Notes on Python Functions with Variable Arguments

What are Variable Arguments? In Python, functions can accept a variable number of arguments. There are two types:

- **`*args`: Allows a function to take any number of positional arguments.**
- **`**kwargs`: Allows a function to take any number of keyword arguments.**

==========

`*args` (Positional Variable Arguments)

`*args` allows you to pass any number of positional arguments to a function. These arguments are collected into a tuple.

Syntax:

```
def my_function(*args):
    for arg in args:
        print(arg)
```

Example:

```
def add_numbers(*args):
    return sum(args)

print(add_numbers(1, 2, 3))  # Output: 6
print(add_numbers(4, 5))     # Output: 9
```

==========

`**kwargs` (Keyword Variable Arguments)

`**kwargs` allows you to pass any number of keyword arguments (name-value pairs). These arguments are stored in a dictionary.

Syntax:

```
def my_function(**kwargs):
    for key, value in kwargs.items():
        print(f"{key}: {value}")
```

Example:

```
def greet(**kwargs):
    for key, value in kwargs.items():
        print(f"Hello {key}, your age is {value}")

greet(John=25, Jane=30)
# Output:
# Hello John, your age is 25
# Hello Jane, your age is 30
```

Note 1:

You can combine both **`*args`** and **`**kwargs`** in the same function.

```python
def display_info(*args, **kwargs):
    print("Positional arguments:", args)
    print("Keyword arguments:", kwargs)

display_info(1, 2, 3, name="Zahra", age=25)
# Output:
# Positional arguments: (1, 2, 3)
# Keyword arguments: {'name': 'Zahra', 'age': 25}
```

Note 2:

Order of Arguments

When using **`*args`** and **`**kwargs`**, the order of arguments should be:

1. Regular positional arguments
2. `*args`
3. `**kwargs`

```python
def my_function(a, b, *args, **kwargs):
    print(a, b)
    print(args)
    print(kwargs)
```

Note 3:

Unpacking `*args` and `**kwargs`

You can pass a tuple to **`*args`** or a dictionary to **`**kwargs`** when calling a function, which will automatically unpack them. Example:

```
def show_numbers(*args):
    print(args)

numbers = (1, 2, 3)
show_numbers(*numbers)  # Output: (1, 2, 3)
```

==========

Sample Questions

Question 1:

Write a function called ***`multiply_numbers`*** that accepts any number of positional arguments and returns their product. If no arguments are passed, return 1.

Question 2:

Write a function called ***`user_info`*** that takes a user's first name, last name, and any number of additional keyword arguments (like age, email, etc.). The function should print the first name, last name, and any additional information in a readable format.

==========

Answers

Question 1:

```python
def multiply_numbers(*args):
    result = 1
    for num in args:
        result *= num
    return result

# Test cases
print(multiply_numbers(2, 3, 4))  # Output: 24
print(multiply_numbers())         # Output: 1
```

Question 2:

```python
def user_info(first_name, last_name, **kwargs):
    print(f"First Name: {first_name}")
    print(f"Last Name: {last_name}")
    for key, value in kwargs.items():
        print(f"{key}: {value}")

# Test case
user_info("Zahra", "Doe", age=25, email="zahra@example.com")
# Output:
# First Name: Zahra
# Last Name: Doe
# age: 25
# email: zahra@example.com
```