

Introduzione a Python

Francesco Gobbi

I.I.S.S. Galileo Galilei, Ostiglia

29 aprile 2025

Cos'è Python?

Python è un linguaggio di programmazione di alto livello, interpretato e orientato agli oggetti. È ampiamente utilizzato per sviluppare applicazioni web, analisi dei dati, automazione e machine learning.

Alcune sue peculiarità:

- ▶ Sintassi semplice e leggibile
- ▶ Linguaggio dinamico (tipizzazione dinamica)
- ▶ Supporta paradigmi multipli: imperativo, orientato agli oggetti e funzionale
- ▶ Grande ecosistema di librerie e framework

Differenze con altri linguaggi

Rispetto ad altri linguaggi come C++ o Java, Python è più semplice da scrivere e leggere. Alcune differenze:

- ▶ Non richiede la dichiarazione esplicita del tipo di variabili (tipizzazione dinamica)
- ▶ Non necessita di dichiarazione del tipo di ritorno in funzione
- ▶ Le funzioni sono oggetti di prima classe
- ▶ Python utilizza un "interprete" per eseguire il codice direttamente, mentre Java utilizza la compilazione bytecode (JVM)

Che cos'è un interprete?

Un interprete è un programma che esegue il codice sorgente riga per riga, senza necessità di compilazione preventiva.

In Python, l'interprete legge il codice, lo esegue immediatamente e restituisce i risultati, senza creare un file eseguibile.

Vantaggi:

- ▶ Semplicità di utilizzo
- ▶ Debugging più immediato
- ▶ Codice portabile tra diverse piattaforme

Svantaggi:

- ▶ Velocità inferiore rispetto ai linguaggi compilati


Cos'è una classe?

In Python, una **classe** è un modello per creare oggetti (istanze) che condividono attributi e metodi.

Una classe definisce le proprietà (attributi) e i comportamenti (metodi) che un oggetto può avere. Le classi sono il fondamento della programmazione orientata agli oggetti. **Quando si crea una classe si deve creare il costruttore, ovvero un "metodo" che ci permette di creare degli oggetti(istanze) di quella classe. Questo, in Python, è definito da `__init__`(parametri per creare l'oggetto) .**

Esempio di una semplice classe in Python:

```
1 class Persona:
2     def __init__(self, nome, et ):
3         self.nome = nome
4         self.et    = et
5
6     def saluta(self):
7         return f"Ciao, mi chiamo {self.nome} e ho {
            self.et } anni."
```

Listing 1: Definizione di una classe 

Cos'è un oggetto?

Un **oggetto** è un'istanza di una classe. Ogni oggetto creato da una classe avrà le proprie copie degli attributi definiti nella classe e potrà utilizzare i metodi della classe.

Ogni oggetto ha:

- ▶ **Attributi**: variabili che descrivono lo stato dell'oggetto
- ▶ **Metodi**: funzioni che definiscono i comportamenti dell'oggetto

Esempio di creazione di un oggetto:

```
1  persona1 = Persona("Marco", 25)
2  print(persona1.saluta())    # Output: Ciao, mi chiamo
    Marco e ho 25 anni.
```

Listing 2: Creazione di un oggetto

Attributi e Metodi

Ogni oggetto ha dei **attributi** che ne descrivono lo stato e dei **metodi** che definiscono cosa l'oggetto può fare.

- ▶ **Attributi:** sono variabili che appartengono all'oggetto. Nell'esempio, "nome" e "età" sono attributi della classe Persona.
- ▶ **Metodi:** sono funzioni definite all'interno della classe, che possono essere invocate sugli oggetti. Nell'esempio, `saluta()` è un metodo della classe Persona.

Esempio di chiamata di un metodo:

```
1  persona2 = Persona("Anna", 30)
2  print(persona2.saluta())    # Output: Ciao, mi chiamo
    Anna e ho 30 anni.
```

Listing 3: Chiamata di un metodo

Cos'è self in Python?

'self' è un riferimento all'oggetto stesso all'interno della classe. È un modo per accedere agli attributi e ai metodi dell'istanza corrente della classe.

In Python, ogni volta che creiamo un metodo all'interno di una classe, dobbiamo aggiungere `self` come primo parametro del metodo. Questo parametro consente al metodo di accedere agli attributi dell'oggetto.

N.B. `'self'` non è una parola chiave in Python, ma una convenzione. Potremmo chiamarlo anche in altro modo, ma è buona prassi utilizzare `'self'`.

Perché si usa self?

L'uso di `self` è necessario per distinguere tra variabili e metodi locali e variabili e metodi dell'oggetto.

