

# Business Intelligence per i servizi finanziari

---

## INTRODUZIONE A PYTHON - 1

Silvio Bencini

[silvio.bencini@unimib.it](mailto:silvio.bencini@unimib.it)  
[antonio.candelieri@unimib.it](mailto:antonio.candelieri@unimib.it)

# Perché questo corso?

---

- Perché la finanza si propone di rispondere a domande centrali nel mondo del lavoro ma presenti anche nelle scelte individuali quali ad esempio:
  - il rendimento di un investimento;
  - La pensione che ci si può attendere da un certo piano di accumulo;
  - La differenza fra diversi possibili investimenti;
  - La combinazione migliore di investimenti
- Gli strumenti della «finanza» servono anche a un'azienda per decidere l'investimento in un impianto, ma in questo corso ci limitiamo agli investimenti in strumenti finanziari liquidi (vedi tavola successiva), cioè scambiati in mercati che assicurino prezzi con sufficiente frequenza
- Perché la finanza si basa sempre di più su modelli quantitativi che si prestano ad essere automatizzati ed elaborati con calcolatori
- Perché la finanza produce una quantità enorme di dati che gli strumenti messi a disposizione dalla «data science» sono adatti ad elaborare
- Perché capire il contesto nel quale si lavora è fondamentale

# Cosa sono gli «strumenti finanziari»

---

Gli strumenti finanziari sono elencati nel Testo Unico della Finanza all'articolo 1, comma 2. Essi sono:


- le azioni e gli altri titoli rappresentativi di capitale di rischio negoziabili sul mercato dei capitali;
- le obbligazioni, i titoli di Stato e gli altri titoli di debito negoziabili sul mercato dei capitali;
- le quote di fondi comuni di investimento;
- i titoli normalmente negoziati sul mercato monetario;
- qualsiasi altro titolo normalmente negoziato che permetta di acquisire gli strumenti precedentemente indicati;
- i contratti futures su strumenti finanziari, su tassi di interesse, su valute, su merci e sui relativi indici;
- i contratti di scambio a pronti e a termine (swaps) su tassi di interesse, su valute, su merci nonché su indici azionari (equity swaps);
- i contratti a termine collegati a strumenti finanziari, a tassi d'interesse, a valute, a merci e ai relativi indici;
- i contratti di opzione per acquistare o vendere gli strumenti indicati nelle precedenti lettere e i relativi indici, nonché i contratti di opzione su valute, su tassi d'interesse, su merci e sui relativi indici;
- le combinazioni di contratti o di titoli indicati precedentemente.

I mezzi di pagamento non sono considerati strumenti finanziari.

# Due parole sul docente

linkedin.com/in/silvio-bencini-4b19a815/

in Search Home My Network 6 Jobs Messaging



**Silvio Bencini**  
Managing Partner - European Investment Consulting  
Milan, Lombardy, Italy · [Contact info](#)  
500+ connections

[Open to](#) [Add section](#) [More](#)

Show recruiters you're open to work — you control who sees this. [Get started](#) X

Find potential clients by showcasing the services you provide. [Get started](#) X

European Investment Consulting  
University of Turin

in Search Home My Network 6 Jobs Messaging

**Silvio Bencini**  
Managing Partner - European Investment Consulting

**Experience** +

**Managing Partner**  
European Investment Consulting  
Jan 2012 – Present · 9 yrs 11 mos  
Advisor to major Italian Pension Funds on asset allocation, manager selection and portfolio monitoring issues  
[www.strategiedinamiche.it](http://www.strategiedinamiche.it)

**Founder**  
Strategie dinamiche  
Jan 2010 – Present · 11 yrs 11 mos  
Milan Area, Italy  
This web site allows the user to simulate the behavior of several dynamic and option replicating strategies  
<https://www.strategiedinamiche.it>

**Senior Advisor**  
Accenture  
Feb 2010 – Jan 2012 · 2 yrs  
Senior Advisor in the Management Consulting Practice focusing on Private Banking and Wealth Management

