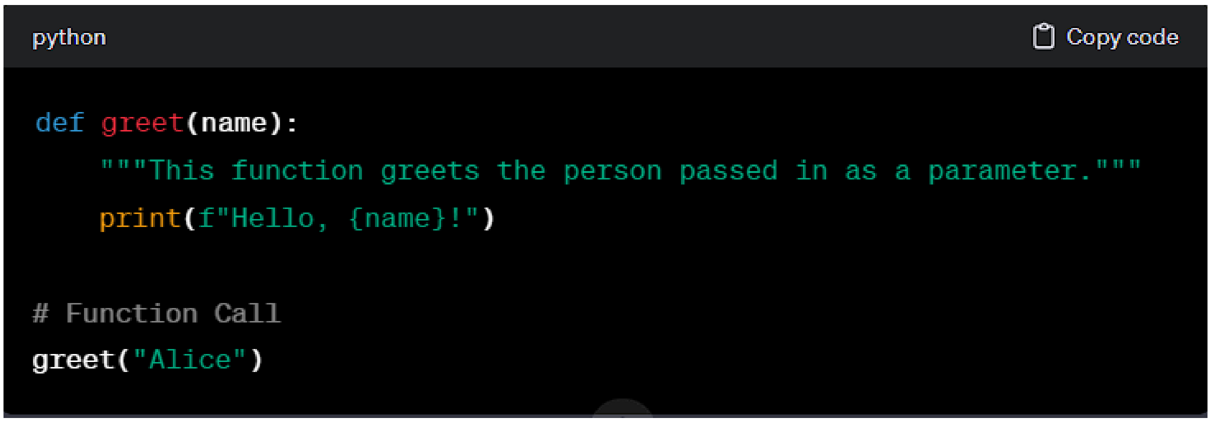
Functions

a function is a block of reusable code that performs a specific task. Functions help in organizing code, promoting code reusability, and making the code more readable. Functions are defined using the `def` keyword, followed by the function name, parameters, and a block of code. Here are the key aspects of functions in Python:

1. Function Definition:

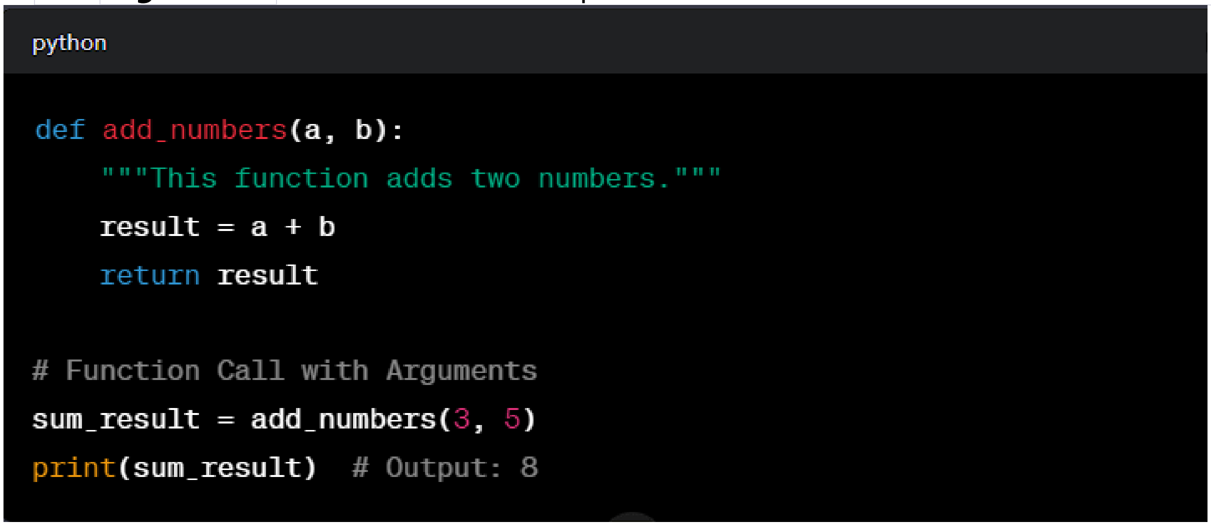


```python
def greet(name):
    """This function greets the person passed in as a parameter."""
    print(f"Hello, {name}!")

# Function Call
greet("Alice")
```

