

Fondamenti di Python

Patrick Mazzucchetti
Devops Specialist



HACKERSGEN

POWERED BY
SORINT
lab

Lezione 1



HACKERGEN

POWERED BY
SORINT_{lab}

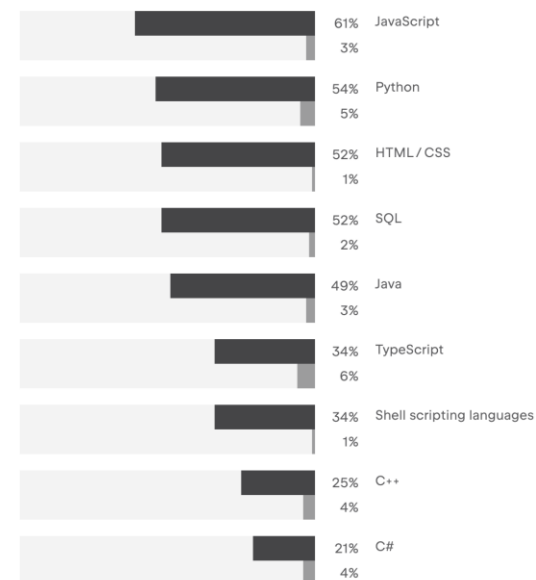
Uso di Python rispetto ad altri linguaggi

Python negli anni ha incrementato sempre di più la sua popolarità grazie ai suoi punti di forza a partire soprattutto dalla sintassi chiara e leggibile

Programming, scripting, and markup languages

■ Used in the last 12 months

■ Planning to adopt or migrate



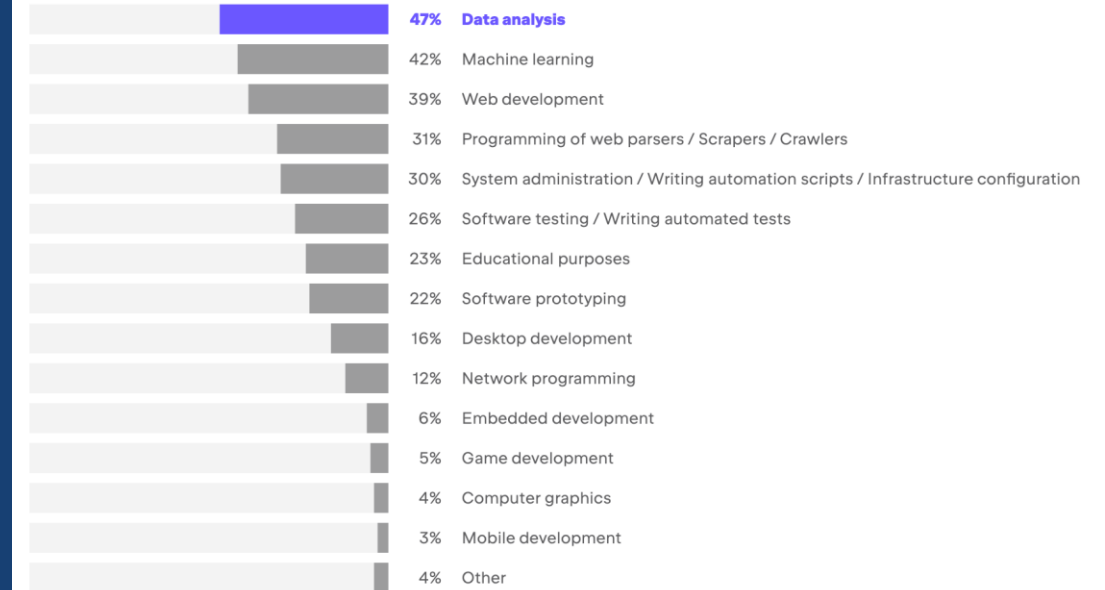
HACKERSGEN
POWERED BY
SORINT

Ambiti di utilizzo di Python

Il linguaggio Python dispone di tantissime e potenti librerie che coprono moltissimi ambiti

Soprattutto riguardanti IA, Machine Learning e analisi dei dati

What do you use Python for?