**Adjunct professor - Decision support systems in finance**  
Università degli Studi di Milano-Bicocca  
Jan 2011 – Jun 2011 · 6 mos  
Milan

**Head of Private Banking**  
Ersel  
Dec 2006 – Nov 2009 · 3 yrs

Show 5 more experiences

**Education** +

**University of Turin**  
Master of Science (M.Sc.), Economics and Business Administration  
1974 – 1982

# Il programma

---

- Breve introduzione (ripasso) di Python
- I valori finanziari nel tempo – Tassi d'interesse, valori attuali, rendite
- Opzioni – Payoffs e calcolo del valore delle opzioni
- Come scaricare i dati dal web
- Proprietà dei rendimenti finanziari
- Il modello di mercato (CAPM)
- Portafogli – Rendimento e rischio
- Portafogli – Frontiera efficiente
- Serie temporali – Modelli lineari
- Serie temporali – Previsioni 1
- Serie temporali – Modelli non lineari (NN & C)
- Serie temporali – Previsioni 2

tag

Talent Garden

COWORKING SCHOOL CORPORATES EVENT SPACES AGENDA GUIDES ABOUT BLOG

Home / Coding / 7 Most Popular Programming Languages in 2021

# 7 Most Popular Programming Languages in 2021

CODING

SHARE: f t in

THE 7 MOST POPULAR CODING LANGUAGES IN 2021

Talent Garden Innovation School

2021

#CODING

As the technology ecosystem evolves the same happens for what lies at the core of each computer program, application, device or system, namely programming languages. As new demands emerge, new programming languages designed for or that specifically fit those demands also come to the fore. There are various programming language rankings (PYPL, TIOBE, Stackoverflow, etc) that are updated every year and that take into account many factors, including diffusion, easiness, generality vs specificity, and test but not least how useful a certain programming language is to get a job or a salary increase! And the tops of the pops for the year 2021 are...

1. Javascript

Javascript is one of the most popular programming languages among developers and has been for many years, thanks to its wide range of applications, its flexibility, and its capability of adding responsive elements to web pages. Javascript is surely an essential asset of the know-how of any coder, now and for the years to come. If you feel you're not as proficient in Javascript as you should be for your career's sake, you can enroll at the [Codemaster Online Bootcamp](#), a crash course that will bring you from 0 to Javascript master in just 12 weeks! All the tech giants (Facebook, Google, Microsoft, etc) use Javascript for their apps, and so do millions of websites around the world. So if you are choosing a language to study in 2021, this is the one!

2. Python

The widespread diffusion of Python, which is undoubtedly one of the languages most used, is due to its ease. Python isn't a complex, complicated language. On the contrary it's quite easy to learn. Its flexibility means you can do almost anything and you have an enormous number of libraries that come with it. Another factor to take into account is that Python can be successfully used for trending technologies like artificial intelligence, machine learning and data analysis. And of course, it can be found everywhere: more than 80% of all the websites are built using Python. Developers, do add Python to your CV!

3. C/C++

C and C++ may be old but they are by no means outdated! They are still used in many companies and systems (C ranks number 1 in the 2021 TIOBE index, and C++ number 4). One of the major advantages of learning them is that many other programming languages are based on or inspired by C and/or C++. So if you learn them it's really easier to learn other languages as well. In comparison to other programming languages both C and C++ are low level languages and so closer to hardware and particularly good at programming hardware resources. But they can also be used for games development, GUI, operating systems...they really are languages for all seasons (and uses).

