Function Arguments & Default Values

Function arguments and default values play a crucial role in defining flexible and reusable functions in Python. They allow functions to accept varying inputs and provide default values for parameters when they are not explicitly provided by the caller. Here's an explanation of function arguments and default values:

Function Arguments:

- Function arguments are the values passed to a function when it is called.
- Functions can accept zero or more arguments, depending on how they are defined.
- Arguments are specified within the parentheses `()` of the function definition.

```
def greet(name):
 print("Hello,", name)

greet("Du") # "Du" is passed as an argument to the greet function
```

Positional Arguments:

- Positional arguments are passed to a function based on their position in the function call.
- The order and number of arguments must match the function definition.

```
def add(x, y):
 return x + y

result = add(3, 5) # 3 is assigned to x, 5 is assigned to y
```

Keyword Arguments:

- Keyword arguments are passed to a function using the `key=value` syntax, allowing arguments to be passed in any order.
- Keyword arguments are useful for making function calls more readable and self-explanatory.

```
def greet(name, message):
 print(f"{message}, {name}")
```

```python
greet(message="Welcome", name="Du") # Arguments can be passed in any order
```

Arbitrary Number of Arguments:

- Functions can accept an arbitrary number of arguments using `*args` and `**kwargs` syntax.
- `*args` collects positional arguments into a tuple, and `**kwargs` collects keyword arguments into a dictionary.

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

my_function(1, 2, 3, name="Du", age=30)
```

Default Values:

- Default values can be specified for function parameters, allowing them to be omitted when the function is called.
- Parameters with default values must come after parameters without default values in the function definition.

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

greet("Du") # Output: Hello, Du
```

Mutable Default Values:

- Be cautious when using mutable objects (e.g., lists, dictionaries) as default values, as they are shared across function calls.

```python
def append_to_list(value, lst=[]):
 lst.append(value)
 return lst

print(append_to_list(1)) # Output: [1]
print(append_to_list(2)) # Output: [1, 2]
```

Function arguments and default values provide flexibility and versatility in defining functions that can handle a wide range of inputs while providing sensible defaults when necessary. They are essential tools for writing reusable and maintainable code in Python.

Exercises and Answers for Function Arguments & Default Values

Exercise 1:
Define a function named `calculate_area` that calculates the area of a rectangle. The function should accept two arguments: `length` and `width`, with default values of 5 and 3 respectively. Test the function by calling it with different values for `length` and `width`.

Answer 1:

```python
def calculate_area(length=5, width=3):
 return length * width

# Test the function
print("Area with default values:", calculate_area()) # Output: 15
print("Area with custom values:", calculate_area(7, 4)) # Output: 28
```

Exercise 2:
Create a function called `format_name` that formats a person's name. The function should accept three arguments: `first_name`, `last_name`, and `title`. The default value for `title` should be an empty string. The function should return the formatted name in the format "Title First Name Last Name" if a title is provided, or "First Name Last Name" if no title is provided. Test the function with different combinations of arguments.

Answer 2:

```python
def format_name(first_name, last_name, title=""):
 if title:
 return f"{title} {first_name} {last_name}"
 else:
 return f"{first_name} {last_name}"

# Test the function
print(format_name("Du", "Wun")) # Output: Du Wun
print(format_name("Htet", "Arkar", "Mr.")) # Output: Mr. Htet Arkar
```

Exercise 3:
Write a function called `calculate_cost` to calculate the total cost of items in a shopping cart. The function should accept two arguments: `items` (a list of item prices) and `discount` (discount percentage, default value 0). The function should return the total cost after applying the discount. Test the function with different lists of item prices and discount values.

Answer 3:

```
def calculate_cost(items, discount=0):
 total_cost = sum(items)
 discounted_cost = total_cost - (total_cost * (discount / 100))
 return discounted_cost

# Test the function
cart_items = [10, 20, 30, 40]
print("Total cost without discount:", calculate_cost(cart_items)) # Output: 100
print("Total cost with 10% discount:", calculate_cost(cart_items, 10)) # Output: 90
```

Exercise 4:
Create a function called `display_info` that prints information about a person. The function should accept four arguments: `name`, `age`, `city`, and `country`. The default values for `age`, `city`, and `country` should be None. If any of these values are not provided, the function should print "Information not available" for that value. Test the function with different combinations of arguments.

Answer 4:

```
def display_info(name, age=None, city=None, country=None):
 print("Name:", name)
 if age is not None:
 print("Age:", age)
 else:
 print("Age: Information not available")
 if city:
 print("City:", city)
 else:
 print("City: Information not available")
 if country:
 print("Country:", country)
 else:
 print("Country: Information not available")

# Test the function
display_info("Du", 31, "Yangon", "Myanmar")
display_info("May", city="Yangon")
```

Exercise 5:
Write a function called `calculate_score` that calculates the total score based on scores obtained in three subjects. The function should accept three arguments: `subject1`, `subject2`, and `subject3`, with default values of 0. The function should return the sum of the scores. Test the function with different scores.

Answer 5:

```python
def calculate_score(subject1=0, subject2=0, subject3=0):
 return subject1 + subject2 + subject3

# Test the function
print("Total score:", calculate_score(85, 90, 75)) # Output: 250
print("Total score with default values:", calculate_score()) # Output: 0
```