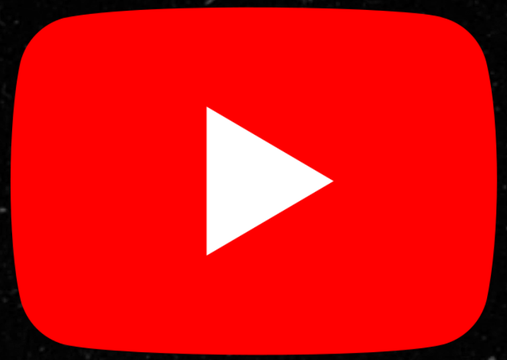
<Lorenzo Bruno>

PYTHON3 PER IL MACHINE LEARNING

New →

<PYTHON3> <Dov'è usato Python?>



Chi sono?

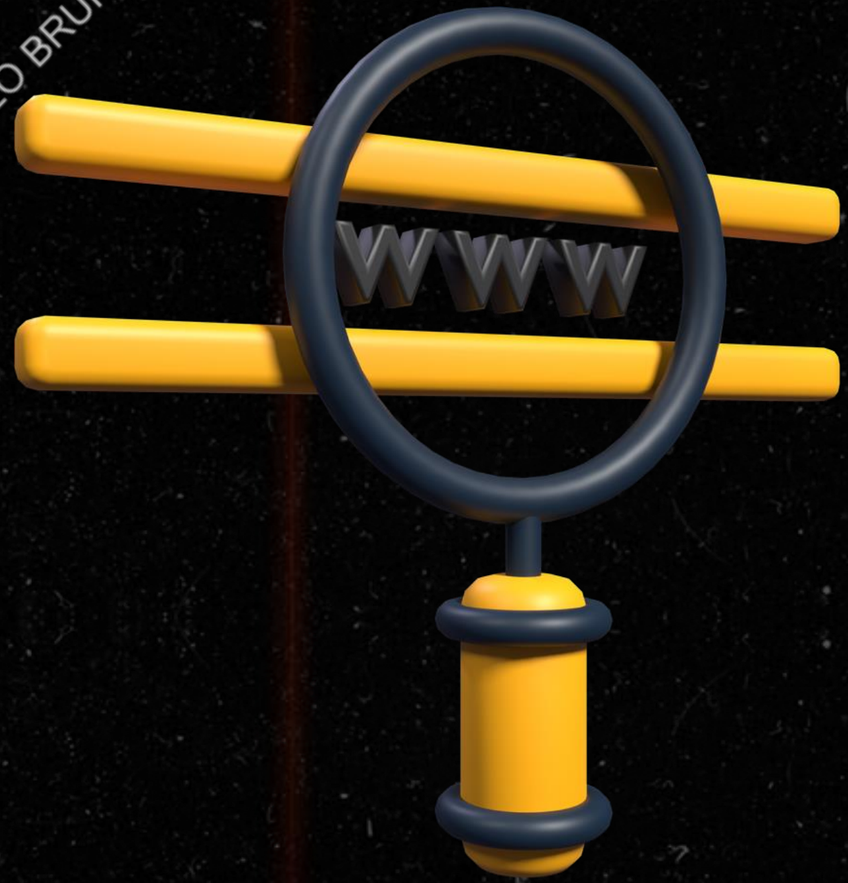
{01} Ex-Galileiano

Ex studente del
Liceo Scientifico
Galileo Galilei

{02} Appassionato cybersecurity

Studente di laurea magistrale in
Sicurezza Informatica a Milano

{03} Videogamer



Come è strutturato il corso

- Teoria
- Esercizi
- Quiz a fine di ogni modulo principale

 : Concetto di teoria importante

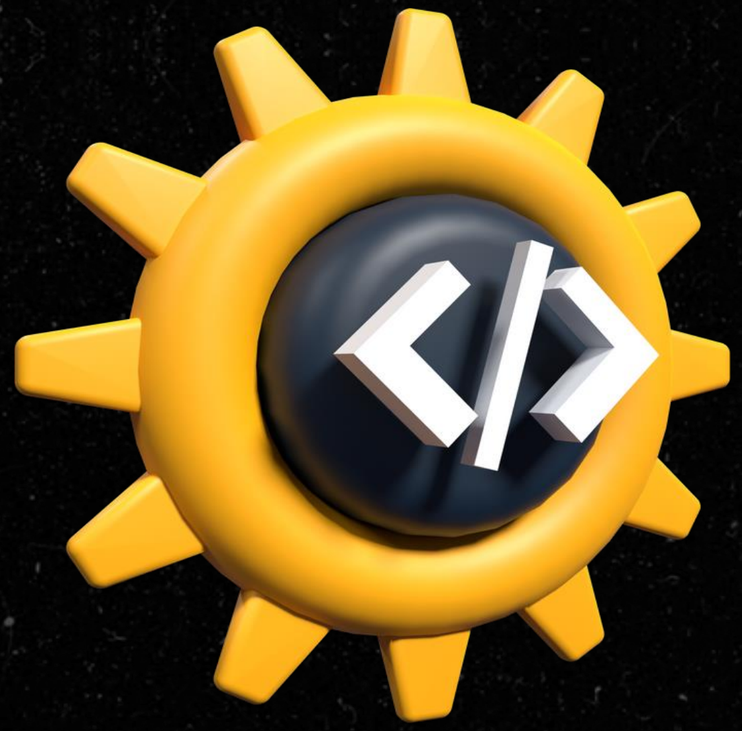
: Esempio da svolgere

{01} Programmare

La programmazione è l'arte di dire a un computer cosa fare attraverso una serie di istruzioni scritte in un linguaggio specifico.