### Growth of major programming languages

Based on Stack Overflow question views in World Bank high-income countries

Language	2012	2014	2016	2018
python	4.0%	5.5%	7.5%	8.8%
javascript	7.5%	8.5%	8.5%	8.5%
java	8.5%	8.5%	8.5%	8.5%
c#	4.0%	5.5%	7.0%	7.0%
php	4.0%	4.0%	3.5%	3.5%
c++	4.0%	3.5%	3.0%	3.0%

Business Intelligence per i servizi finanziari – 2021-2022 – A. Candelieri, S.Bencini – pag. 6

## Perché Python? - 2

Worldwide, Oct 2021 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Python	29.66 %	-2.1 %
2		Java	17.18 %	+0.8 %
3		JavaScript	8.81 %	+0.4 %
4		C#	7.3 %	+1.1 %
5	↑	C/C++	6.48 %	+0.7 %
6	↓	PHP	5.92 %	+0.1 %
7		R	4.09 %	+0.2 %
8		Objective-C	2.24 %	-1.2 %
9	↑	TypeScript	1.91 %	+0.1 %
10	↑↑	Kotlin	1.9 %	+0.3 %
11	↓↓	Swift	1.86 %	-0.6 %
12	↓	Matlab	1.58 %	-0.2 %

<https://pypl.github.io/PYPL.html>

### Is Python used in the real world?

Let's have a look at some of its most popular applications:

- **Google Search** relies on Python and TensorFlow to create machine learning models to increase search relevance.
- **Gmail** uses Python to create predictive email filtering models.
- **Spotify** uses over 6000 Python processes to work together in their back-end music recommendation engine.
- **Quora** uses Python because of its simplicity and ease of use to analyze statistics about trends in Q&As.
- **Netflix** uses Python bots called "monkeys" to track and alert any changes in EC2 security-related policies.
- **Dropbox** even managed to recruit Python's creator, Guido van Rossum. They use it heavily!
- **Reddit** serves more than 500,000,000 users with a Python backend. Impressive!
- **Instagram** currently features one of the world's largest deployment of the Django web framework written in Python.
- **Uber** uses Python as one of two main programming languages for its services. (The other one is Node.js.)

Want to learn the language of the 21st century? 🚀

<https://www.finxter.com>

# Che cos'è Python?

- Un linguaggio di programmazione molto intuitivo, facile da imparare, col quale si possono fare cose complesse
- Orientato agli oggetti, ma supporta anche gli altri metodi di programmazione
- Open source
- È circondato da una vasta famiglia di librerie per calcoli matriciali, statistici, e di «machine learning»
- Capacità grafiche
- E' un linguaggio interpretato, non compilato, perciò è relativamente lento ma...
- È un linguaggio «colla» («glue») capace di integrarsi facilmente con C e Java
- Ciò consente di avere librerie scritte in C che sono veloci e superano la lentezza originale (es. Numpy per le matrici)
- Sostenuto da una grande comunità di utenti

<https://www.python.org>

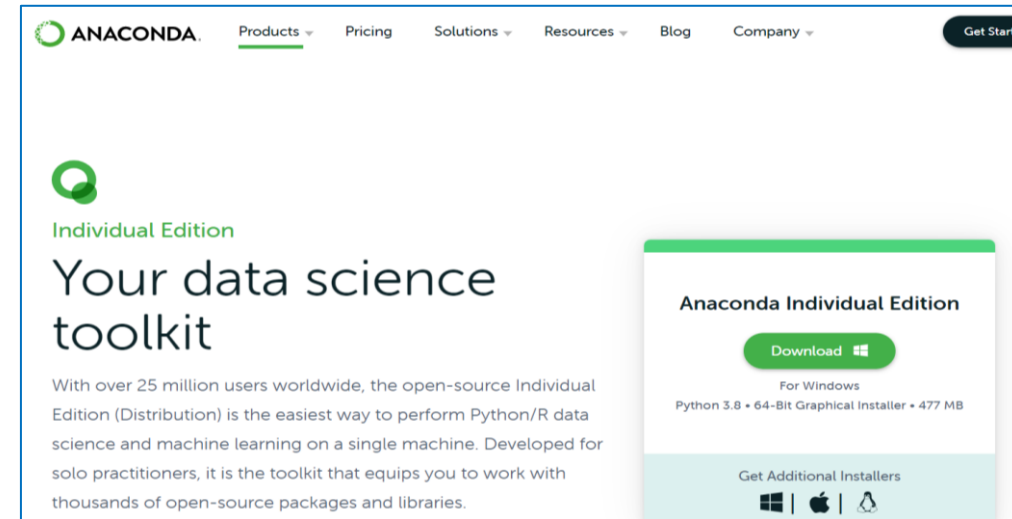
Comparison Factors	Java	Python
Speed	✓	✗
Legacy	✗	✓
Code	✗	✓
Practical Agility	✓	✓
Trends	✗	✓
Salary	✗	✓
Syntax	✓	✓

