**Assignment – Functions**

1. What is the difference between a function and a method in Python?

Ans :- In Python, a function and a method are both blocks of code that can be executed, but they differ in their binding and usage:

Functions

- A function is a standalone block of code that can be called multiple times from different parts of a program.

- Functions are not bound to any specific class or object.

- Functions are defined using the def keyword.

Methods

- A method is a block of code that is bound to a specific class or object.

- Methods are used to perform actions on an object or to modify its state.

- Methods are defined inside a class definition and are bound to that class.

Here are three examples to illustrate the difference:

Example 1: A Simple Function

def greet(name):

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

greet("John") # Output: Hello, John!

In this example, greet is a standalone function that takes a name parameter and prints a greeting message.

Example 2: A Method in a Class

class Person:

def \_\_init\_\_(self, name):

self.name = name

def greet(self):

print(f"Hello, my name is {self.name}!")

person = Person("Jane")

person.greet() # Output: Hello, my name is Jane!

In this example, greet is a method bound to the Person class. It takes no parameters (except self, which refers to the instance of the class) and prints a greeting message.

Example 3: A Static Method in a Class

class MathUtils:

@staticmethod

def add(a, b):

return a + b

result = MathUtils.add(2, 3)

print(result) # Output: 5

In this example, add is a static method bound to the MathUtils class. It takes two parameters, a and b, and returns their sum. Static methods are not bound to any specific instance of the class and can be called without creating an instance.

1. Explain the concept of function arguments and parameters in Python.

Ans :- In Python, functions can take arguments, which are values passed to the function when it's called. These arguments are assigned to parameters, which are variables defined inside the function.

Parameters:

Parameters are the variables defined in the function definition. They are the names given to the arguments that will be passed to the function. Parameters are defined inside the parentheses of the function definition.

Example:

def greet(name, age):

print(f"Hello, {name}! You are {age} years old.")

In this example, name and age are parameters.

Arguments:

Arguments are the values passed to the function when it's called. Arguments can be literals, variables, or expressions. When a function is called, the arguments are assigned to the corresponding parameters.

Example:

greet("John", 30)

In this example, "John" and 30 are arguments.

Types of Parameters:

Python supports several types of parameters:

1. Positional Parameters: These are parameters that are defined without a default value. They must be passed in the correct order when calling the function.

2. Keyword Parameters: These are parameters that are defined with a default value. They can be passed in any order when calling the function, as long as the parameter name is specified.

3. Default Parameters: These are parameters that have a default value assigned to them. If no argument is passed for these parameters, the default value is used.

4. Variable-Length Parameters: These are parameters that can accept a variable number of arguments. They are defined using the \* or \*\* syntax.

Example:

def greet(name, age=30, \*args, \*\*kwargs):

print(f"Hello, {name}! You are {age} years old.")

print("Additional arguments:")

for arg in args:

print(arg)

for key, value in kwargs.items():

print(f"{key}: {value}")

In this example, name is a positional parameter, age is a keyword parameter with a default value, \*args is a variable-length parameter that accepts a variable number of positional arguments, and \*\*kwargs is a variable-length parameter that accepts a variable number of keyword arguments.