È un processo creativo e logico che ci permette di sviluppare software, siti web, applicazioni e molto altro.

Padroneggiando i linguaggi di programmazione, è possibile creare soluzioni innovative, automatizzare compiti e dare vita alle idee nel mondo digitale.



<PYTHON3> <Iniziamo>

Python3 <Le basi>

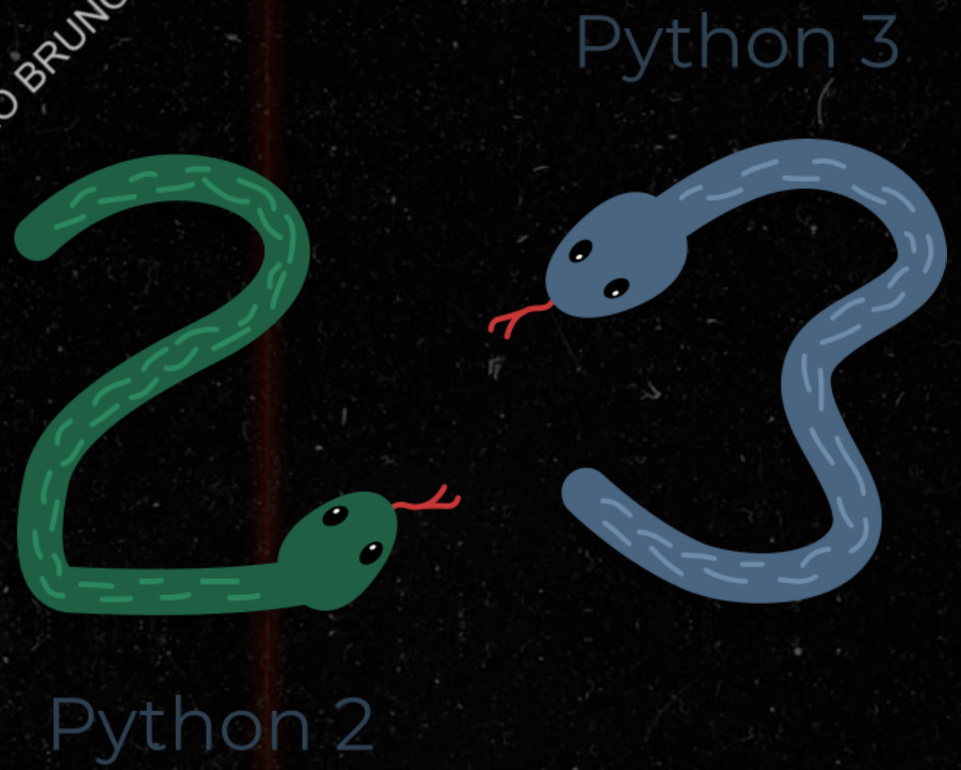
Esiste più di un Python

Esistono due versioni principali di Python, denominati Python 2 e Python 3.

Python 2 è una versione più vecchia. Il suo sviluppo è stato intenzionalmente interrotto, anche se questo non significa che non ci siano aggiornamenti.

Python 3 è la versione più recente (o, per essere più precisi, quella attuale) del linguaggio.

Python 3 non è solo una versione migliore di Python 2: è un linguaggio "completamente diverso", anche se molto simile al suo predecessore. Quando li si guarda da lontano, sembrano uguali, ma quando li si osserva da vicino si notano molte differenze.



Come funziona un programma?

Un programma rende un computer utilizzabile. Senza un programma, un computer, anche il più potente, non è altro che un soprammobile.

Immaginate di voler conoscere la velocità media raggiunta durante un lungo viaggio. Conoscete la distanza, conoscete il tempo, vi serve la velocità.

Naturalmente il computer sarà in grado di calcolarla, ma non è a conoscenza di elementi come la distanza, la velocità o il tempo. Pertanto, è necessario istruire il computer a:

- accettare un numero che rappresenta la distanza;
- accettare un numero che rappresenta il tempo di percorrenza;
- dividere il primo valore per il secondo e memorizzare il risultato;
- visualizzare il risultato (che rappresenta la velocità media) in un formato leggibile.

Queste quattro semplici azioni formano un **programma**. Naturalmente, questi esempi non sono formalizzati.

La parola chiave è **linguaggio**.

Come funziona un programma?

https://www.youtube.com/watch?v=cDA3_5982h8&t=106s

Linguaggio naturale vs programmazione

Possiamo dire che ogni linguaggio (**macchina o naturale**, non importa) è composto dai seguenti elementi:

- **Alfabeto**: insieme di simboli usati per comporre parole
- **Lessico** (dizionario): insieme di parole
- **Sintassi**: insieme di regole per determinare se una parola o un insieme di parole è ben formato
- **Semantica**: un insieme di regole che determinano il significato di come queste parole sono messe insieme tra loro

Esempi:

"Lui sono un'monglofiera"
"Una *vecchietta portare l'auto con i piedi*"
"*Non tirare le zampe ad una rana se non vuoi che la lattuga sia blu*"

Compilatore e/o Interprete

Programmare significa quindi comporre gli elementi che sono parte di un linguaggio affinché il computer esegua le azioni da noi desiderate.