HACKERSGEN
POWERED BY
SORINT

Python

Python è

- Un linguaggio ad alto livello
- Un linguaggio interpretato
- Un linguaggio non fortemente tipizzato o tipizzato dinamicamente
- Un linguaggio multi paradigma
- Un linguaggio con Garbage Collector



HACKERSGEN

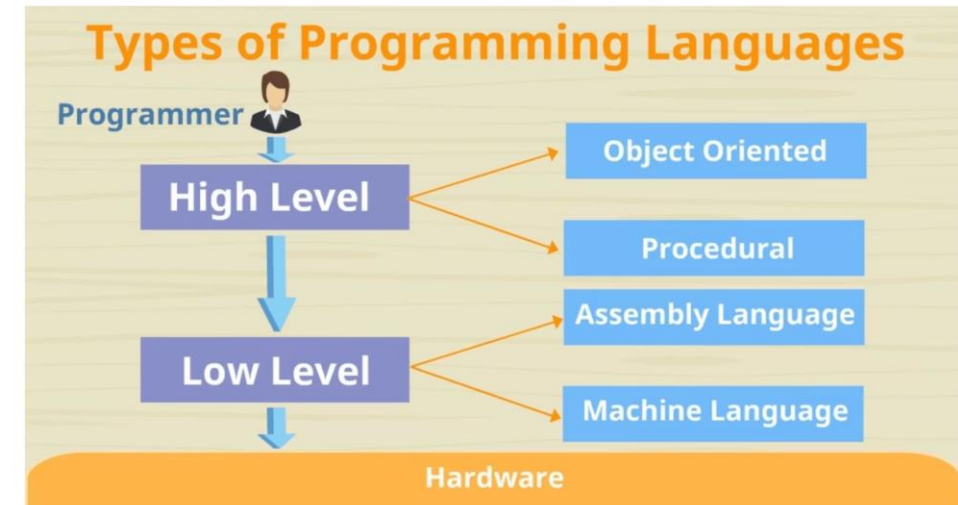
POWERED BY
SORINT

Alto livello vs Basso livello

I linguaggi di programmazione a basso livello sono simili al linguaggio macchina, non forniscono astrazione dai dettagli del funzionamento fisico dell'hardware

I linguaggi di programmazione ad alto livello invece si avvicinano di più alla logica umana