2. Parameters and Arguments:

- **Parameters:** These are the variables listed in the function definition. They act as placeholders for values that will be passed to the function.
- **Arguments:** These are the values passed to the function when it is called.

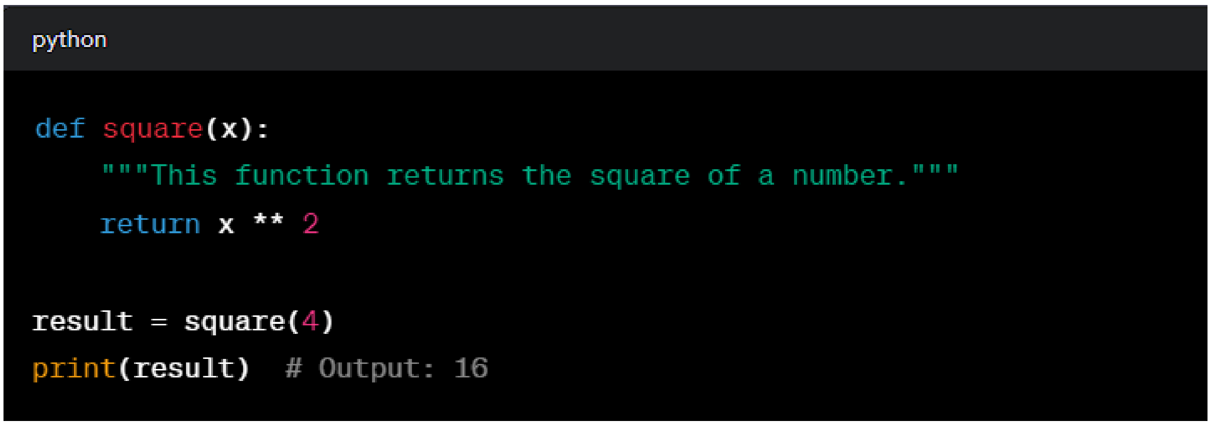


```python
def add_numbers(a, b):
    """This function adds two numbers."""
    result = a + b
    return result

# Function Call with Arguments
sum_result = add_numbers(3, 5)
print(sum_result)  # Output: 8
```

3. Return Statement:

A function can return a value using the `return` keyword. If there is no `return` statement, the function returns `None` by default.

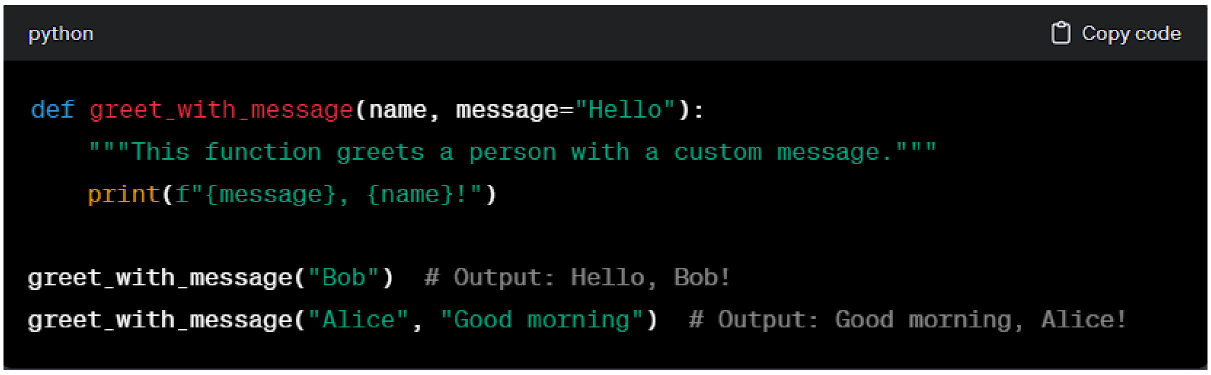


```python
def square(x):
    """This function returns the square of a number."""
    return x ** 2

result = square(4)
print(result)  # Output: 16
```