Per fare ciò dobbiamo avere del codice che sia corretto dal punto di vista sintattico

Affinché il programma, con un linguaggio ad alto livello, sia trasformato in un linguaggio che la macchina possa comprendere è necessario che questo venga trasformato da un compilatore o da un interprete.

Vediamo le differenze

Compilatore

Strumento che converte il codice sorgente in un file (ad esempio .exe) contenente linguaggio macchina eseguibile dal processore.

Una volta che il codice è stato compilato, il file eseguibile è distribuibile a chiunque.

N.B.: La compilazione va eseguita ogni volta che vengono effettuate modifiche al codice sorgente

Interprete

Strumento che converte (interpreta) il codice sorgente in linguaggio macchina ogni volta che vogliamo eseguire il programma stesso

*N.B.: Python è un linguaggio interpretato**

Shell

Una shell è un programma che interpreta i comandi inseriti dall'utente e li esegue, interagendo con il sistema operativo. (Command line interpreter)

È l'ambiente che legge i comandi, li traduce in istruzioni comprensibili per il sistema e restituisce l'output all'utente.

Esistono vari tipi di shell, ognuna con sintassi e caratteristiche particolari:

- Bash (Bourne Again SHell): molto usata in ambienti Unix/Linux.
- cmd.exe: la shell predefinita di Windows prima di PowerShell.
- PowerShell: shell avanzata di Windows, orientata agli oggetti.

Terminale

Il terminale è l'interfaccia che permette di interagire con il sistema operativo tramite una shell.

In passato, il terminale era un dispositivo fisico con una tastiera e uno schermo (come i terminali VT100).

Oggi, il terminale è solitamente un programma software che permette di inviare comandi al sistema operativo e visualizzare il risultato.
Esempi di terminale includono:

- Terminale su macOS
- Gnome Terminal o Konsole su Linux
- Windows Terminal su Windows

Shell e terminale?

Aspetto	Terminale	Shell
Ruolo	Interfaccia per interagire con una shell	Interprete dei comandi che esegue istruzioni del sistema
Funzione	Visualizza l'output e accetta l'input dell'utente	Interpreta e processa i comandi
Esempi	Windows Terminal, Gnome Terminal, Konsole	Bash, Zsh, Fish, cmd.exe, PowerShell
Relazione	Ospita una o più shell	Viene eseguita all’interno di un terminale

Verificare Python

Verifichiamo che python sia correttamente installato nel nostro sistema

Come installarlo?

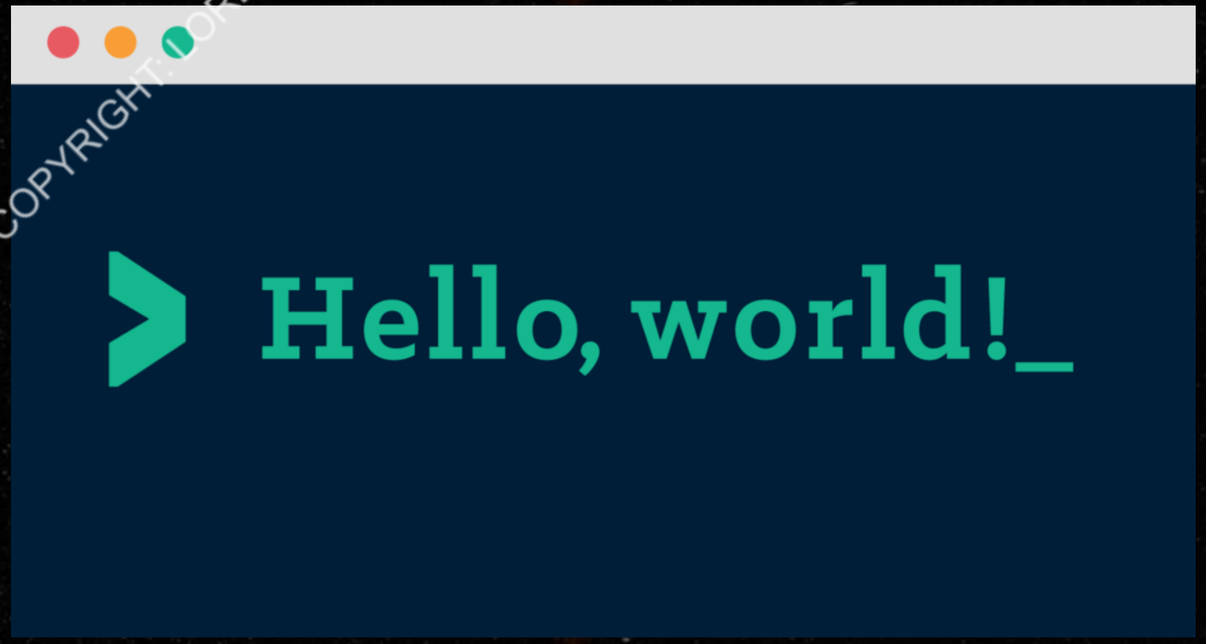
Lo zen di Python

- Bello è meglio di brutto
- Esplicito è meglio di implicito
- Semplice è meglio di complesso
- Complesso è meglio di complicato
- La leggibilità è importante

Il primo programma python

Scriviamo il nostro primo programma python:

Un semplice Hello World



Impara dai tuoi errori!

