Defining your own functions

Defining your own functions in Python is a fundamental aspect of programming. Functions allow you to break down your code into modular and reusable components, making your code more organized and easier to understand. Here's a guide to defining your own functions in Python:

Function Syntax:

In Python, a function is defined using the `def` keyword, followed by the function name, a pair of parentheses `()`, and a colon `:`. The function body, where the code inside the function resides, is indented.

```python
def my_function():
    # Function body
    print("Hello, I'm a function!")

# Calling the function
my_function()
```

Parameters and Arguments:

You can pass values to functions using parameters. Parameters are specified in the function definition within the parentheses. When you call the function, you provide values for these parameters, which are called arguments.

```python
```

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

# Calling the function with an argument
greet("Alice")
```

In this example, `name` is a parameter of the `greet` function, and when the function is called with `greet("Alice")`, "Alice" is an argument passed to the `name` parameter.

Return Statement:

Functions can return values using the `return` statement. This allows the function to produce a result that can be used elsewhere in your code.

```python
def add_numbers(a, b):
    return a + b

result = add_numbers(3, 5)
print(result)  # Output: 8
```

Default Parameters:

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

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

greet()         # Output: Hello, Guest!
greet("Alice")   # Output: Hello, Alice!
```

Docstrings:

It's good practice to include a docstring, a triple-quoted string, at the beginning of your function to describe its purpose and usage.

```python
def multiply(a, b):
    """
    Multiply two numbers.

    Parameters:
    - a (int): The first number.
    - b (int): The second number.

    Returns:
    int: The product of a and b.
    """
```

```
    return a * b
```

Scope:

Variables defined inside a function have local scope, meaning they are only accessible within that function. Variables defined outside any function have global scope and can be accessed throughout the program.

```python
global_variable = 10

def my_function():
    local_variable = 5
    print(global_variable)  # Accessing global variable
    print(local_variable)   # Accessing local variable

my_function()
print(global_variable)      # Still accessible
# print(local_variable)     # This would result in an error; local_variable is not accessible outside the function
```

By understanding and utilizing these concepts, you can create well-structured and reusable code in Python through the definition of your own functions.