4. Default Parameters:

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

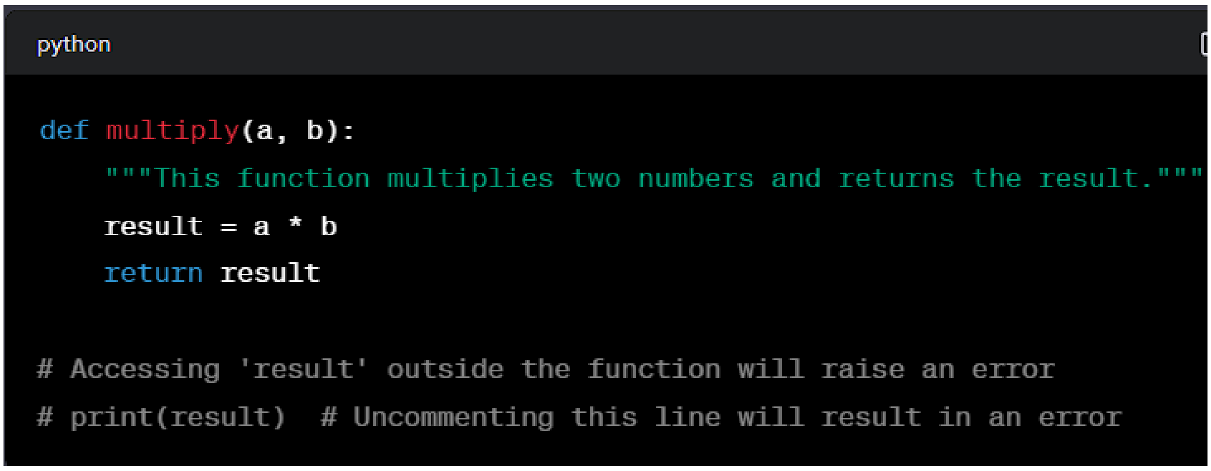


```python
def greet_with_message(name, message="Hello"):
    """This function greets a person with a custom message."""
    print(f"{message}, {name}!")

greet_with_message("Bob")  # Output: Hello, Bob!
greet_with_message("Alice", "Good morning")  # Output: Good morning, Alice!
```

5. Variable Scope:

Variables defined inside a function are local to that function and are not accessible outside of it, unless explicitly returned.



```python
def multiply(a, b):
    """This function multiplies two numbers and returns the result."""
    result = a * b
    return result

# Accessing 'result' outside the function will raise an error
# print(result)  # Uncommenting this line will result in an error
```

6. *args and **kwargs:

These allow a function to accept any number of positional and keyword arguments.

python

```python
def print_args(*args, **kwargs):
    """This function prints positional and keyword arguments."""
    print("Positional arguments:", args)
    print("Keyword arguments:", kwargs)

print_args(1, 2, 3, name="Alice", age=30)
```

7. Lambda Functions:

Lambda functions are anonymous functions defined using the `lambda` keyword. They are often used for short, simple operations.

python

```python
multiply = lambda x, y: x * y
print(multiply(3, 4))  # Output: 12
```

8. Recursion:

A function can call itself, leading to recursive function calls.

python Copy code

```python
def factorial(n):
    """This function calculates the factorial of a number using recursion."""
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)

print(factorial(5))  # Output: 120
```

These are fundamental concepts and examples of functions in Python. Functions play a crucial role in structuring code and promoting code reusability.