Procedure e funzioni

In Python, una funzione è un blocco di codice riutilizzabile che esegue un'attività specifica. Le funzioni ti permettono di organizzare il codice, renderlo più leggibile e riutilizzabile. Ecco come si definiscono le funzioni in Python:

Sintassi di base

```
def nome_funzione(parametri):
    return valore_di_ritorno # Opzionale
```

- **def**: La parola chiave che indica l'inizio della definizione di una funzione.
- **nome_funzione**: Il nome della funzione, che deve seguire le convenzioni di denominazione di Python (ad esempio, snake_case).
- **parametri**: Valori di input opzionali che la funzione può accettare.
- **:**: I due punti che indicano l'inizio del blocco di codice della funzione.
- **Docstring**: Una stringa di documentazione opzionale che descrive lo scopo della funzione.
- **Corpo della funzione**: Il blocco di codice che esegue l'attività della funzione.
- **return valore_di_ritorno**: Un'istruzione opzionale che restituisce un valore dalla funzione.

Esempi

1. **Procedura, funzione senza parametri e senza valore di ritorno:**

```
def saluta():
    """Stampa un messaggio di saluto."""
    print("Ciao!")

saluta() # Chiamata alla funzione
```

1. **Procedura, funzione con parametri e senza valore di ritorno:**

```
def saluta_nome(nome):
    """Stampa un messaggio di saluto personalizzato."""
    print(f"Ciao, {nome}!")

saluta_nome("Alice") # Chiamata alla funzione con un argomento
```

1. **Funzione con parametri e con valore di ritorno:**

```
def somma(a, b):
    """Restituisce la somma di due numeri."""
    return a + b

risultato = somma(5, 3) # Chiamata alla funzione e assegnazione del
valore di ritorno
print(risultato) # Stampa 8
```

1. **Funzione con parametri opzionali:**

```
def potenza(base, esponente=2):
    """Calcola la potenza di un numero (l'esponente predefinito è 2)."""
    return base ** esponente

print(potenza(3)) # Stampa 9 (3^2)
print(potenza(3, 3)) # Stampa 27 (3^3)
```

Chiamata di una funzione

Per eseguire il codice all'interno di una funzione, è necessario chiamarla utilizzando il suo nome seguito da parentesi tonde. Se la funzione accetta parametri, è necessario passare i valori corrispondenti all'interno delle parentesi.

Funzioni con argomenti variabili

In Python, *args e **kwargs sono usati nelle definizioni di funzione per consentire un numero variabile di argomenti. Funzioni polimorfe

Perché sono utili?

- **Flessibilità**: Permettono di creare funzioni che possono gestire un numero variabile di input, rendendo il codice più adattabile.
- **Codice più pulito**: Invece di dover definire molteplici versioni di una funzione per diversi numeri di argomenti, si può usare *args e **kwargs.
- **Compatibilità**: Sono spesso usati in librerie e framework per permettere agli utenti di passare opzioni aggiuntive alle funzioni.

In Python, *args e **kwargs sono usati nelle definizioni di funzione per consentire un numero variabile di argomenti. Ecco una spiegazione dettagliata:

***args (Argomenti Posizionali Variabili)**

- **Scopo**:
  - *args permette a una funzione di accettare un numero arbitrario di argomenti posizionali.
  - Gli argomenti posizionali sono quelli passati alla funzione senza un nome specifico.
- **Funzionamento**:
  - Quando una funzione è chiamata con *args, tutti gli argomenti posizionali extra vengono raccolti in una tupla.
  - All'interno della funzione, args è una tupla che contiene questi argomenti.
- **Esempio**:

```
def somma(*args):
    risultato = 0
    for numero in args:
        risultato += numero
    return risultato

print(somma(1, 2, 3))  # Output: 6
print(somma(1, 2, 3, 4, 5))  # Output: 15
```

****kwargs (Argomenti con Nome Variabili)**

- **Scopo**:
  - **kwargs consente a una funzione di accettare un numero arbitrario di argomenti con nome (o argomenti "keyword").
  - Gli argomenti con nome sono passati alla funzione utilizzando la sintassi nome=valore.
- **Funzionamento**:
  - Quando una funzione è chiamata con **kwargs, tutti gli argomenti con nome extra vengono raccolti in un dizionario.
  - All'interno della funzione, kwargs è un dizionario che contiene questi argomenti.
- **Esempio**:

```
def stampa_info(**kwargs):
    for chiave, valore in kwargs.items():
        print(f"{chiave}: {valore}")

stampa_info(nome="Alice", eta=30, citta="Roma")
# Output:
# nome: Alice
# eta: 30
# citta: Roma
```

Uso Combinato

È possibile utilizzare sia *args che **kwargs nella stessa definizione di funzione:

```
def funzione_mista(*args, **kwargs):
    print("Argomenti posizionali:", args)
    print("Argomenti con nome:", kwargs)

funzione_mista(1, 2, 3, nome="Bob", professione="Programmatore")
```

```
    # Output:
    # Argomenti posizionali: (1, 2, 3)
    # Argomenti con nome: {'nome': 'Bob', 'professione': 'Programmatore'}
```

Python offre un modo per gestire un numero arbitrario di argomenti in una funzione attraverso simboli speciali.
Questi simboli sono * e ** , e vengono utilizzati davanti ai nomi delle variabili per indicare rispettivamente *args (posizionali) e **kwargs(parole chiavi).
Esempio
*args

```
def add(*args):
   return sum(args)
#add(1,2,3,4)

#10

def supporter(**kwargs):
        for k,v in kwargs.items():
                print(f"{k} tifa per la squadra {v}")
 supporter(davide="Napoli")
```

link per scaricare i file di esempio
https://github.com/albertocesaretti/python--funzioni