## Functions

Functions in programming allow for the reuse of code that needs to be executed multiple times with slight variations. Instead of repeating the code, a function can be defined to perform the operation. Functions can take zero or more inputs, known as parameters, allowing for flexibility in their behavior. This helps to streamline the code and conserve memory.

## Functions Fundamentals

Creating Functions:

Functions are defined with a name, parameters, and optional code block.

**def greet(name):**

**print(f"Hello, {name}!")**

Argument vs Parameter:

Argument: Value passed to a function during a call. Parameter: Variable defined in the function.

**def greet(name):** # 'name' is a parameter

**print(f"Hello, {name}!")**

**greet("Alice")** # 'Alice' is an argument

Return Statement:

Indicates a function's result or can be used to exit.

**def add(x, y):**

**return x + y**

Function Types:

Ordinary functions perform calculations and return a result.

**def square(x):**

**return x \*\* 2**

Procedure functions execute a procedure and may not return a result.

Factory Functions:

Generate values without taking parameters.

**def generate\_random():**

**import random**

**return random.randint(1, 100)**

Default Parameters:

Assigns a default value if no specific value is provided.

**def greet(name="Guest"):**

**print(f"Hello, {name}!")**

Scope of Variables:

Variables inside a function are local and not accessible outside.

**def multiply(x, y):**

**result = x \* y** # 'result' is local to the function return result

Returning Values:

Use **return** to make a local variable's value available outside.

**def calculate\_tax(salary):**

**tax = salary \* 0.2**

**return tax**

Function Call:

Functions need to be defined before they can be called.

**def say\_hello():**

**print("Hello!")**

**say\_hello()**

## Random Module

## Recursive Functions