



Python in pratica

Pascal Brunot

09/01/2026

FabLab Bergamo



FabLab – una rete internazionale



FabLab è un format che nasce negli Stati Uniti, al MIT di Boston

Nasce dall'idea di un professore universitario, che pensa che i suoi studenti siano molto bravi nella teoria, e molto carenti nella pratica.

Crea uno spazio in cui «mettere le mani in pasta», in cui realizzare oggetti fisici.

Ci sono in Italia e nel mondo molti FabLab, dove è possibile imparare la fabbricazione digitale in un ambiente collaborativo.

FabLab Bergamo APS

www.fablabbergamo.it

Fondato nel 2013



Associazione di Promozione Sociale
iscritta al RUNTS
ospitata nel Patronato S. Lorenzo

100% gestita da
volontari che
frequentano per hobby.

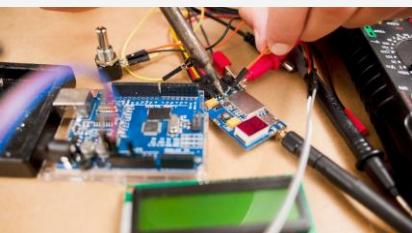
Introiti da
tesseramenti, CRE
estivo, donazioni, corsi

Tesseramento
30 EUR/anno

Associarsi su
FABL
BERGAMO



 ARDUINO
DAYS 2025



BGlug
Bergamo Linux Users Group



Regole del corso



ORARI

15,30 – 16,30
pausa 15 minuti
16.45 – 17.45

Le domande sono molto gradite, a qualunque momento. Alzate la mano se non vi noto.

Dovete usare il PC e fare gli esercizi/quizz man mano, se volete imparare qualcosa.

Obiettivi del corso

- ❖ I partecipanti saranno in grado di usare autonomamente librerie comuni di Python per compiere attività come:
 - ❖ Leggere e scrivere formati di file comuni
 - ❖ Testo, CSV, Excel, immagini...
 - ❖ Manipolazione di dati.
 - ❖ Python è un linguaggio molto diffuso per il Data Science.
 - ❖ Chiamare un API (interfaccia) di servizi online.
 - ❖ Python è un'ottima soluzione per «incollare» servizi diversi.
 - ❖ Automatizzare operazioni ripetitive
- ❖ E' un corso pratico. Voglio mostrarvi elementi pratici del linguaggio Python per fare cose concrete, non l'aspetto strettamente informatico. Tuttavia se avete domande in merito proverò a rispondervi !

Piano Lezione 1

- ❖ Introduzione docente
- ❖ Obiettivi del corso
- ❖ Python, storia del linguaggio
- ❖ L'ambiente di programmazione del corso
- ❖ Concetti elementari
 - ❖ Variabili
 - ❖ Tipi
 - ❖ Espressioni
 - ❖ Condizioni e cicli
 - ❖ Funzioni

Piano lezioni

Lezione 1 – intro del linguaggio, basi (variabili, cicli, funzioni), uso Notebook CoLab

Lezione 2 – Installare librerie comuni, leggere file, esempio con elenco spesa Amazon

Lezione 3 – usare Python sul proprio PC (ambienti, IDE) per processare dati localmente

Lezione 4 – progetto finale – analisi spesa Amazon con classificazione ChatGPT (o simili), grafici.

Il creatore di Python

- ❖ Python è stato creato nel 1989 dall'informatico olandese Guido van Rossum
- ❖ Guido ha creato Python in Google, poi ha fondato DropBox, e dopo una breve pensione è tornato a lavorare, in Microsoft.
- ❖ Ideato originariamente come linguaggio di scripting, Python si è poi evoluto come linguaggio completo
- ❖ Il nome fu scelto per via della passione di van Rossum e per la loro serie televisiva Monty Python's Flying Circus





Perché Python?



