

# La Programmazione ad Oggetti in Python

**Docente:** Ambra Demontis

Anno Accademico: 2023 - 2024



University of Cagliari, Italy

Department of Electrical and Electronic Engineering



# La Programmazione ad Oggetti in Python

In queste slide vedremo:

- Il metodo \_\_new\_\_
- La classe object



# **Definizione di Classi**

Consideriamo il diagramma di classe della classe libro mostrato sotto.

# libro titolo autore

Il metodo \_\_init\_\_ viene richiamato automaticamente quando un nuovo oggetto viene creato e ci permette di inizializzarlo.

Questa funzione spesso viene anche erroneamente chiamata costruttore dell'oggetto.



Creiamo una classe *libro* che abbia gli attributi: *titolo* e *autore*.

class CLibro:

def \_\_init\_\_(self, titolo, autore):

• • •

Il metodo init riceve i valori degli attributi dell'oggetto e li inizializza.



Creiamo una classe *libro* che abbia gli attributi: *titolo*, *autore*.

class CLibro:

```
def __init__(self, titolo, autore):
    self.titolo = titolo
    self.autore = autore
```

Il metodo \_\_init\_\_ viene spesso chiamato metodo "costruttore". Questo farebbe pensare che è il metodo \_\_init\_\_ a costruire (creare) l'istanza.

E' realmente il metodo \_\_init\_\_ che costruisce l'oggetto? ..



class CLibro:

```
def __init__(self, titolo, autore):
    self.titolo = titolo
    self.autore = autore

oggetto_libro = CLibro("LPO", "Dusty Phillips")
```

Ci sono 2 cose in contrasto con il fatto che sia \_\_init\_\_ a costruire l'oggetto..

class CLibro:

```
def __init__(self, titolo, autore):
    self.titolo = titolo
    self.autore = autore
```

oggetto\_libro = CLibro("LPO", "Dusty Phillips")

1. Il metodo \_\_init\_\_ riceve in automatico come primo argomento l'istanza dell'oggetto.



class CLibro: def \_\_init\_\_(self, titolo, autore): self.titolo = titolo self.autore = autore oggetto\_libro = CLibro("LPO", "Dusty Phillips") 2. CLibro("LPO", "Dusty Phillips") restituisce un'istanza dell'oggetto ma il

metodo init non restituisce alcun valore (non ha un'istruzione return)..



Quindi chi si occupa di creare l'oggetto?

Esiste un metodo che ha questo scopo: il metodo statico \_\_new\_\_

Perchè nella definizione delle classi non lo vediamo?

# Definizione di Classi - la Classe Padre Object

Sebbene dalla definizione di una classe non si veda, <u>tutte le classi ereditano</u> <u>da una classe chiamata object</u>.

Definire una classe con la sintassi:

class <nome\_classe>:

È uguale a definirla con la sintassi:

class < nome\_classe > (object):

La classe object implementa diversi metodi che vengono ereditati da tutte le classi (compreso il metodo \_\_new\_\_).



#### Definizione di Classi - Creazione di un'Istanza

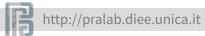
```
Quando usiamo sintassi:
<variabile> = <nome classe>()
Viene prima invocato prima il metodo __new__ che crea (costruisce) l'oggetto.
Se restituisce un'istanza di classe (come di default) viene poi invocato il metodo
__init__ che inizializza l'oggetto (setta i valori degli attributi).
Quindi in realtà:
-__init__ è chiamato impropriamente "costruttore"
-in realtà il metodo "costruttore" è il metodo __new__
```



# Definizione di Classi - i metodi \_\_new\_\_ e \_\_init\_\_

Inoltre il metodo \_\_new\_\_ come anche il metodo \_\_init\_\_ vengono ereditati dalla classe object.

Per questo è possibile creare una classe senza definire né metodo \_\_init\_\_ né il metodo new.



#### Definizione di Classi - i metodi \_\_new\_\_ e \_\_init\_\_

```
class CSomma:
 @staticmethod
 def somma(v1, v2):
   return v1 + v2
                             Crea un'istanza dell'oggetto (vengono invocati
oggetto_somma = CSomma()
                                _new___ e __init___ della classe object).
risultato_somma = oggetto_somma.somma(3, 5)
print(risultato_somma)
```

Stamperà: 8