High-Level vs Low-Level

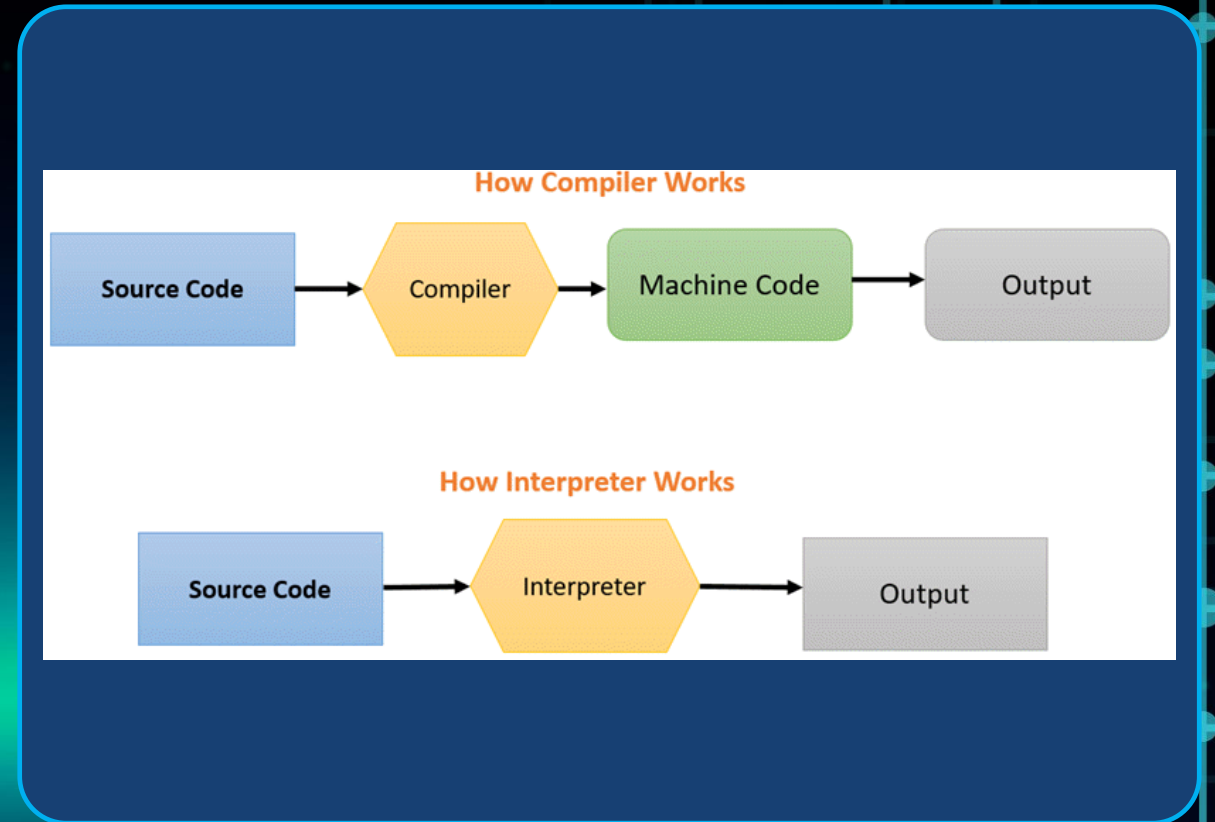


Compilato vs interpretato

I linguaggi compilati generalmente offrono un'esecuzione più rapida

I linguaggi compilati devono essere compilati più volte per adattarsi a diversi sistemi operativi e architetture

I linguaggi interpretati creano software che può essere eseguito solo se prima viene installato l'interprete



Fortemente tipizzato vs tipizzato dinamicamente

I linguaggi fortemente tipizzati sono più verbosi e meno flessibili perché bisogna dichiarare esplicitamente tutti i tipi delle variabili

I linguaggi fortemente tipizzati forniscono più sicurezza e meno errori a runtime a causa dei controlli rigorosi sui tipi delle variabili

Nei linguaggi tipizzati dinamicamente, il tipo di una variabile può cambiare durante l'esecuzione

Static vs Dynamic Typing

Java

Static typing:

<code>String name;</code>	Variables have types
<code>name = "John";</code>	Values have types
<code>name = 34;</code>	Variables cannot change type

JavaScript

Dynamic typing:

<code>var name;</code>	Variables have no types
<code>name = "John";</code>	Values have types
<code>name = 34;</code>	Variables change type dynamically