**Linguaggio conciso,
sarete veloci**

Ottimo per programmi usa-e-getta (risolvere un problema puntuale)



**Sintassi meno
«punitiva» di tanti altri
linguaggi.**

“A programming language that would be easy to read, write, and understand — both complex and simple at once”



**E' usabile da
principianti e non-
informatici**

Ma comunque molto potente, usato anche nella ricerca scientifica



**Ha tante librerie, che
consentono di fare
molto semplicemente
operazioni complesse.**



**Gli LLM (Claude,
ChatGPT, ...) sono
molto bravi a generare
codice Python!**

Popolarità online dei linguaggi di programmazione

Jan 2026	Jan 2025	Change	Programming Language	Ratings	Change
1	1		 Python	22.61%	-0.68%
2	4		 C	10.99%	+2.13%
3	3		 Java	8.71%	-1.44%
4	2		 C++	8.67%	-1.62%
5	5		 C#	7.39%	+2.94%
6	6		 JavaScript	3.03%	-1.17%
7	9		 Visual Basic	2.41%	+0.04%
8	8		 SQL	2.27%	-0.14%
9	11		 Delphi/Object Pascal	1.98%	+0.19%
10	18		 R	1.82%	+0.81%
11	32		 Perl	1.63%	+1.14%
12	10		 Fortran	1.61%	-0.42%
13	14		 Rust	1.51%	+0.34%
14	15		 MATLAB	1.40%	+0.34%
15	13		 PHP	1.38%	-0.00%

TIOBE Index – TIOBE
2026

A close-up photograph of a bright yellow Python snake. The snake is coiled elegantly around a piece of weathered wood. Its body curves from the bottom left towards the top right, with its head raised and facing towards the upper right corner. The snake's scales are clearly visible, showing a pattern of large, irregular polygons. The background is a solid, dark black, which makes the yellow snake stand out sharply.

Cosa posso fare con Python?

Esempi di applicazioni



○ AI

Keras, tensorflow, opencv

○ Embedded

MicroPython

Talk Arduino Day
2025

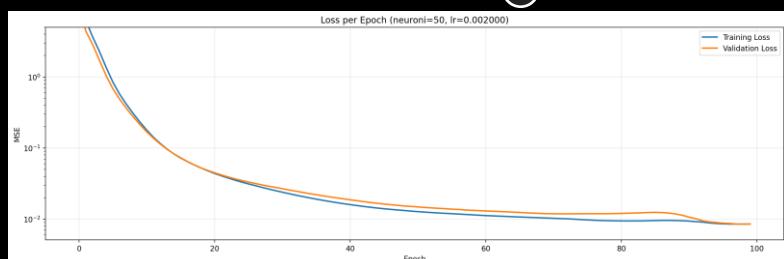
○ Scripting, Data Processing

pandas

Estrarre targhe e codici fiscali da centinaie di PDF

Matematica e grafici

numpy, matplotlib, manim



Giochi

pygame



Web

flask

Esempio: Fab-O-Matic



Pronti ?

- ❖ Per le prossime slides dovete avere un ambiente Python disponibile

Google COLAB



- ❖ <https://colab.research.google.com/>
- ❖ Python Notebooks
 - ❖ Un modo di salvare testo, dati, e programma in un unico ambiente online (in questo caso i file sono salvati su Google Drive)
 - ❖ Un modo di condividere programmi e sperimentare senza installazioni sul proprio PC
- ❖ Ci sono varie alternative più o meno free / open-source
 - ❖ [Try Jupyter!](#) (online)
 - ❖ pip install jupyter (installazioni locali)

Accedere al Notebook di riferimento

- ❖ <https://colab.research.google.com/drive/1NtbWbUVsBO0x9NM9N4e4c74jBtDNA8oZ?usp=sharing>
 - ❖ Contiene gli esempi del corso suddivisi per capitoli
- ❖ Aprire in un altro TAB un **notebook vuoto**

import this



Variabili e tipi

- ❖ Una variabile è una scatola ..
 - con un'etichetta («nome»)
 - con un contenuto («valore»)
 - con una capienza («tipo»)
- ❖ Le variabili consentono ai programmi di «prendere nota» e fare cose più interessanti
 - ❖ salvare un risultato intermedio («totale += 1»)
 - ❖ astrarre operazioni (come x in matematica)



Cos'è un tipo di dato ?

