**Variable Positional Arguments in Python**

In Python, **variable positional arguments** allow a function to accept an arbitrary number of positional arguments. This is achieved using the \*args syntax.

**Using \*args**

The \*args parameter allows you to pass multiple arguments to a function as a tuple.

**Example:**

def greet(\*names):

for name in names:

print(f"Hello, {name}!")

greet("Alice", "Bob", "Charlie")

**Output:**

Hello, Alice!

Hello, Bob!

Hello, Charlie!

Here, "Alice", "Bob", and "Charlie" are passed as positional arguments and packed into the args tuple.

**Using \*args with Other Parameters**

You can combine \*args with regular parameters.

**Example:**

def introduce(greeting, \*names):

for name in names:

print(f"{greeting}, {name}!")

introduce("Hi", "Alice", "Bob", "Charlie")

**Output:**

Hi, Alice!

Hi, Bob!

Hi, Charlie!

Here, greeting is a normal parameter, while \*names captures additional arguments.

**Using \*args with Keyword Arguments (\*\*kwargs)**

You can combine \*args with \*\*kwargs (which captures keyword arguments as a dictionary).

**Example:**

def display\_info(\*args, \*\*kwargs):

print("Positional arguments:", args)

print("Keyword arguments:", kwargs)

display\_info(1, 2, 3, name="Alice", age=25)

**Output:**

Positional arguments: (1, 2, 3)

Keyword arguments: {'name': 'Alice', 'age': 25}

Here:

* (1, 2, 3) is captured as a tuple in args
* {'name': 'Alice', 'age': 25} is captured as a dictionary in kwargs

**Unpacking Arguments Using \***

You can also use \* to unpack a list or tuple when passing arguments.

**Example:**

def add\_numbers(a, b, c):

return a + b + c

numbers = (1, 2, 3)

print(add\_numbers(\*numbers)) # Unpacks the tuple

**Output:**

6

**Order of Parameters in Function Definitions**

If using multiple types of parameters, they must follow this order:

1. **Regular parameters**
2. **\*args**
3. **Keyword parameters**
4. **\*\*kwargs**

**Example:**

def example\_func(a, b, \*args, x=10, y=20, \*\*kwargs):

print(a, b)

print(args)

print(x, y)

print(kwargs)

example\_func(1, 2, 3, 4, 5, x=100, z=200)

**Output:**

1 2

(3, 4, 5)

100 20

{'z': 200}

**Key Takeaways**

* \*args collects extra **positional** arguments as a tuple.
* It allows functions to accept a **variable number** of arguments.
* It can be combined with normal parameters and \*\*kwargs.
* \*args can also be used for **argument unpacking** when calling functions.