eggplantwander



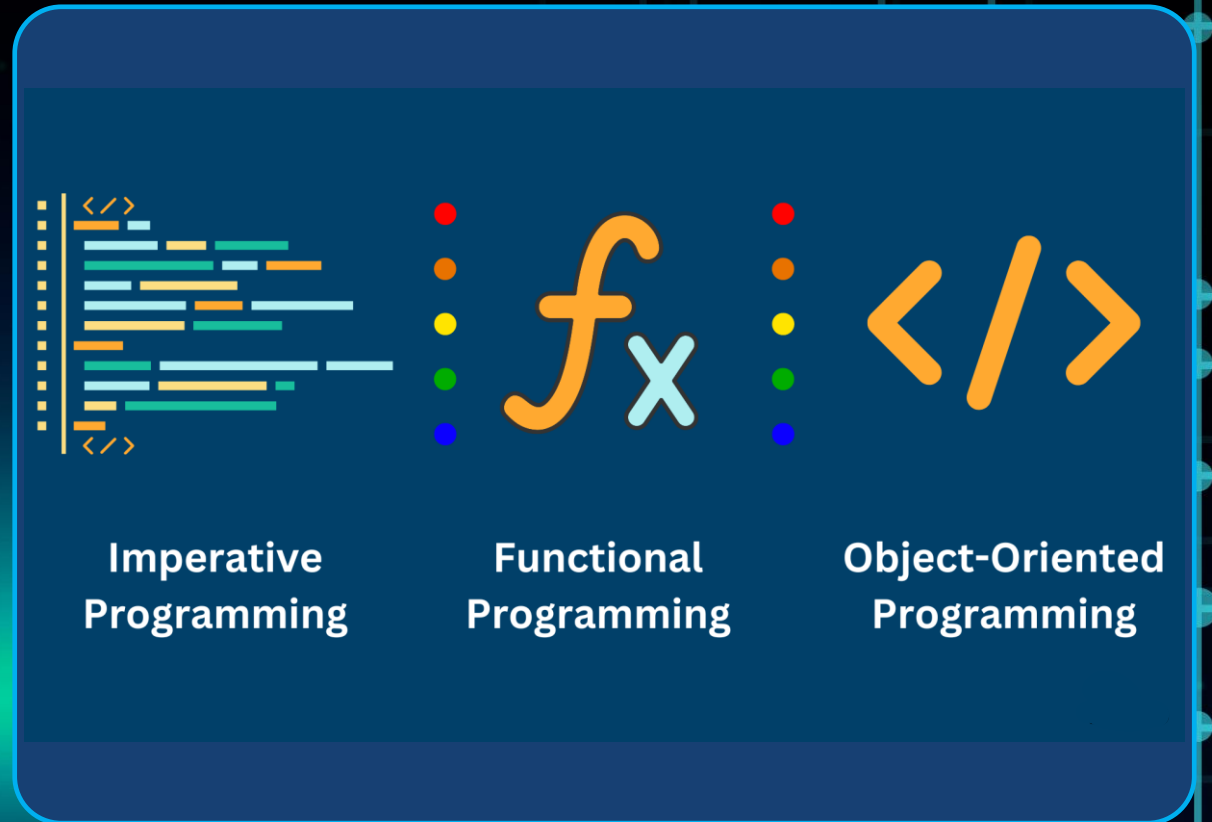
HACKERSGEN
POWERED BY
SORINT

Paradigmi

Imperative / procedural

Functional

Object Oriented



HACKERSGEN
POWERED BY
SORINT

Garbage Collector

In Python la memoria viene gestita automaticamente grazie al Garbage Collector

Il Garbage Collector è quel componente di un linguaggio di programmazione che si occupa di liberare in automatico la memoria dalle risorse non più in uso



HACKERSGEN
POWERED BY
SORINT

Python 2 vs Python 3

Nel 2008 viene lanciato Python 3, un grande aggiornamento che ha modificato alcuni aspetti chiave del linguaggio non rendendo compatibili le due versioni tra di loro



HACKERSGEN
POWERED BY
SORINT

Installazione di Python e tool



HACKERSGEN

POWERED BY
SORINT

Installazione su Windows

Scarica Python:

Visita <https://www.python.org/downloads/>.

Seleziona l'ultima versione stabile per Windows.

Scarica l'installer eseguibile (.exe).

Esegui l'installer:

1. Apri il file .exe scaricato.
2. Assicurati di selezionare "Add Python 3.x to PATH".
3. Clicca su "Install Now".
4. Riavvia il computer

3.Verifica l'installazione:

1. Apri il prompt dei comandi.
2. Digita `python --version` o `python -V` per verificare la versione.
3. Puoi anche digitare `python` per entrare nella shell interattiva.



HACKERSGEN
POWERED BY
SORINT

Installazione su Ubuntu

Apri il Terminale:

Puoi premere Ctrl + Alt + T per aprire il Terminale su Ubuntu.

Installazione di Python:

Verifica se Python è già installato digitando `python3 --version`. Se non è installato, esegui `sudo apt update` seguito da `sudo apt install python3`.

Verifica l'installazione:

Digita `python3 --version` o `python3 -V` per verificare la versione.

Puoi eseguire `python3` nel Terminale per entrare nella shell interattiva di Python.



HACKERSGEN
POWERED BY
SORINT

Installazione su MacOS

Scarica Python:

Visita <https://www.python.org/downloads/>.
Seleziona l'ultima versione stabile per macOS.
Scarica il pacchetto di installazione (.pkg).

Esegui l'installer:

Apri il file .pkg scaricato.
Segui le istruzioni di installazione, assicurandoti di selezionare "Install for all users" e "Add Python 3.x to PATH".

Verifica l'installazione:

Apri il Terminale.
Digita `python3 --version` o `python3 -V` per verificare la versione.
Puoi eseguire `python3` nel Terminale per entrare nella shell interattiva di Python.