Il programma ha bisogno di manipolare dati diversi e li raggruppa in classi diverse

- Testo (str) – fra virgolette
- Vero/Falso (bool)
- Piccoli numeri (int)
- Numeri a virgola (float) – si usa il punto come separatore decimale (1.23 non 1,23)
- Numeri complessi (complex)
- Liste di elementi – si usano parentesi e virgole (1,2,3) o [1,2,3,4]
- Sequenze contigue (range)
- ...

Le operazioni fattibili dipendono dal tipo.

- Posso avere la lunghezza di un testo o di una lista, ma non di un bool
- Posso dividere due numeri, ma non fra due testi
- Posso calcolare il resto della divisione fra due numeri, non fra due liste

Tuttavia con Python posso cambiare come voglio il tipo di una variabile («dynamic typing»)

- A = 1
- A = [1, 2, 3] # Valido

Tipi diversi

2+5

7

"due"+"cinque"

'duecinque'

["due"] + ["cinque"]

???
???

2.0 + 5

7.0

Indovinate il
tipo

	1	1j	True
	1.0	'a'	False
	[1]	'ciao!'	1+1
	[1.0]	"a"	range(5)

Creiamo una variabile

- ❖ Scegliamo un nome (senza spazi!)
luogo, a, il_mio_nome
 - ❖ Chiediamo di assegniamo un contenuto
=
 - ❖ Diamo un valore (e implicitamente il tipo)
1, ‘Bergamo’
- luogo = ‘Bergamo’

Esercizi pratici

```
A = 1                      # type(A) ?  
A = "Pippo"                # type(A) ?  
A = [1, 2, "Tre"]          # len(A) ?  
A = 12.5                  # type(A) ?
```

```
Variabile_1 = 1  
variabile_1 = 2  
# qual è il valore di Variabile_1?
```

Creiamo più variabili

```
A,B = 0           qual'è il valore di A? di B ?  
C,D = 1,1         qual'è il valore di C? di D ?  
A = D           qual'è il valore di A?  
  
alluno_1, alluno_2 = "Alex", "Isabelle"      # alluno_1, alluno_2 ?  
alluno_2, alluno_1 = alluno_1, alluno_2      # cosa ha fatto questa operazione?
```

Cambiare il tipo della variabile

- ❖ «due» -> 2 usare int()
 - ❖ «2,145» -> 2.14 usare float()

Espressioni

- ❖ Fare Operatori su numeri → numero

```
+ - *      # somma, sottrazione, prodotto
/
//        # divisione intera: 5 // 2 == 2
%
**        # elevamento a potenza
abs       # valore assoluto: abs(-3) == 3
round     # restituisce l'intero più vicino
```

- ❖ Operatori su numeri → booleano

```
== != # uguale, diverso
> <  # maggiore, minore
>= <= # maggiore o uguale, minore o uguale
```

Prova

- ❖ `z = (1 + 2) * 3`
- ❖ `q1 = 6 / 2`
- ❖ `parola = "mappa" + "mondo"`
- ❖ True or False
- ❖ True and False
- ❖ `True != True`

```
import math  
math.factorial(40)
```

Liste in Python

- ❖ Le liste sono importanti perché in molte applicazioni dobbiamo manipolare elenchi più o meno complessi (elenchi di dati, files, matrici...)
- ❖ Esempio : elenco della frutta, da riordinare
 - sort
- ❖ Esempio : l'ultima lettera dell'alfabeto

Riassunto

- ❖ Sappiamo creare variabili
- ❖ Sappiamo fare espressioni (calcoli) con le variabili
- ❖ Sappiamo ordinare una lista

Domande ?