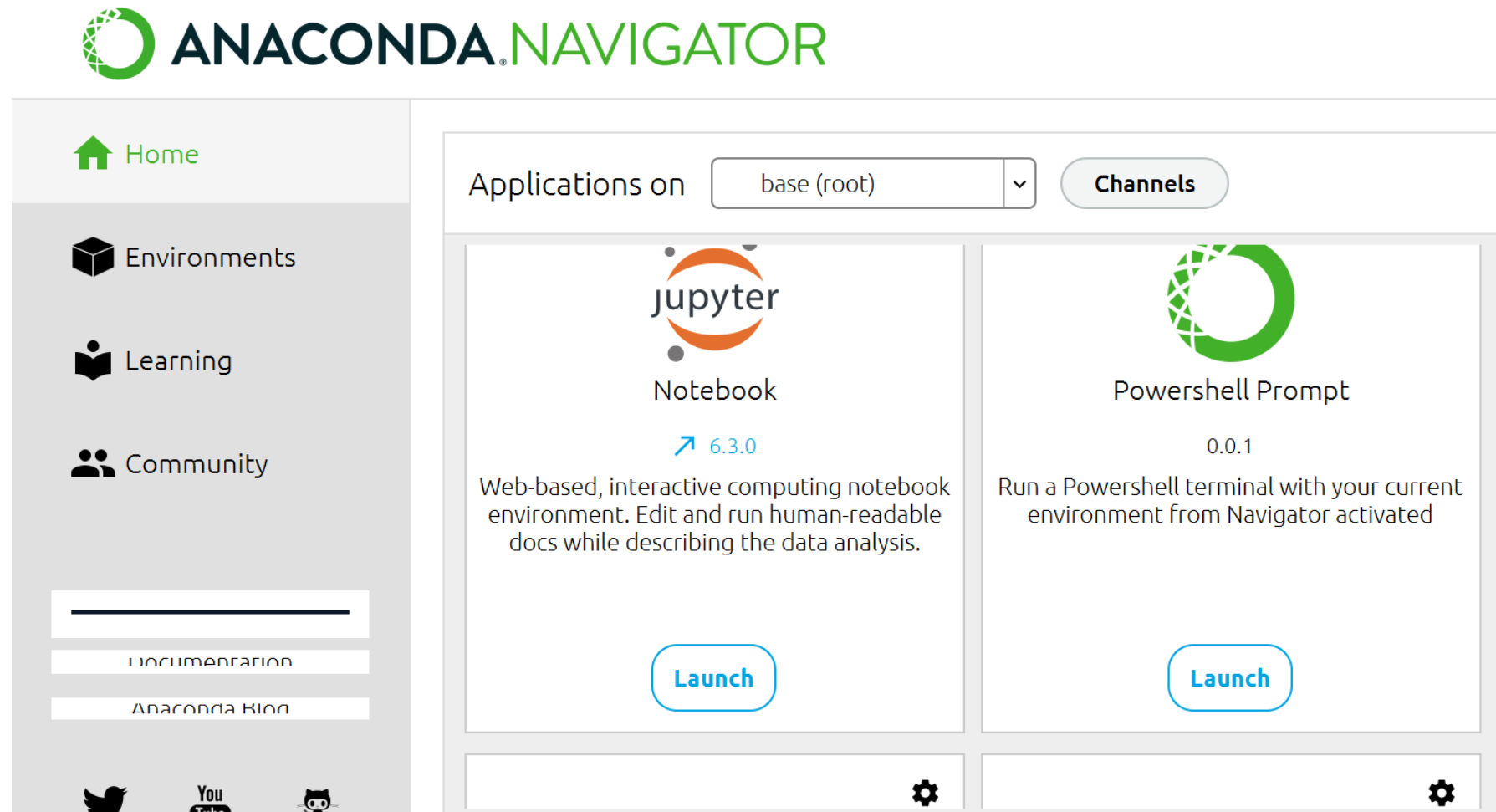
<https://www.edureka.co/blog/java-vs-python/>