Utilizzo dell'interprete Python

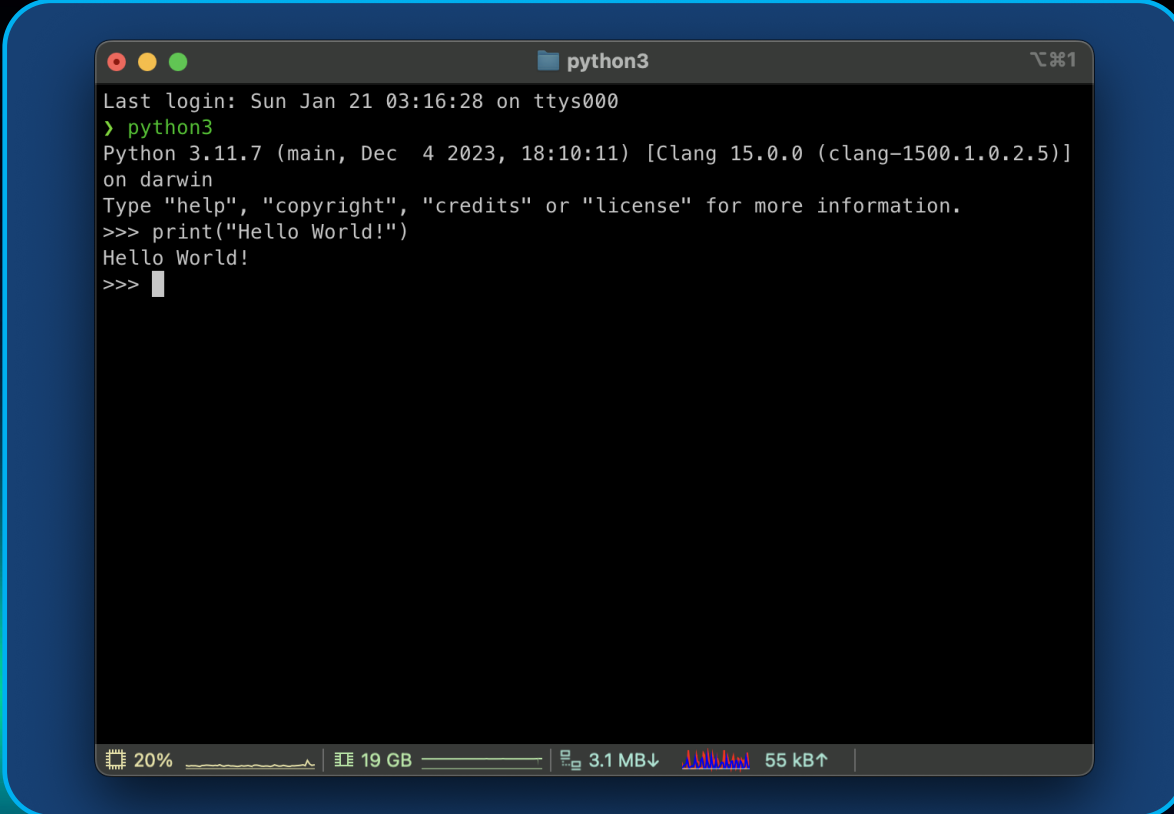


HACKERGEN

POWERED BY
SORINT

Uso dell'interprete da terminale

Python dà la possibilità di eseguire l'interprete direttamente dal terminale



```
python3
Last login: Sun Jan 21 03:16:28 on ttys000
> python3
Python 3.11.7 (main, Dec 4 2023, 18:10:11) [Clang 15.0.0 (clang-1500.1.0.2.5)]
on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello World!")
Hello World!
>>> █
```

The screenshot shows a macOS terminal window with a dark background and light blue text. The window title is 'python3'. The terminal output shows the last login time, the command to run 'python3', the Python version and build information, a prompt to enter help, the execution of 'print("Hello World!")', and the output 'Hello World!'. The terminal is currently at the 'python3' prompt with a cursor. At the bottom of the terminal window, there is a status bar showing system metrics: 20% battery, 19 GB of memory used, 3.1 MB of disk space used, and 55 kB of network activity.



Eseguire file Python

Tramite il comando python è possibile mandare in esecuzione i file python con estensione .py

A terminal window titled 'pmazzucchetti@PMAZZUCCHETTI-PC:~' with a dark background and light blue text. The window shows the command 'python3 main.py' being executed, resulting in the output 'Hello World!'. The terminal has standard macOS window controls (red, yellow, green buttons) at the top left. At the bottom, there is a status bar showing system information: 18% battery, 16 GB memory, 2.1 MB download speed, and 1.3 MB upload speed. A small macOS dock is visible at the bottom left of the terminal window.

```
pmazzucchetti@PMAZZUCCHETTI-PC:~  
Last login: Sun Jan 21 03:23:13 on ttys000  
> python3 main.py  
Hello World!  
✓ < at 03:23:49
```



HACKERSGEN
POWERED BY
SORINT

Variabili e tipi di dati



HACKERSGEN

POWERED BY
SORINT

Tipi di Dati

Categoria	Tipo
Testo	str
Numerico	int, float, complex
Sequenza	list, tuple, range
Mapping	dict
Set	set, frozenset
Booleani	bool
Binari	byte, bytearray, memoryview
None	None