Proviamo a sbagliare di proposito il nostro codice e vediamo cosa succede

```
Print("This could be an error"
Prin("This is definitely an error")
```

Come imparare dagli errori

Quattro elementi da tenere in considerazione:

- Il traceback (ovvero il percorso che il codice compie attraverso le diverse parti del programma)
- la posizione dell'errore (il nome del file contenente l'errore, il numero di riga e il nome del modulo)

nota: il numero può essere fuorviante, poiché Python di solito mostra il punto in cui nota per la prima volta gli effetti dell'errore, non necessariamente l'errore stesso

- il contenuto della riga errata
- Il nome dell'errore e una breve descrizione

Riassumiamo brevemente

1. La funzione print() è una funzione integrata (built-in).

2. Le funzioni integrate, a differenza di quelle definite dall'utente, sono sempre disponibili e non devono essere importate. Python 3.8 è dotato di 69 funzioni integrate. L'elenco completo è riportato in ordine alfabetico nella Libreria standard di Python.

3. Per chiamare una funzione (questo processo è noto come invocazione di funzione o chiamata di funzione), è necessario utilizzare il nome della funzione seguito da parentesi. È possibile inserire argomenti in una funzione inserendoli all'interno delle parentesi. Gli argomenti devono essere separati da una virgola, ad esempio print(“Hello,”, “world!”). Una funzione print() “vuota” invia sullo schermo una riga vuota.

4. Le stringhe in Python sono delimitate da virgolette, ad esempio “Io sono una stringa” (virgolette doppie) o 'Anch'io sono una stringa' (virgolette singole).

5. I programmi sono raccolte di istruzioni. Un'istruzione è un comando per eseguire un compito specifico quando eseguito

Quiz n.1

Che cos'è il linguaggio macchina?

A. Un linguaggio di programmazione di basso livello costituito da bit/cifre binarie che il computer legge e comprende.

B. Un linguaggio di programmazione di basso livello costituito da cifre esadecimali che compongono le istruzioni del linguaggio di alto livello.

C. Un linguaggio di programmazione di medio livello costituito dal codice assembly progettato per il processore del computer

D. Un linguaggio di programmazione di alto livello costituito da elenchi di istruzioni che l'uomo può leggere e comprendere.

Quiz n.1

Che cos'è il linguaggio macchina?

A. Un linguaggio di programmazione di basso livello costituito da bit/cifre binarie che il computer legge e comprende.

B. Un linguaggio di programmazione di basso livello costituito da cifre esadecimali che compongono le istruzioni del linguaggio di alto livello.

C. Un linguaggio di programmazione di medio livello costituito dal codice assembly progettato per il processore del computer

D. Un linguaggio di programmazione di alto livello costituito da elenchi di istruzioni che l'uomo può leggere e comprendere.

Quiz n.2

Quali sono i quattro elementi fondamentali di un linguaggio?

A. Un alfabeto, un lessico, una fonetica e una semantica

B. Un alfabeto, un lessico, una sintassi e una semantica

C. Un alfabeto, una morfologia, una fonetica e una semantica

D. Un alfabeto, una fonetica, una fonologia e una semantica

Quiz n.2

Quali sono i quattro elementi fondamentali di un linguaggio?

A. Un alfabeto, un lessico, una fonetica e una semantica

B. Un alfabeto, un lessico, una sintassi e una semantica

C. Un alfabeto, una morfologia, una fonetica e una semantica

D. Un alfabeto, una fonetica, una fonologia e una semantica

{02} Tipi (Literals)

- Numeri
- Stringhe
- Boolean
- None
- Liste
- Tuple
- Set
- Dizionari

Tipi numerici

I numeri possono essere

- interi (int)
- decimali (float): rappresentati con il punto come separazione tra la parte intera e quella decimale.

Python3 ha anche altri tipi numerici ma per le nostre finalità non verranno trattati.

Stringhe

Le stringhe vengono utilizzate quando è necessario elaborare del testo, non i numeri. Ne sapete già qualcosa, ad esempio che le stringhe hanno bisogno di virgolette come i float hanno bisogno di punti.

Tuttavia, c'è un problema. Il problema è come codificare una citazione all'interno di una stringa già delimitata da virgolette.
Supponiamo di voler stampare un messaggio molto semplice che dica:

Mi piace “Monty Python”.

Come possiamo farlo senza generare un errore? Ci sono due possibili soluzioni.

Come gestire le stringhe

Il primo si basa sul concetto già noto di carattere di escape, che ricordiamo essere interpretato dal backslash (\). Il backslash può anche sfuggire alle virgolette.

- `print("Mi piace \"Monty Python\"")`

La seconda soluzione può essere un po' sorprendente. Python può usare un apostrofo al posto delle virgolette. Entrambi i caratteri possono delimitare le stringhe, ma bisogna essere coerenti.Se si apre una stringa con una citazione, bisogna chiuderla con una citazione. Se si inizia una stringa con un apostrofo, bisogna chiuderla con un apostrofo.

- `print('Mi piace "Monty Python"')`

Booleani

Utilizzati per rappresentare un valore molto astratto: la veridicità (vero/falso).

Ogni volta che si chiede a Python se un numero è maggiore di un altro, la domanda porta alla creazione di un dato specifico: un valore booleano.

I computer conoscono solo due tipi di risposte:

- Sì, vero;
- No, falso.

Non si otterrà mai una risposta del tipo: Non lo so o Probabilmente sì, ma non ne sono sicuro.