Altri elementi di linguaggio

- ❖ Scrivere un commento in mezzo al programma

```
# sono un commento  
A = 1
```

- ❖ `print(variabile o espressione)`

- ❖ Chiede a Python di scrivere nel terminale il valore o il risultato dell'espressione

```
A=1  
print(A)                                # Print è una funzione, le vediamo dopo  
print("A")
```

Stringhe formattate

E' molto utile poter scrivere il valore di una variabile in mezzo ad un testo con le «f-strings»

Provate:

```
nome = input("Come ti chiami?")
# Input è una funzione
print(f"Ciao {nome} !")
```

```
import math
print(f"PI vale circa {math.pi:.3}")
```

Esercizio

- ❖ Scrivere un programma che chieda un numero all'utente fra 1 e 10
- ❖ Stampare il numero richiesto di asterischi *

Output atteso:

Dammi un numero: 5

[Microsoft Copilot: Your AI companion](#)

<https://chatgpt.com/share/69621229-c9d0-8003-9402-32fcfe296482>

<https://claude.ai/share/c9320579-94ad-42b7-90c3-130aefa35c62>

Siamo pronti per imparare parole chiavi di Python

- ❖ if, for, while, def,...

```
['False', 'None', 'True', 'and', 'as',
'assert', 'async', 'await', 'break',
'class', 'continue', 'def', 'del', 'elif',
'else', 'except', 'finally', 'for', 'from',
'global', 'if', 'import', 'in', 'is',
'lambda', 'nonlocal', 'not', 'or', 'pass',
'raise', 'return', 'try', 'while', 'with',
'yield']
```



Istruzioni di controllo

```
if <condizione>:  
    (blocco)
```

Esercizio : Non abbiamo controllato che il numero fosse ≤ 10

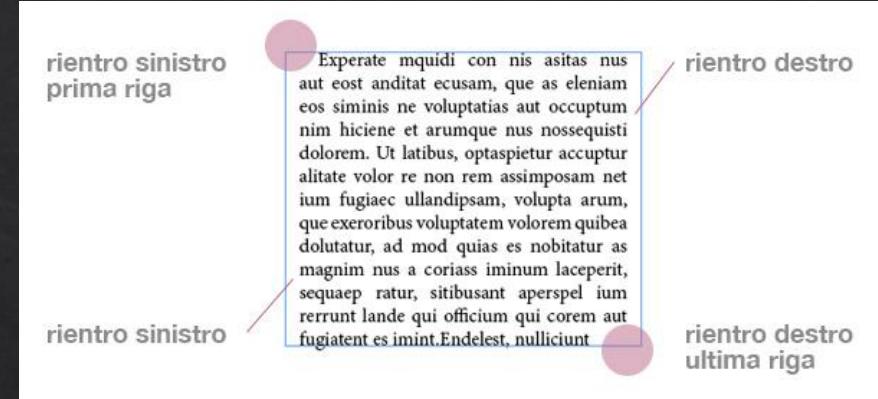
```
if <condizione>:  
    (blocco)  
else:  
    (blocco)
```

Indentazione («rientro») del codice

- ❖ Come nei libri di testo,
serve seguire regole di formattazione per facilitare la lettura.
- ❖ Python vuole codice leggibile («Bello è meglio di brutto»)

```
if 1 <= num_asterischi <= 10:  
    print("*"-* num_asterischi)
```

- ❖ Prima di un blocco, ci sono due punti :
- ❖ Quanti spazi devo mettere?
 - ❖ Non importa, ma bisogna essere consistente. Io suggerisco 2.
- Cosa succede se sbaglio ?
 - Messaggio di errore. Possibile avere la spiegazione.