Quando creiamo un oggetto, vogliamo che ogni oggetto possa avere valori distinti per gli attributi. `self` ci permette di accedere e modificare questi attributi per ciascun oggetto.

Esempio:

- ▶ `self.nome`: accede all'attributo `nome` dell'oggetto corrente
- ▶ `self.saluta()`: invoca il metodo `saluta` sull'oggetto corrente

Dove si usa self?

self viene utilizzato nei seguenti casi:

- ▶ Nei metodi di una classe per accedere agli attributi e metodi dell'istanza
- ▶ Per modificare o recuperare i valori degli attributi dell'oggetto
- ▶ Per riferirsi all'oggetto stesso all'interno della classe

Esempio:

```
1 class Persona:
2     def __init__(self, nome, et ):
3         self.nome = nome
4         self.et   = et
5
6     def cambia_nome(self, nuovo_nome):
7         self.nome = nuovo_nome # Modifica l'attributo
                                nome dell'oggetto
```

Listing 4: Uso di self in un metodo

Esempio con self

Esempio in cui utilizziamo self per modificare l'attributo di un oggetto:

```
1  persona1 = Persona("Marco", 25)
2  print(persona1.saluta())    # Output: Ciao, mi chiamo
    Marco e ho 25 anni.
3  persona1.cambia_nome("Luca")
4  print(persona1.saluta())    # Output: Ciao, mi chiamo
    Luca e ho 25 anni.
```

Listing 5: Modifica con self

Creazione di una nuova classe: Auto

La classe avrà gli attributi `marca`, `modello` e `anno` e un metodo `descrizione()` che restituisce una stringa con le informazioni dell'auto.

Ecco il codice:

```
1 class Auto:
2     def __init__(self, marca, modello, anno):
3         self.marca = marca
4         self.modello = modello
5         self.anno = anno
6
7     def descrizione(self):
8         return f"Auto: {self.marca} {self.modello},
           anno {self.anno}"
```

Listing 6: Classe Auto

Esempio di oggetto della classe Auto

Creiamo un oggetto della classe Auto e mostriamo l'output del metodo descrizione():

```
1 auto1 = Auto("Fiat", "Punto", 2010)
2 print(auto1.descrizione()) # Output: Auto: Fiat Punto
   , anno 2010
```

Listing 7: Oggetto Auto

Creazione di una nuova classe: Libro

La classe avrà gli attributi `titolo`, `autore` e `anno_pubblicazione` e un metodo `info()` che restituisce una descrizione del libro.

Ecco il codice:

```
1 class Libro:
2     def __init__(self, titolo, autore,
3         anno_pubblicazione):
4         self.titolo = titolo
5         self.autore = autore
6         self.anno_pubblicazione = anno_pubblicazione
7
8     def info(self):
9         return f"Libro: {self.titolo}, scritto da {
10             self.autore}, anno {self.
11                 anno_pubblicazione}"
```

Listing 8: Classe Libro

Esempio di oggetto della classe Libro

Creiamo un oggetto della classe Libro e mostriamo l'output del metodo info():

```
1 libro1 = Libro("1984", "George Orwell", 1949)
2 print(libro1.info()) # Output: Libro: 1984, scritto
    da George Orwell, anno 1949
```

Listing 9: Oggetto Libro

Riepilogo: Concetti chiave visti fino ad ora in Python

- ▶ **Python:** Linguaggio di programmazione ad alto livello, interpretato, orientato agli oggetti. È usato per web development, data analysis, automazione, e machine learning.
- ▶ **Classe:** Una classe è un modello per creare oggetti (istanze) che condividono attributi e metodi.
- ▶ **Oggetto:** Un'istanza di una classe che contiene attributi e metodi specifici per ogni elemento creato dalla classe.
- ▶ **Attributi e Metodi:** Gli attributi definiscono lo stato dell'oggetto, mentre i metodi definiscono i suoi comportamenti.
- ▶ **self:** È un riferimento all'oggetto stesso all'interno della classe, utilizzato per accedere agli attributi e metodi dell'istanza corrente.

Riepilogo: Concetti chiave visti fino ad ora in Python

- ▶ **Interprete:** Python è un linguaggio interpretato, il che significa che il codice viene eseguito riga per riga senza necessità di compilazione preventiva.
- ▶ **Differenze con altri linguaggi:** Python è più semplice rispetto a linguaggi come C++ o Java, grazie alla sua sintassi chiara e alla tipizzazione dinamica. Infatti, nella creazione di oggetti non è necessario definire il tipo.