello "*3 $\rightarrow$ "hello hello hello "
- Indicizzamento: "world"[3] $\rightarrow$ "l"
  - Nota: liste python sono zero-offset
- Cercare: "o" in "hello" $\rightarrow$ 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') $\rightarrow$ ['algoritmi','strutture','dati']
- words.insert(1,'e')$\rightarrow$ ['algoritmi','e', 'strutture','dati']
- words.reverse() $\rightarrow$ ['dati','strutture','e','algoritmi']
- words.remove('strutture') $\rightarrow$ ['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)
```

Interi

```
x = 3
print type(x)        # =>  <type 'int'>

print(c)              # =>  3
print(x + 1)          # =>  4
print(x - 1)          # =>  2
print(x * 2)          # =>  6
print(x ** 2)         # =>  9
x += 1
print(x)              # =>  4
x *= 2
print(x)              # =>  8
y = 2.5
print(type(y))        # =>  <type 'float'>"
print(y, y + 1, y * 2, y ** 2) # =>  2.5 3.5 5.0 6.25
```


Booleani

```
t = True
f = False
print(type(t))     # =>  <type 'bool'>

print(t and f)     # AND  => False
print(t or f)      # OR   => True

print(not t)       # NOT  => False
print(t != f)      # XOR  => True
```

Stringhe

```
hello = 'hello'     # Due modi per le stringhe
world = "world"     # singole o doppie virgolette
print(hello)         #   =>   hello
print(len(hello))    #   =>   5

hw = hello + ' ' + world #  concatenazione
print(hw)            #   =>  hello world

# formattazione tipo sprintf
hw1 = '%s %s %d' % (hello, world, 1)
print(hw1)           #   =>  hello world 1
```


Stringhe: metodi

```
s = "hello"
print(s.capitalize())     #   =>   Hello
print(s.upper())          #   =>   HELLO

print(s.rjust(7))         # giustifica a dx  => "  HELLO"
print(s.center(7))        # centra           => " HELLO "

print(s.replace('l','el')) # =>   heellelo
print(' world '.strip())  #   =>  world
```

Liste

Una lista in Python è un array ridimensionabile che può contenere elementi di tipo diverso

```
xs = [3, 1, 2]          # Crea una lista
print(xs, xs[2])         # => [3, 1, 2] 2
print(xs[-1])            # Indici <0 da fine lista => 2

xs[2] = 'abc'           # Liste con tipi diversi
print(xs)                # =>  [3, 1, 'abc']

xs.append('def')        # aggiunge in fondo
print(xs)                #  => [3, 1, 'abc', 'def']

x = xs.pop()            # Rimuove e restituisce l'ultimo elemento
print(x, xs)             #  => def [3, 1, 'abc']
```


Slicing

Permette di accedere a sottoliste

```
nums = range(5)      # crea una lista
print(nums)           #  => [0, 1, 2, 3, 4]
print(nums[2:4])      # da indice 2 a 4 (escluso) => [2,3]

print(nums[2:])       # da 2 alla fine => [2, 3, 4]
print(nums[:2])       # dall'inizio a 2 (escluso) => [0, 1]

print(nums[:])        # Tutta la lista => [0, 1, 2, 3, 4]
print(nums[:-1])      # indici negativi =>  [0, 1, 2, 3]

nums[2:4] = [8, 9]   # nuova sottolista
print(nums)           # => [0, 1, 8, 9, 4]
```

Cicli su liste

```
animals = ['cat', 'dog', 'monkey']
for animal in animals:
    print(animal)
# => "cat", "dog", "monkey" (su linee separate)
```

```
animals = ['cat', 'dog', 'monkey']
for idx, animal in enumerate(animals):
    print('#%d: %s' % (idx + 1, animal))
# => "#1: cat", "#2: dog", "#3: monkey", (su linee separate)
```


Dizionari

Memorizza coppie (chiave, valore)

```
d = {'cat': 'cute', 'dog': 'furry'} # Nuovo dizionario
print(d['cat'])                      # => "cute"
print('cat' in d)                    # => True

d['fish'] = 'wet'                   # Aggiunge voce
print(d['fish'])                     # => "wet"

print(d['monkey'])                   # => KeyError: ...
print(d.get('monkey', 'N/A'))        # Default  => "N/A"
print(d.get('fish', 'N/A'))          # => "wet"

del d['fish']                       # Rimuove un elemento
print(d.get('fish', 'N/A'))          # ora manca => "N/A"
```

Cicli con dizionari

```
d = {'person': 2, 'cat': 4, 'spider': 8}
for animal in d:
    legs = d[animal]
    print('A %s has %d legs' % (animal, legs))


=>
# "A person has 2 legs"
"A spider has 8 legs"
"A cat has 4 legs"
```


Insiemi

```
animals = {'cat', 'dog'}
print('cat' in animals)      # =>  "True"
print('fish' in animals)     # =>  "False"

animals.add('fish')          # Aggiunge un elemento
print('fish' in animals)     # =>  "True"
print(len(animals))          # =>  "3"

animals.add('cat')           # Aggiunge un elemento presente
print(len(animals))          # =>  "3"

animals.remove('cat')        # Rimuove an elemento
print(len(animals))          # => "2"
```

Tupla

Lista ordinata di valori immutabile

```
# Crea un dizionario con tuple
d = {(x, x + 1): x for x in range(10)}
print(type(d))                                  # => <type 'dict'>

t = (5, 6)                                     # Crea una tupla
print(type(t))                                  # => <type 'tuple'>

print(d[t])                                     # => 5 (posizione)
print(d[(1, 2)])                                # => 1
```


Funzioni

```
def sign(x):
  if x > 0:
    return 'positive'
  elif x < 0:
    return 'negative'
  else:
    return 'zero'

for x in [-1, 0, 1]:
  print(sign(x))

# => "negative", "zero", "positive"
```

Funzioni con argomenti

```
def hello(name, loud=False):
  if loud:
     print('HELLO, %s!') % name.upper()
  else:
     print('Hello, %s') % name


hello('Bob')                        # =>   "Hello, Bob"
hello('Fred', loud=True)            # =>   "HELLO, FRED!"
```


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

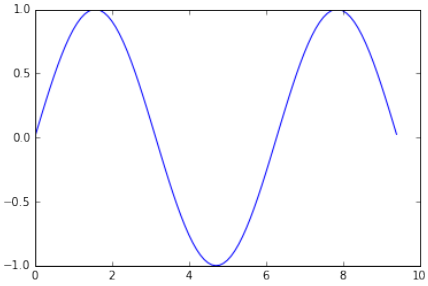
```
import numpy as np
a = np.zeros((2,2))      # Crea array di zero
print(a)                  #    =>  "[[ 0. 0.]
#          [ 0. 0.]]"

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)
```

Indicizzamento 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()
```