While

- ❖ Ripeti un'operazione fin quando una condizione è vera
- ❖ Utile quando:
 - ❖ Non si sa quante volte una cosa deve essere fatta
 - ❖ Si vuole ripetere un'operazione «per sempre»

```
import machine
import time
# Crea un oggetto Pin come segnale in uscita
led = machine.Pin(25, machine.Pin.OUT)

while True:
    led.value(1)      # Accendi il led
    time.sleep(1)     # Aspetta 1s
    led.value(0)      # Spegni il led
    time.sleep(1)     # Aspetta 1s
```

Range

◆ RANGE(A)

- ◆ I numeri interi fra 0 e A escluso

$$\text{RANGE}(10) = [0,1,2,3,4,5,6,7,8,9]$$

Sono 10 elementi

◆ RANGE(A,B)

- ◆ I numeri interi fra A (incluso) e B (escluso)

$$\text{RANGE}(1,10) = [1,2,3,4,5,6,7,8,9]$$

Sono 9 elementi !

◆ RANGE(A, B, C)

- ◆ I numeri interi fra A (incluso) e B (escluso) a passo di C

$$\text{◆ RANGE}(1,20,2) = \text{gli interi dispari fra } 1 \text{ e } 20$$

$$[1,3,5,7,9,11,13,15,17,19]$$

Range - domande

- ❖ Voglio i primi 100 numeri partendo da 0
- ❖ Voglio i primi 100 numeri dispari partendo da 1

For

- ❖ For consente di ENUMERARE (ITERARE) su una sequenza di oggetti : una lista, un range.

```
frutti = ["mela", "banana", "arancia"]
```

```
for frutto in frutti:  
    print(frutto)
```

Spiegazione:

- **for frutto in frutti:** prendi ogni elemento da frutti
- **frutto** = la variabile che cambia ad ogni iterazione
- Stampa: mela → banana → arancia

Esercizio

- ❖ Stampare i numeri interi fra 1 e 10 inclusi.
- ❖ Stampare i quadrati dei numeri interi fra 1 e 10 inclusi.

Funzioni

- ❖ Un blocco di codice che svolge un compito specifico
- ❖ Puoi usarla più volte senza riscrivere il codice
- ❖ Rende il codice più organizzato e pulito
- ❖ Facilita la correzione degli errori



Esempi

- ❖ Funzione che non ritorna valori

```
def saluta(nome):  
    print(f"Ciao {nome} !")  
  
saluta("Alex")  
saluta("Isabelle")
```

- ❖ Funzione che ritorna valori

```
def applica_sconto(prezzo, percentuale):  
    sconto = prezzo * percentuale / 100  
    prezzo_finale = prezzo - sconto  
    return prezzo_finale  
  
prezzo_maglietta = 50  
nuovo_prezzo =  
applica_sconto(prezzo_maglietta, 20)  
print(nuovo_prezzo) #Stampa: 40.0
```

Esercizio

- ❖ Creare una funzione che ritorna la circonferenza di un cerchio di raggio r e stampare il risultato con 2 cifra decimali

Circonferenza = $2 * \pi * \text{raggio}$

Fine lezione 1

- ❖ Colab
 - ❖ Variabili , tipo, espressioni
 - ❖ Liste
 - ❖ Istruzioni di controllo if / for / while
 - ❖ Funzioni
-
- ❖ FEEDBACK

<https://form.typeform.com/to/IRhiMLnP>



Slicing

- ❖ **Slicing** : tagliare a fette una lista
 - ❖ `lista[inizio:fine]` dove inizio è l'indice 0-based, fine escluso.
 - ❖ `lista[:n]` – la lista dei n primi elementi
 - ❖ `lista[n:]` – la lista dall'n-esimo elemento in poi
 - ❖ `lista[:-n]` – la lista senza gli ultimi n elementi
 - ❖ `lista[-n:]` – la lista degli ultimi n elementi