# Installare Python

- 1) Utilizzare l'installatore che si trova sul sito di Python (Windows/MacOS/Linux)
  - <https://www.python.org/downloads/> (scegliere l'installatore per Python 3.8)
  - Per installare librerie aggiuntive utilizzare **Pip** («**Python Packager Installer**»)
    - <https://pypi.org/project/pip/>
    - Nella cmd line digitare «pip install package\_name»
- 2) Utilizza ANACONDA sul sito
  - <https://www.anaconda.com/download/> (scegliere l'installatore per Python 3.10)
  - Per installare nuove librerie con Anaconda utilizzare:
    - Conda con il comando «Conda install package\_name»
    - Pip con il comando «pip install package\_name»
  - Anaconda è un ambiente nato per il Data Science. Anaconda semplifica il processo di configurazione di un ambiente di sviluppo in Python, perché include tutto ciò di cui si ha bisogno per programmare. Il package manager **Conda**, incluso in Anaconda, comprende circa 300 pacchetti in ambito DataScience pronti all'uso come Pandas, NLTK, Numpy, Matplotlib, Jupiter, Requests, tensorflow ed altri.





# Lavorare in Python – Jupiter Notebook - 2

localhost:8888/notebooks/Dropbox/Corso\_Bicocca/Introduzione%20a%20Python.ipynb

jupyter Introduzione a Python Last Checkpoint: Ieri alle 12:38 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

Run

### Assegnazione di variabili e nomi

```
In [1]: x = 3
        x

Out[1]: 3
```

```
In [2]: x = x + 1
        x

Out[2]: 4
```

```
In [3]: z = "hello BISF"
        z

Out[3]: 'hello BISF'
```

```
In [4]: y = a

-----
NameError                                Traceback (most recent call last)
<ipython-input-4-3afc6a63a546> in <module>
```

## Lavorare in Python –Dove trovare aiuto – 4.1


```
In [9]: ?np.sqrt
```

```
Call signature: np.sqrt(*args, **kwargs)
Type:          ufunc
String form:    <ufunc 'sqrt'>
File:          c:\users\s.bencini\anaconda3\lib\site-packages\numpy\__init__.py
Docstring:
sqrt(x, /, out=None, *, where=True, casting='same_kind', order='K', dtype=None, subok=True[, signature, extobj])

Return the non-negative square-root of an array, element-wise.

Parameters
-----
x : array_like
    The values whose square-roots are required.
out : ndarray, None, or tuple of ndarray and None, optional
    A location into which the result is stored. If provided, it must have
    a shape that the inputs broadcast to. If not provided or None,
```

# Lavorare in Python –Dove trovare aiuto – 4.2

stackoverflow

[About](#)

[Products](#)

[For Teams](#)

[Google](#)

Home

PUBLIC

Questions

Tags

Users

COLLECTIVES

Explore Collectives

FIND A JOB

Jobs

Companies

TEAMS

Stack Overflow for

How do I calculate square root in Python?

Asked 9 years, 8 months ago · Active 12 days ago · Viewed 491k times

155

Why does Python give the "wrong" answer?

```
x = 16

sqrt = x**(.5) #returns 4
sqrt = x**(1/2) #returns 1
```

23

Yes, I know `import math` and use `sqrt`. But I'm looking for an answer to the above.

python

math

python-2.x


sqrt

Share

Improve this question


Follow

edited Jan 6 at 15:52

Andrew Mascillaro

587 ● 3 ● 17

asked Mar 7 '12 at 2:48

Merlin

20.9k ● 34 ● 113 ● 192

Business Intelligence per i servizi finanziari – 2021-2022 – A. Candelieri, S.Bencini – pag. 13



🔍 Search the docs ...

