

# La Programmazione ad Oggetti in Python

**Docente:** Ambra Demontis

**Anno Accademico:** 2020 - 2021

Corso di Laurea in Ingegneria Elettrica, Elettronica e Informatica



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.



# Definizione di Classi - il Metodo `__init__`

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.

# Definizione di Classi - il Metodo `__init__`

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.

# Definizione di Classi - il Metodo `__init__`

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

```
class CLibro:
```

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

# Definizione di Classi - il Metodo `__init__`

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? ..

# Definizione di Classi - il Metodo `__init__`

```
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..



# Definizione di Classi - il Metodo `__init__`

```
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.

# Definizione di Classi - il Metodo `__init__`

```
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`)..

# Definizione di Classi - il Metodo `__new__`

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, **tutte le classi ereditano da una classe chiamata *object*.**

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
```

```
oggetto_somma = CSomma()
```

```
risultato_somma = oggetto_somma.somma(3, 5)
```

```
print(risultato_somma)
```

Crea un'istanza dell'oggetto (vengono invocati `__new__` e `__init__` della classe object).

Stamperà: 8