Questi due valori booleani sono rappresentati da: True False

N.B.: Questi simboli sono immutabili e vanno presi così come sono, compresi di Case Sensitive

True False

I valori booleani, True e False, prendono nome da George Boole (1815-1864), autore dell'opera "Le leggi del pensiero", che contiene la definizione di algebra booleana.

L'algebra booleana fa uso di due soli valori distinti:

Vero e Falso, indicati come 1 e 0.

Cosa significa nella pratica?

Cosa manca?

- None
- Liste
- Tuple

Vedremo questi tipi di dati successivamente

Quiz n.3

Qual è il tipo di questi valori?

- "Hello ", "007"
- "1.5", 2.0, 528, False

Quiz n.3

Qual è il tipo di questi valori?

- "Hello ", "007" -> Stringhe
- "1.5", 2.0, 528, False -> Stringa, Float, Booleano

{03} Operatori

Un operatore è un simbolo del linguaggio di programmazione in grado di operare sui valori.

Tuttavia, non tutti gli operatori di Python sono così ovvi, quindi esaminiamo alcuni degli operatori disponibili in Python e spiegheremo il loro uso e come interpretare le operazioni che eseguono.

Inizieremo con gli operatori associati alle operazioni aritmetiche più conosciute

Tutti gli operatori necessari

- `+` addizione (concatenazione)
- `-` sottrazione
- `*` moltiplicazione
- `/` divisione
- `//` divisione intera
- `%` modulo
- `**` esponenziale

{04} Variabili

È normale chiedersi come memorizzare i risultati delle operazioni effettuate per poterli utilizzare successivamente

Come si fa a salvare i risultati intermedi e a riutilizzarli per produrre quelli successivi?

Python offre speciali “scatole”, che vengono chiamate variabili - il nome stesso suggerisce che il contenuto di questi contenitori può essere variato in (quasi) tutti i modi.

Com'è fatta una variabile?

Una variabile è costituita da due elementi principali:

- Un nome: che sia valido e che segua sempre delle convenzioni condivise (es. Camel Case)
- Un valore: essendo una scatola, essa dovrà contenere al suo interno un valore qualsiasi

Variabile: nome

È necessario seguire alcune regole precise:

- Il nome della variabile deve essere composto da lettere maiuscole o minuscole, cifre e dal carattere _ (trattino basso);
- Il nome della variabile deve iniziare con una lettera;
- Il carattere di sottolineatura è una lettera;
- Le lettere maiuscole e minuscole sono trattate in modo diverso (Alice e ALICE sono due variabili diverse);
- Il nome della variabile non deve essere una delle parole riservate di Python, dette keywords

N.B.: Le stesse restrizioni si applicano ai nomi delle funzioni.

Keywords

['False', 'None', 'true', 'and', 'as', 'assert', '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']

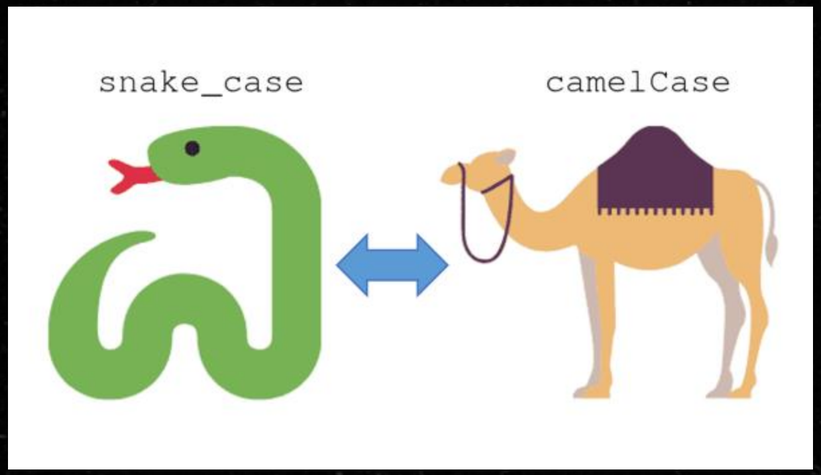
Si chiamano parole chiave o (più precisamente) parole chiave riservate. Sono riservate perché non devono essere usate come nomi: né per le variabili, né per le funzioni, né per qualsiasi altra entità denominata che si voglia creare.

Fortunatamente, grazie al fatto che Python è sensibile alle maiuscole e alle minuscole, è possibile modificare una qualsiasi di queste parole cambiando il caso di una qualsiasi lettera, creando così una nuova parola, che non è più riservata.

PEP 8 – Style Guide for Python Code

La "*PEP 8 -- Guida allo stile del codice Python*" raccomanda la seguente convenzione di denominazione per le variabili e le funzioni in Python:

- i nomi delle variabili devono essere minuscoli, con le parole separate da trattini bassi per migliorare la leggibilità (ad esempio, var, my_variable)
- i nomi delle funzioni seguono la stessa convenzione dei nomi delle variabili (ad esempio, fun, my_function)
- è anche possibile usare le lettere miste (ad esempio myVariable secondo convenzione cammello), ma solo in contesti in cui questo è già lo stile prevalente, per mantenere la compatibilità con la convenzione adottata.



Variabile: come crearla

Cosa possiamo mettere dentro una variabile?

Qualsiasi cosa.

È possibile utilizzare una variabile per memorizzare qualsiasi valore di uno dei tipi già presentati e molti altri non ancora mostrati.

Il valore di una variabile è quello che avete inserito al suo interno. Può variare tutte le volte che lo si desidera. Può essere un numero intero un momento, un float un momento dopo, per poi diventare una stringa.