Array objects

Constants

Universal functions ( **ufunc** )

**Routines**

Array creation routines

Array manipulation routines

Binary operations

String operations

C-Types Foreign Function

Interface ( **numpy.ctypeslib** )

Datetime Support Functions

Data type routines

Optionally SciPy-accelerated  
routines ( **numpy.dual** )

## numpy.sqrt

```
numpy.sqrt(x, /, out=None, *, where=True, casting='same_kind', order='K',  
dtype=None, subok=True[, signature, extobj]) = <ufunc 'sqrt'>
```

Return the non-negative square-root of an array, element-wise.

Parameters: **x** : *array\_like*

The values whose square-roots are required.







**out** : *ndarray, None, or tuple of ndarray and None, optional*

A location into which the result is stored. If provided, it must have a shape that the inputs broadcast to. If not provided or None, a freshly-allocated array is returned. A tuple (possible only as a keyword argument) must have length equal to the number of outputs.

**where** : *array\_like, optional*

This condition is broadcast over the input. At locations where the condition is True, the *out* array will be set to the ufunc result.

## Lavorare in Python –Dove trovare aiuto – 4.4

<div> <div>finxter</div> <div>The Ultimate Python Cheat Sheet</div> <div>  </div> </div>		
Keywords		
Keyword	Description	Code Examples
False, True	Boolean data type	<pre>False == (1 &gt; 2) True == (2 &gt; 1)</pre> 
and, or, not	Logical operators → Both are true → Either is true → Flips Boolean	<pre>True and True    # True True or False    # True not False        # True</pre>
break	Ends loop prematurely	<pre>while True:     break # finite loop</pre>
continue	Finishes current loop iteration	<pre>while True:     continue     print("42") # dead code</pre>
class	Defines new class	<pre>class Coffee:     # Define your class</pre>
def	Defines a new function or class method.	<pre>def say_hi():     print('hi')</pre>
if, elif, else	Conditional execution: - "if" condition == True? - "elif" condition == True? - Fallback: else branch	<pre>x = int(input("ur val:")) if x &gt; 3: print("Big") elif x == 3: print("3") else:     print("Small")</pre>
for, while	# For loop for i in [0,1,2]: print(i)	<pre># While loop does same j = 0 while j &lt; 3:     print(j); j = j + 1</pre> 
in	Sequence membership	<pre>42 in [2, 39, 42] # True</pre>
is	Same object memory location	<pre>y = x = 3 x is y    # True [3] is [3] # False</pre>
None	Empty value constant	<pre>print() is None # True</pre>
lambda	Anonymous function	<pre>(lambda x: x+3) (3) # 6</pre> 
return	Terminates function. Optional return value defines function result.	<pre>def increment(x):     return x + 1 increment(4) # returns 5</pre>
Basic Data Structures		
Type	Description	Code Examples
Boolean	The Boolean data type is either <b>True</b> or <b>False</b> . Boolean operators are ordered by priority: <b>not</b> → <b>and</b> → <b>or</b>	<pre>## Evaluates to True: 1&lt;2 and 0&lt;=1 and 3&gt;2 and 2&gt;=2 and 1==1 and 1!=0  ## Evaluates to False: bool(None or 0 or 0.0 or '' or [] or {} or set())</pre> <p>Rule: None, 0, 0.0, empty strings, or empty container types evaluate to False</p> <div> <div> <pre>{ } → </pre>  </div> <div> <pre>{ 1, 2, 3 } → </pre>  </div> </div>
Integer, Float	An integer is a positive or negative number without decimal point such as 3.  A float is a positive or negative number with floating point precision such as 3.1415926.	<pre>## Arithmetic Operations x, y = 3, 2 print(x + y) # = 5 print(x - y) # = 1 print(x * y) # = 6 print(x / y) # = 1.5 print(x // y) # = 1 print(x % y) # = 1 print(-x)    # = -3 print(abs(-x)) # = 3 print(int(3.9)) # = 3 print(float(3)) # = 3.0 print(x ** y) # = 9</pre> <p>Integer division rounds toward the smaller integer (example: 3//2==1).</p>
String	Python Strings are sequences of characters.	<pre>## Indexing and Slicing s = "The youngest pope was 11 years" s[0] # 'T' s[1:3] # 'he' s[-3:-1] # 'ar' s[-3:] # 'ars'  s.split() x[-2] + " " + x[2] + " " + "11 popes"  ## String Methods y = " Hello world!\n " y.strip() # Remove Whitespaces "Hi".lower() # Lowercase: 'hi' "Hi".upper() # Uppercase: 'HI' "hello".startswith("he") # True "hello".endswith("lo") # True "hello".find("ll") # Match at 2 "cheat".replace("ch", "m") # 'meat' ''.join(["F", "B", "I"]) # 'FBI' len("hello world") # Length: 15 "ear" in "earth" # True</pre> <p>String Creation Methods: 1. Single quotes &gt;&gt;&gt; 'Yes' 2. Double quotes &gt;&gt;&gt; "Yes" 3. Triple quotes (multi-line) &gt;&gt;&gt; """Yes     We Can""" 4. String method &gt;&gt;&gt; str(5) == '5' True 5. Concatenation &gt;&gt;&gt; "Ma" + "hatma" 'Mahatma'</p> <p>Whitespace chars: Newline \n, Space \s, Tab \t</p> <div> <div>Slice [::2]</div> <div> <div>1</div> <div>2</div> <div>3</div> <div>4</div> </div> <div>0 1 2 3</div> </div>