Operatori ed espressioni



HACKERSGEN

POWERED BY
SORINT

Operatori Aritmetici

Nome	Operatore	Esempio
Addizione	+	$10 + 5 = 15$
Sottrazione	-	$10 - 5 = 5$
Moltiplicazione	*	$10 * 5 = 50$
Divisione	/	$9 / 5 = 1.8$
Modulo	%	$9 \% 5 = 4$
Elevazione a potenza	**	$10 ** 5 = 100000$
Divisione intera	//	$9 // 5 = 1$



Operatori di Assegnazione

Operatore	Esempio	Uguale ad
=	$x = 5$	$x = 5$
+=	$x += 3$	$x = x + 3$
-=	$x -= 3$	$x = x - 3$
*=	$x *= 3$	$x = x * 3$
/=	$x /= 3$	$x = x / 3$
%=	$x \% = 3$	$x = x \% 3$
//=	$x //= 3$	$x = x // 3$
**=	$x ** = 3$	$x = x ** 3$



Operatori di Confronto

Nome	Operatore	Esempio
Uguale	==	10 == 10
Diverso	!=	10 != 9
Maggiore di	>	10 > 5
Maggiore o uguale di	>=	10 >= 10
Minore di	<	5 < 10
Minore o uguale di	<=	8 <= 10



Operatori Logici

Nome	Operatore	Esempio
And	and	True and False = False
Or	or	True or False = True
Not	not	not True = False



Controllo del flusso



HACKERSGEN

POWERED BY
SORINT

Istruzione if

Per eseguire del codice sulla base di una condizione

```
main.py ×  
1 a = 10  
2 b = 15  
3  
4 if a > b:  
5     print("a è maggiore di b")  
6 else:  
7     print("a è minore di b")  
8  
9 # output: a è minore di b  
10
```



Cicli While

Per ripetere l'esecuzione di una porzione di codice fintanto che non si verifica una condizione

main.py x

```
1 a = 0
2 while a < 10:
3     print(a)
4     a = a + 1
5
6 # output: 0 1 2 3 4 5 6 7 8 9
7 |
```



HACKERSGEN
POWERED BY
SORINT

Cicli For

Per ripetere l'esecuzione di una porzione di codice, solitamente per un numero noto di volte

```
main.py ×  
1 a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]  
2 for i in a:  
3     print(i)  
4  
5 # output: 1 2 3 4 5 6 7 8 9 10  
6
```



HACKERSGEN
POWERED BY
SORINT

Cicli: Break

L'istruzione break permette di uscire da un ciclo

```
main.py ×  
1 a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]  
2 for i in a:  
3     if i == 5:  
4         break  
5     else:  
6         print(i)  
7  
8 # output: 1 2 3 4  
9 |
```



HACKERSGEN
POWERED BY
SORINT

Cicli: Continue

L'istruzione continue permettere di saltare il codice rimanente in una iterazione del ciclo e passare alla prossima

```
main.py ×  
1 a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]  
2 for i in a:  
3     if i == 5:  
4         continue  
5     else:  
6         print(i)  
7  
8 # output: 1 2 3 4 6 7 8 9 10  
9 |
```



Cicli: Clausola Else

La clausola else viene eseguita quando il ciclo termina naturalmente senza essere interrotto da un'istruzione break

main.py ×

```
1 a = [1, 2, 3, 4, 5]
2 for i in a:
3     if i == 6:
4         print("Trovato il numero 6")
5 else:
6     print("Numero 6 non trovato, esco nel ciclo all'interno della clausola else")
7
8 # output: Numero 6 non trovato, esco nel ciclo all'interno della clausola else
9 |
```



HACKERSGEN
POWERED BY
SORINT

Cicli: Pass

L'istruzione pass non fa nulla, viene utilizzata come segnaposto quando la sintassi richiede un'istruzione ma non serve fare nulla

```
main.py x
1 a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
2 for i in a:
3     if i == 5:
4         pass
5     else:
6         print(i)
7
8 # output: 1 2 3 4 6 7 8 9 10
9 |
```



HACKERSGEN
POWERED BY
SORINT

Cicli: Range

Permette di generare una sequenza di numeri in uno specifico intervallo

Si può indicare da dove iniziare, finire e di quanto incrementare la sequenza

```
main.py x
1  for i in range(10):
2      print(i)
3
4  # output: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
5  |
```



THANKS!

@pmazzucchetti



HACKERSGEN

POWERED BY