Parliamo ora di due cose importanti: come si creano le variabili e come si inseriscono i valori al loro interno (o meglio, come si danno o si passano i valori alle variabili).

Creazione

Una variabile nasce come risultato dell'assegnazione di un valore. A differenza di altri linguaggi (es. C++), non è necessario dichiararla in modo particolare.

Se si assegna un valore a una variabile inesistente, la variabile viene creata automaticamente. Non è necessario fare altro.

La creazione (in altre parole, la sua sintassi) è estremamente semplice: basta usare il nome della variabile desiderata, poi il segno di uguale (=) e il valore che si vuole inserire nella variabile.

Manipolazione: shortcut

Expression	Shortcut operator
`i = i + 2 * j`	`i += 2 * j`
`var = var / 2`	`var /= 2`
`rem = rem % 10`	`rem %= 10`
`j = j - (i + var + rem)`	`j -= (i + var + rem)`
`x = x ** 2`	`x **= 2`

101010
101010
101010

esercizio_05, esercizio_06

Riassumiamo brevemente

- Una variabile viene creata o inizializzata automaticamente quando le si assegna un valore per la prima volta.
- Ogni variabile deve avere un nome univoco, un identificatore. Non può essere una parola chiave Python. Il primo carattere può essere seguito da trattini bassi, lettere e cifre. Gli identificatori in Python sono sensibili alle maiuscole e alle minuscole.
- Python è un linguaggio a **tipizzazione dinamica**, il che significa che non è necessario dichiarare le variabili. Per assegnare valori alle variabili, si può usare un operatore di assegnazione semplice nella forma del segno di uguale (=). Si può anche usare un operatore di assegnazione composto (= 1).
- Si possono anche utilizzare operatori di assegnazione composti (operatori di scelta rapida) per modificare i valori assegnati alle variabili, ad esempio: var += 1, o var /= 5 * 2.

<Company> <Lesson>

Features of ‘the Topic’ {

Step 01

Welcome to Step 01 of your journey! This is where you lay the foundation for all that is to come. In this step, you will begin by setting clear goals and intentions for yourself. What is it that you want to achieve?

Step 02

Once you have a clear vision in mind, it's time to take the first steps towards making it a reality. Break down your goals into smaller, manageable tasks that you can tackle one by one. R

Step 03

Welcome to Step 01 of your journey! This is where you lay the foundation for all that is to come. In this step, you will begin by setting clear goals and intentions for yourself. What is it that you want to achieve?

Who Should `Attend`

This workshop is designed for absolute beginners with little to no prior programming experience. Whether you're a student exploring new interests, a professional looking to switch careers, or someone curious about the world of coding, this workshop is perfect for you.

{  }Individuals’

Individuals with little to no prior programming experience, seeking to explore the world of coding.

{  }{Students}

Students interested in acquiring essential programming skills to enhance their academic and career prospects.

Programming} language

{ASM} Variables `

Variables are placeholders for storing data values. They can hold various types of data, such as numbers, strings (text), and Boolean (true/false) values. We'll learn how to declare variables, assign values to them, and manipulate their contents

{101010 101010 101010} Data }`Types

Different types of data require different ways of handling and storing them. We'll cover basic data types like integers, floating-point numbers, strings, and Boolean values. Understanding data types is crucial for writing robust and efficient code.

Gallery [Our Student`

What can you say about your projects? Share it here!

‘Recommendation{]

Practice Regularly:

Like any skill, programming requires consistent practice to master. Set aside dedicated time each day or week to work on coding exercises, solve problems, and build projects.

Start Small:

Begin with simple programs and gradually increase complexity as you become more comfortable with the language. Break down larger problems into smaller, manageable tasks to avoid feeling overwhelmed.

<Company> <Lesson>

Practical }Exercise: Simple Calculator `Program

Here's a step-by-step guide to building the calculator program:

Prompt User for Input: Ask the user to enter the first number, the operator (+, -, *, or /), and the second number.

Perform Calculation: Based on the operator entered by the user, perform the corresponding arithmetic operation on the two numbers.

Display Result: Output the result of the calculation to the user.

<Company> <Lesson>

Simple} Calculator `Program

To try out this program:

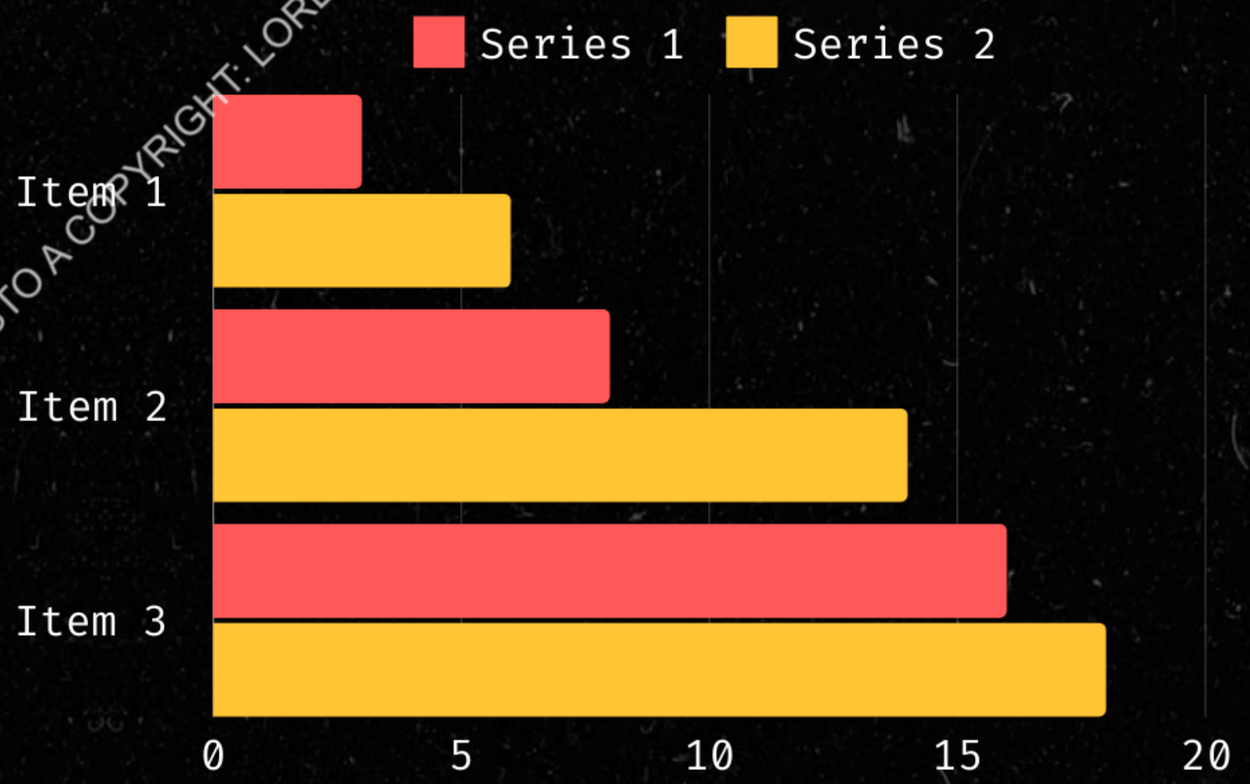
Copy the code into a Python environment or editor.
Run the program.
Enter the numbers and operator as prompted.
View the result and decide whether to perform another calculation or exit the program.
This exercise provides a hands-on opportunity to apply basic programming concepts such as user input, conditional statements, and functions. Feel free to modify and expand upon the program to add more functionality or improve its user experience.
Happy coding!


```python
def calculate():
    # Prompt user for input
    num1 = float(input("Enter the first number: "))
    operator = input("Enter the operator (+, -, *, /): ")
    num2 = float(input("Enter the second number: "))

    # Perform calculation based on operator
    if operator == '+':
        result = num1 + num2
    elif operator == '-':
        result = num1 - num2
    elif operator == '*':
        result = num1 * num2
    elif operator == '/':
        if num2 != 0:
            result = num1 / num2
        else:
            print("Error: Cannot divide by zero!")
            return

    # Display result
    print("Result:", result)

    # Ask user if they want to perform another calculation
    repeat = input("Do you want to perform another calculation? (yes/no): ")
    if repeat.lower() == 'yes':
        calculate()
    else:
        print("Thank you for using the calculator!")

# Call the calculate function to start the program
calculate()
```

Did `you know this}

?

Did you know that the Earth is the only planet in our solar system known to support life? With its perfect distance from the sun, a breathable atmosphere, and abundant water, Earth provides a unique environment for a diverse range of living organisms to thrive. The intricate web of ecosystems, from lush rainforests to vast oceans, showcases the incredible biodiversity of our planet. This delicate balance of nature highlights the importance of conservation and stewardship to ensure a sustainable future for all life on Earth.