<https://blog.finxter.com/collection-10-best-numpy-cheat-sheets-every-python-coder-must-own/>

Il progetto di laboratorio in Python consiste nel progettare e realizzare sistema di supporto alle decisioni («decision support system») che dovrà avere due gruppi di funzionalità:

- **Acquisizione, analisi descrittiva e visualizzazione dei dati**
  - Scaricare e salvare serie di prezzi di singoli titoli o portafogli
  - Calcolare e rappresentare graficamente i rendimenti e commentarne il comportamento
  - Presentare grafici diagnostici (istogrammi, kernel density plots, boxplots e qq-plots) dei rendimenti e commentare
  - Calcolare statistiche descrittive e rispondere a domande collegate
- **Analisi e modellazione di serie temporali**
  - Analisi e modellazione di serie temporali
  - Correlazioni
  - Tecniche di classificazione
  - Previsioni con modelli lineari e non



# Laboratorio in Python – Dove trovare i dati

HomeMailNewsFinanceSportsEntertainmentSearchMobileMore...

yahoo!

finance

Search for news, symbols or companies

Q

Sign in

Mail

Finance Home

Watchlists

My Portfolio

Screeners

Yahoo Finance Plus

Markets

News

Personal Finance

Cryptocurrencies

Videos

S&P 500

4,600.17

-5.21 (-0.11%)

Dow 30

35,853.32

+33.76 (+0.09%)

Nasdaq

15,509.96

+11.57 (+0.07%)

Russell 2000

2,338.61

+41.42 (+1.80%)

Crude Oil

84.75

+1.18 (+1.41%)

Gold

1,797.00

+13.10 (+0.73%)

U.S. markets close in 5 hours 16 minutes

<>

S&P 500 (^GSPC)

SNP - SNP Real Time Price. Currency in USD

☆ Add to watchlist

4,600.17

-5.21 (-0.11%)

As of 10:44AM EDT. Market open.

