Classi in Python

Le **classi** sono il cuore della **Programmazione Orientata agli Oggetti (OOP)** in Python. Permettono di raggruppare dati (attributi) e le funzioni che operano su quei dati (metodi) in un'unica unità logica.

1. Cosa Sono le Classi?

Una classe è essenzialmente un **modello** (o *blueprint*) per creare **oggetti** (istanze).

- **Classe:** Definisce la struttura (gli attributi) e il comportamento (i metodi) che gli oggetti di quel tipo avranno.
- **Oggetto (Istanza):** È una realizzazione concreta della classe, un'entità individuale creata seguendo quel modello.

2. Definizione di una Classe

Si usa la parola chiave `class` seguita dal nome della classe (per convenzione, in **PascalCase** o **CamelCase**).

Python
```
class Veicolo:
    # Qui vanno gli attributi e i metodi
    pass # 'pass' è usato per classi vuote
```

3. Il Costruttore ($\_\_init\_\_$)

Il metodo speciale `__init__` (chiamato **costruttore**) viene eseguito automaticamente quando si crea un nuovo oggetto della classe. È usato per **inizializzare** gli attributi dell'oggetto.

Esempio

Python
```
class Auto:
    # Il costruttore prende 'self' e altri argomenti per inizializzare gli
attributi
    def __init__(self, marca, modello, anno):
        # 'self' si riferisce all'istanza specifica che viene creata
        self.marca = marca      # Attributo dell'oggetto
        self.modello = modello  # Attributo dell'oggetto
        self.anno = anno        # Attributo dell'oggetto

# Creazione di oggetti (Istanze)
mia_auto = Auto("Fiat", "Panda", 2020)
tua_auto = Auto("Ford", "Focus", 2018)

# Accesso agli attributi
print(f"Marca della mia auto: {mia_auto.marca}")      # Output: Fiat
print(f"Modello della tua auto: {tua_auto.modello}")  # Output: Focus
```

---

4. Metodi

I **metodi** sono funzioni definite all'interno della classe e definiscono il comportamento degli oggetti. Devono sempre avere `self` come primo parametro.

Esempio con Metodi

Python
```
class Cane:
    def __init__(self, nome, razza):
        self.nome = nome
        self.razza = razza
        self.affamato = True

    # Metodo per definire un'azione
    def abbaia(self):
        return f"{self.nome} dice: Bau! Bau!"

    # Metodo che modifica lo stato interno dell'oggetto
    def mangia(self):
        if self.affamato:
            self.affamato = False
            return f"{self.nome} ha mangiato ed è soddisfatto."
        else:
            return f"{self.nome} non ha fame al momento."

# Creazione dell'oggetto
fido = Cane("Fido", "Labrador")

# Chiamata ai metodi
print(fido.abbaia())      # Output: Fido dice: Bau! Bau!
print(fido.mangia())      # Output: Fido ha mangiato ed è soddisfatto.
print(fido.affamato)      # Output: False (lo stato è cambiato)
print(fido.mangia())      # Output: Fido non ha fame al momento.
```

---

5. Ereditarietà

L'**ereditarietà** è un principio fondamentale dell'OOP che consente a una classe (chiamata **sottoclasse** o **classe figlia**) di ereditare gli attributi e i metodi da un'altra classe (chiamata **superclasse** o **classe padre**). 🧬

Per ereditare, si inserisce il nome della classe padre tra parentesi tonde nella definizione della classe figlia.

Esempio di Ereditarietà

Python
```
# Superclasse (Classe Padre)
class Animale:
    def __init__(self, nome):
        self.nome = nome

    def presenta(self):
        return f"Ciao, sono un Animale e mi chiamo {self.nome}."
```

```
# Sottoclasse (Classe Figlia) che eredita da Animale
class Gatto(Animale):
    def __init__(self, nome, colore):
        # Chiama il costruttore della classe padre per inizializzare 'nome'
        super().__init__(nome)
        self.colore = colore # Aggiunge un nuovo attributo

    # Metodo specifico della sottoclasse
    def miagola(self):
        return f"{self.nome} miagola (il suo colore è {self.colore})."

# Creazione dell'oggetto Gatto
palla_di_pelo = Gatto("Arturo", "rosso")

# Usa il metodo ereditato dalla classe Animale
print(palla_di_pelo.presenta()) # Output: Ciao, sono un Animale e mi chiamo
Arturo.

# Usa il metodo specifico della classe Gatto
print(palla_di_pelo.miagola()) # Output: Arturo miagola (il suo colore è rosso).
```

Riassunto Concetti Chiave

Termine	Significato
Classe	Modello per gli oggetti.
Oggetto/Istanza	Realizzazione concreta della classe.
Attributo	Variabile che memorizza dati all'interno di un oggetto (i suoi dati).
Metodo	Funzione definita all'interno di una classe (il suo comportamento).
`self`	Riferimento all'istanza corrente dell'oggetto.
`__init__`	Metodo costruttore per inizializzare gli attributi.
`super()`	Funzione usata per chiamare metodi della classe padre (superclasse).