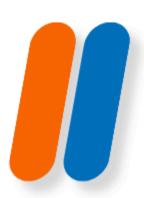


### Funzioni come oggetti di prima classe:

 In Python, le funzioni sono trattate come oggetti di prima classe, il che significa che possono essere assegnate a variabili, passate come argomenti a altre funzioni e restituite da altre funzioni.

#### **Funzioni Lambda:**

 Le funzioni lambda sono funzioni anonime definite usando la parola chiave lambda. Sono utili per operazioni semplici e veloci che non richiedono una definizione completa della funzione



- In Python, le funzioni sono trattate come oggetti di prima classe, il che significa che possono essere create dinamicamente, passate come argomenti a altre funzioni, restituite da altre funzioni e assegnate a variabili.
- Questa caratteristica consente una programmazione flessibile e potente, permettendo di creare funzioni di ordine superiore che possono, ad esempio, accettare altre funzioni come parametri o restituire nuove funzioni.



Le funzioni lambda in Python sono funzioni anonime definite usando la parola chiave lambda.

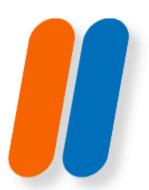
Sono utili per eseguire operazioni brevi e semplici senza dover definire una funzione separata utilizzando def. Le funzioni lambda sono particolarmente utili quando si lavora con funzioni di ordine superiore come map(), filter(), e reduce(), dove una funzione inline può semplificare il codice.

### Funzioni Incorporate (Built-in):

 Python fornisce molte funzioni incorporate come len(), max(), min(), sum(), ecc., che facilitano molte operazioni comuni.

#### **Decoratori:**

 I decoratori sono una caratteristica potente che permette di modificare il comportamento di una funzione o un metodo. Sono applicati utilizzando la sintassi @decorator\_name.



Python fornisce una vasta gamma di funzioni incorporate che facilitano molte operazioni comuni, come len(), max(), min(), sum(), tra molte altre.

Queste funzioni sono sempre disponibili in qualsiasi ambiente Python e aiutano a scrivere codice conciso e leggibile senza dover reinventare funzionalità di base.



I decoratori in Python sono una caratteristica potente che consente di modificare il comportamento di una funzione o di un metodo senza alterarne il codice sorgente.

I decoratori sono applicati utilizzando la sintassi @decorator\_name sopra la definizione della funzione. Possono essere utilizzati per una varietà di scopi, come logging, controllo degli accessi, memoization e altro ancora.



#### Argomenti Posizionali e Nominali:

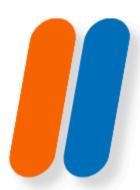
 Le funzioni in Python possono accettare argomenti sia posizionali che nominali. Gli argomenti posizionali sono forniti in base alla loro posizione, mentre quelli nominali sono forniti in base al loro nome.

#### Argomenti di Default:

 Le funzioni possono avere argomenti di default, il che significa che se un argomento non viene fornito, assume un valore predefinito.

#### **Argomenti Arbitrari:**

 Con l'uso di \*args e \*\*kwargs, le funzioni possono accettare un numero variabile di argomenti posizionali e di parola chiave, rispettivamente.



Le funzioni in Python possono accettare sia argomenti posizionali che nominali.

Gli argomenti posizionali sono quelli forniti nella stessa posizione in cui sono definiti nella funzione, mentre gli argomenti nominali sono specificati utilizzando il nome del parametro, consentendo una maggiore chiarezza e flessibilità nella chiamata della funzione.



Questo permette anche di definire valori predefiniti per i parametri.

Le funzioni in Python possono avere argomenti di default, che vengono utilizzati se non viene fornito un valore specifico durante la chiamata della funzione.

Questo consente di creare funzioni più flessibili e di evitare la necessità di sovraccaricare le funzioni per diversi scenari di utilizzo.



Utilizzando \*args e \*\*kwargs, le funzioni possono accettare un numero variabile di argomenti posizionali e di parola chiave, rispettivamente.

Questo è utile quando non si conosce in anticipo il numero di argomenti che verranno passati alla funzione o quando si vuole creare interfacce flessibili per le funzioni.



#### **Funzioni Speciali**

#### Funzioni Magic (o Dunder):

 Le funzioni magic (o "dunder", doppio underscore) sono funzioni speciali che iniziano e finiscono con doppio underscore, come \_\_init\_\_, \_\_str\_\_, \_\_repr\_\_, \_\_add\_\_, ecc.
Queste funzioni permettono di definire o sovrascrivere comportamenti speciali per gli oggetti.

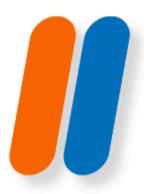
#### Funzione \_\_init\_\_:

 È il costruttore della classe, chiamato automaticamente quando un'istanza della classe viene creata.



Le funzioni magic, conosciute anche come "dunder" (doppio underscore), sono funzioni speciali che iniziano e finiscono con doppio underscore, come \_\_init\_\_, \_\_str\_\_, \_\_repr\_\_, \_\_add\_\_, ecc.

Queste funzioni permettono di definire o sovrascrivere comportamenti speciali per gli oggetti, come la creazione di oggetti, la rappresentazione di oggetti come stringhe, le operazioni aritmetiche, e molto altro.



Sono fondamentali per implementare l'interfaccia e il comportamento degli oggetti in modo personalizzato.

La funzione \_\_init\_\_ è il costruttore della classe, chiamato automaticamente quando un'istanza della classe viene creata.

È utilizzata per inizializzare gli attributi dell'istanza e può accettare argomenti per configurare lo stato iniziale dell'oggetto.

Questa funzione è essenziale per configurare correttamente le istanze di una classe.



#### **Funzioni Speciali**

#### Funzione \_\_str\_\_ e \_\_repr\_\_:

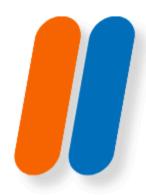
\_str\_\_ è usata per definire una rappresentazione
"informale" o user-friendly dell'oggetto, mentre \_\_repr\_\_
è per una rappresentazione più dettagliata e ufficiale,
utile per il debugging.

#### Funzione \_\_call\_\_:

 Permette a un'istanza di una classe di essere chiamata come una funzione.

#### Funzione \_\_getitem\_\_, \_\_setitem\_\_, \_\_delitem\_\_:

 Permettono di accedere, modificare e eliminare elementi usando la sintassi delle liste o dei dizionari.



La funzione \_\_str\_\_ è usata per definire una rappresentazione "informale" o user-friendly dell'oggetto, ideale per la stampa e la visualizzazione all'utente finale.

La funzione \_\_repr\_\_ fornisce una rappresentazione più dettagliata e ufficiale dell'oggetto, utile per il debugging e lo sviluppo. \_\_repr\_\_ dovrebbe restituire una stringa che, se passata a eval(), dovrebbe creare un oggetto equivalente, se possibile.



La funzione \_\_call\_\_ permette a un'istanza di una classe di essere chiamata come una funzione.

Questa caratteristica può essere utilizzata per creare oggetti che agiscono come funzioni, offrendo un'interfaccia più naturale e intuitiva per certe operazioni.



Queste funzioni speciali permettono di accedere, modificare e eliminare elementi usando la sintassi delle liste o dei dizionari. \_\_getitem\_\_ è utilizzata per l'accesso in lettura, \_\_setitem\_\_ per l'accesso in scrittura e \_\_delitem\_\_ per l'eliminazione di elementi.

Queste funzioni sono fondamentali per creare oggetti che si comportano come container personalizzati.



