**Assignment - 3**

1. Why are functions advantageous to have in your programs?

Ans: Functions bring several advantages to your programs:

1. Reusability: Write code once, use it multiple times. No need to repeat the same logic, saving time and effort.

2. Modularity: Break down complex programs into smaller, independent functions, making the code easier to understand, maintain, and modify.

3. Readability: Improve code organization and clarity by separating concerns and focusing on individual function purposes.

4. Maintainability: Isolate and modify specific code sections without affecting others, simplifying maintenance and reducing the risk of errors.

5. Debugging: Focus on smaller, well-defined functions to pinpoint and fix issues more efficiently.

In essence, functions promote cleaner, more efficient, and maintainable code, making them crucial for any programmer

1. When does the code in a function run: when it's specified or when it's called?

Ans: The code within a function runs only when the function is called, not when it's initially defined.

1. What statement creates a function?

Ans: ‘def’

1. What is the difference between a function and a function call?

Ans: The key difference between a function and a function call lies in their action and purpose:

Function: A function is a reusable block of code that defines a specific task or set of instructions. It acts like a blueprint, providing the structure and logic for performing the task. Think of it as a recipe waiting to be used.

Function call: A function call is the activation of the function. It instructs the program to execute the code defined within the function, similar to following the recipe's instructions. It provides any necessary inputs (arguments) to the function and receives the output (return value), if any.

Here's an analogy: Imagine a function as a tool in your toolbox (e.g., a hammer). Defining the function is like creating the tool itself, while calling the function is like picking up the hammer and using it to complete a task (e.g., hammering a nail).

1. How many global scopes are there in a Python program? How many local scopes?

Ans: A Python program has:

One global scope: This scope encompasses the entire program execution, existing from the start until the program terminates. It holds variables and names defined directly at the top level of the program, outside of any functions.

Multiple local scopes: Each function definition creates its own local scope, which exists only during the execution of that specific function. Local variables and names defined within a function are only accessible within that function's scope.

In essence, the global scope provides a program-wide space, while each function has its own isolated local scope to manage its internal variables.

1. What happens to variables in a local scope when the function call returns?

Ans: When a function call in Python returns, the local variables within that function's scope cease to exist. This means:

* The memory allocated for the local variables is freed by the program.
* The variables become inaccessible from any other part of the program, including other functions or the global scope.

This is because local variables are created on the stack, a temporary memory space that is managed based on function calls. As a function call ends and returns, the stack frame associated with that function is removed, and all local variables within it are destroyed.

1. What is the concept of a return value? Is it possible to have a return value in an expression?

Ans:

A return value is the data or object that a function sends back to the caller when it finishes execution. It allows functions to communicate results or provide outputs to the code that called them.

While expressions cannot directly hold return values, these techniques allow you to effectively use function outputs within larger calculations or manipulations.

1. If a function does not have a return statement, what is the return value of a call to that function?

Ans: In Python, if a function doesn't have an explicit return statement, the function will implicitly return the special value None. This is true even if the function body contains expressions or other statements.

1. How do you make a function variable refer to the global variable?

Ans: Using the global keyword:

global\_var = 10

def modify\_global():

global global\_var # Declare using `global` keyword

global\_var += 5

print(f"Inside function: {global\_var}")

modify\_global()

print(f"Outside function: {global\_var}")

1. What is the data type of None?

Ans: NoneType

1. What does the sentence import areallyourpetsnamederic do?

Ans: Nothing, it will error

1. If you had a bacon() feature in a spam module, what would you call it after importing spam?

Ans:

import spam

bacon\_result = spam.bacon()

1. What can you do to save a programme from crashing if it encounters an error?

Ans: Try-Except blocks: This is a fundamental mechanism for error handling in Python. It allows you to define a block of code (try) that might raise an error and a separate block (except) to handle the error gracefully.

1. What is the purpose of the try clause? What is the purpose of the except clause?

Ans: 1. Try Clause:

The try clause is the foundation of error handling. It encompasses the code block that you anticipate might raise an exception. This code can be any valid Python statements, function calls, calculations, or operations.

If the code within the try clause executes without encountering any exceptions, the program continues normally, and the except clause is skipped.

2. Except Clause:

The except clause is where you define how to handle exceptions that might occur during the execution of the code within the try block.

There are multiple ways to use the except clause:

Bare except: This catches all exceptions raised within the try block, allowing you to perform a general action like printing an error message or logging the issue. However, it's generally discouraged as it can mask specific errors.

Specific exception type: You can specify the type of exception you want to catch within the except clause (e.g., except ZeroDivisionError). This allows for targeted handling of different types of errors, providing more informative and appropriate responses.