Summary

Chart

Conversations

Historical Data

Options

Components

Quote Lookup

Q

Advertisement

Business Intelligence per i servizi finanziari – 2021-2022 – A. Candelieri, S.Bencini – pag. 17

- **Working directory** – La cartella dove salvate i file relativi al progetto corrente
- **Package/Library** – Una collezione integrata di funzioni (per esempio Numpy, Pandas, Scipy ...)
- **Tipi di file in Python**
  - .spydata : file nel quale vengono salvate le variabili di Python quando si lavora con Spyder IDE
  - .py : file dove sono salvati i programmi di Python
  - .jpyb: file creati con Jupyter Notebook
  - .pickle : Il modulo **pickle** implementa un algoritmo per trasformare un oggetto arbitrario Python in un una serie di byte. Questo processo viene anche detto [serializzazione](#) dell'oggetto). Il flusso di byte che rappresenta l'oggetto può essere trasmesso o conservato, e successivamente ricostruito per creare un nuovo oggetto con le stesse caratteristiche.

- Le **variabili** in Python sono «**case-sensitive**», maiuscole e minuscole sono diverse
- Non sono consentiti caratteri speciali e spazi all'interno dei nomi delle variabili
- I nomi delle variabili non possono iniziare con un numero
- In Python non è necessario dichiarare le variabili (int, double, string) come si deve fare con altri linguaggi. Python capisce da se' il tipo di variabile con la quale ha a che fare
- Le variabili possono essere modificate e copiate in altre variabili

- **Scalare** – un singolo numero (integer o float)
- **Strutture di base**
  - **Tuple** - una collezione di oggetti arbitrari – Non è modificabile – Pochi metodi disponibili
  - **List** – una collezione di oggetti di tipo diverso (ad esempi un vettore, una matrici e un dataframe) – Modificabile – Molti metodi disponibili
  - **Dict** (Dictionary) – una serie di dati organizzati per chiave/valore
  - **Set** – un insieme di oggetti unici
- **Array** – un vettore riga o colonna di scalari (utilizzando Numpy)
- **Matrice** – Un insieme a due dimensioni (righe/colonne) di elementi tutti dello stesso tipo
- **Data frame** – un insieme a due dimensioni di elementi ma ciascuna colonna può contenere dati di tipo diverso (ad esempio: date, interi, reali, testo)
- **Categorical** – una classe di dati (per esempio interi o caratteri utilizzando Pandas)

- Attributi
  - Mode – Numerico, carattere, complesso, logico
  - Length – Numero di elementi in un oggetto (il conteggio comincia da 0)
- Creazione
  - Assegnando dei valori
  - Creando un oggetto vuoto
- Casi speciali
  - Valore indefinito «Not a Number» NaN
  - Valore mancante None/Null

- **Le funzioni**
  - servono a effettuare azioni sugli oggetti
  - hanno argomenti e opzioni, spesso di default
  - forniscono un risultato («return»)
  - Il nome della funzione è seguito da due parentesi chiuse, che possono contenere gli argomenti
  - Per sapere cosa fa una funzione basta digitare «?nomefunzione» (vedi «Come chiedere aiuto»)

- Un array (vettore) è una struttura di dati che può contenere dati dello stesso tipo. In un array si possono mettere solo numeri o caratteri/stringhe
- In Python gli «array» sono creati con la libreria Numpy (alias np) perciò per trasformare un vettore (lista o tupla) si usa il comando `np.array()`
- Se cerchiamo di trasformare in array una lista di numeri e stringhe Numpy trasforma tutto in stringhe
- Per aggiungere un elemento a un array si utilizza il metodo `np.append()`
- Gli array sono oggetti sui quali possono essere molte normali operazioni aritmetiche, applicate a ciascun elemento

(1) Un **array** (detto anche **vettore** o **matrice**) in [informatica](#), indica una [struttura dati](#) complessa, statica e omogenea. Gli array, presenti praticamente in tutti i [linguaggi di programmazione](#) o di scripting, sono ispirati alla nozione [matematica](#) di [vettore](#) (quando monodimensionali) o di [matrice](#) (nel caso di array bidimensionali).

- Assegnare un tipo a dei dati («coercion») - **.astype**
- Cambiare il tipo dei dati («conversion»)