Keyword and optional

In Python, keyword and optional arguments provide a way to make functions more flexible by allowing the caller to specify arguments by name and providing default values for parameters. This enhances the readability and usability of functions, especially when dealing with functions that have a large number of parameters.

Keyword Arguments:

In Python, you can use keyword arguments to explicitly specify which parameter each argument should be assigned to when calling a function. This is particularly useful when a function has multiple parameters, and you want to make the code more readable by specifying the parameter names.

```python
def person_info(name, age, country):
    print(f"Name: {name}, Age: {age}, Country: {country}")

# Using keyword arguments
person_info(age=25, name="Alice", country="USA")
```

In this example, the order of the arguments is different from the order in the function definition, but the use of keyword arguments makes it clear which value corresponds to which parameter.

Default or Optional Arguments:

You can provide default values for parameters in a function. If a value is not provided for a parameter during the function call, the default value is used.

```python
def greet(name="Guest"):
    print(f"Hello, {name}!")

# Calling the function
greet()         # Output: Hello, Guest!
greet("Bob")    # Output: Hello, Bob!
```

In this example, if no argument is provided, the default value "Guest" is used for the `name` parameter. If an argument is provided, it overrides the default value.

Combining Keyword and Optional Arguments:

You can combine keyword and optional arguments to create flexible and readable function calls.

```python
def display_info(name, age=30, city="Unknown"):
    print(f"Name: {name}, Age: {age}, City: {city}")

# Using a combination of keyword and optional arguments
display_info("Alice")                # Output: Name: Alice, Age: 30, City: Unknown
display_info("Bob", city="New York")    # Output: Name: Bob, Age: 30, City: New York
display_info("Charlie", 25)          # Output: Name: Charlie, Age: 25, City: Unknown
```

In this example, the `display_info` function has two optional parameters (`age` and `city`). If these parameters are not provided during the function call, the default values are used. The use of keyword arguments makes it clear which values correspond to which parameters, even if they are not provided in the same order as in the function definition.

Understanding and using keyword and optional arguments can make your functions more versatile and improve the readability of your code. It also allows you to provide sensible default values while still allowing customization when needed.