Testimonials`}

Client name

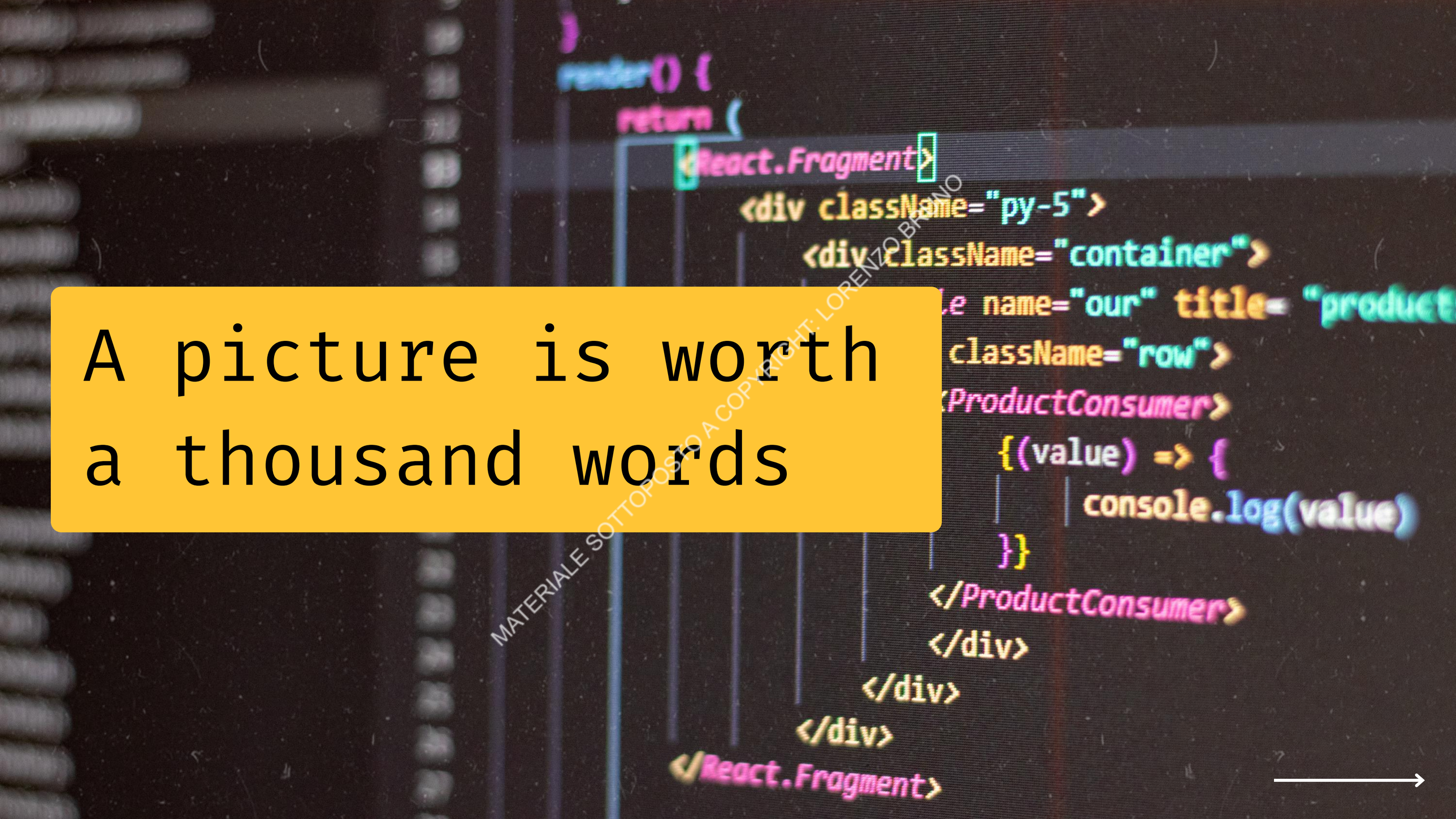
Elaborate on what
you want to discuss.

Client` name

Elaborate on what
you want to discuss.

Client name

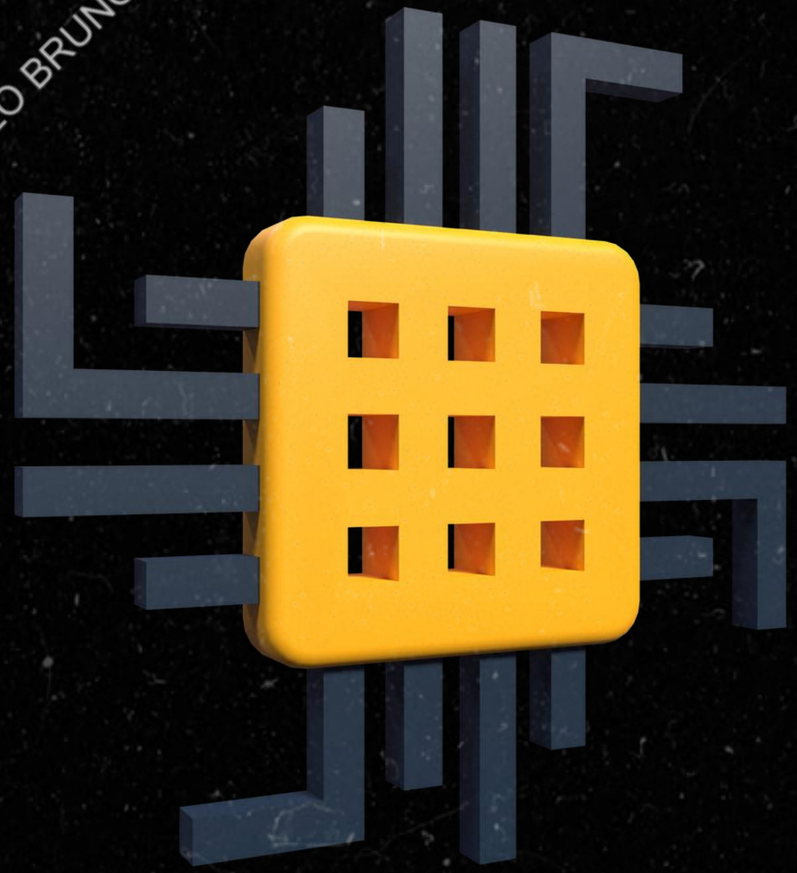
Elaborate on what
you want to discuss.


A picture is worth
a thousand words
MATERIALE SOTTOPOSTO A COPYRIGHT: LORENZO BRUNO

<Company> <Lesson>

Write an original statement or inspiring quote

— Include a credit, citation, or supporting message



That's all folks!

Embark on your journey into the world of programming with us! Join our workshop and take the first step towards becoming a proficient programmer. No matter your background or experience level, we're here to support you in your learning journey.

Ready to dive in? Let's start coding together!

For inquiries and registration, please contact [Your Contact Information].

[Include any logos, affiliations, or additional details as needed]

[End of Presentation]

Resource Page

Use these design resources in your Canva Presentation.

This presentation template uses the following free fonts:

1. Titles: Fira code
2. Headers: Fira code
3. Body Copy: Fira code

You can find these fonts online too.
Happy designing!

Don't forget to delete this page before presenting.

Credits

Lorenzo Bruno

Happy coding!