# Assignment 3:

## Q1

Functions are advantageous to have in our programs for several reasons, especially for beginners. Here are some simple explanations:

Reuse of code: Functions allow us to write a block of code once and reuse it multiple times in our program. Instead of repeating the same code over and over, we can define a function and call it whenever we need that specific functionality.

Modularity: Functions help in organising wer code into logical and manageable chunks. Each function can perform a specific task, making the program easier to understand and maintain. We can focus on writing small, self-contained functions that work together to solve a larger problem.

Readability: By breaking wer code into functions, we can give meaningful names to each function, making wer code more readable and self-explanatory. Functions also help in dividing complex tasks into smaller, manageable parts, making the overall code easier to understand.

Code organisation: Functions help in organising wer code by separating different functionalities into modular units. This makes it easier to navigate through wer codebase, locate specific functionality, and make changes or improvements without affecting other parts of the program.

Code reusability: Functions can be reused in different programs or projects. Once we write a useful function, we can easily import it into other programs, saving time and effort by leveraging our existing code.

Overall, functions provide structure, reusability, and simplicity to wer programs. They promote code reuse, improve readability, and make it easier to manage and maintain wer codebase. Functions are a fundamental building block in programming and play a vital role in creating efficient, organised, and scalable programs.

## Q2

The code inside a function runs when the function is called, not when it is specified or defined.

## Q3

In Python, the def statement is used to create a function. The def keyword stands for "define" and is followed by the name of the function, a pair of parentheses ().

## Q4

The difference between a function and a function call lies in their respective roles and actions within a program.

Function: A function is a block of reusable code that performs a specific task or a set of tasks. It is defined using the def statement and has a name, optional parameters, and a body of code. Functions allow us to organise our code by encapsulating specific functionality.

Function Call: A function call is the act of calling or executing a function. It is the point in the program where the function's code is actually executed. When a function is called, the program flow transfers to the function, executes the code inside the function body, and then returns to the point immediately after the function call.

## Q5

In a Python program, there is typically one global scope and multiple local scopes.

Global Scope: The global scope refers to the outermost level of the program. It is the scope in which variables, functions, and classes defined outside any function or class are accessible. The global scope exists throughout the entire program, and any variables defined in this scope can be accessed from any part of the program.

Local Scopes: Local scopes are created when a function is called or when a block of code, such as a loop or conditional statement, is executed. Each function or block of code creates its own local scope. Local scopes are temporary and exist only within the context of the function or block of code in which they are created. Variables defined in a local scope are accessible only within that scope and are not accessible outside of it.

## Q6

When a function call returns, the local variables defined within that function's local scope cease to exist. They are destroyed, and their values are no longer accessible.

In Python, each time a function is called, a new local scope is created for that function. Any variables created within that local scope, including parameters and local variables defined within the function, only exist within the context of that function call.

## Q7

In Python, the return is used in a function to specify the value that should be sent back or "returned" to the code that called the function. It marks the end of the function's execution and sends the specified value back as the result of the function call.

A return value is not an expression itself, but it can be used as part of an expression in the calling code. For example, we can directly use the return value in calculations or assign it to a variable for later use.

## Q8

If a function does not have a return statement, the return value of a call to that function is None.

## Q9

**Using the global keyword:** The global keyword tells Python that we want to refer to the global variable, even if there is a local variable with the same name.

## Q10

In Python, the data type of None is NoneType. None is a special value that represents the absence of a value or the lack of a specific value.

## Q11

The sentence import areallwerpetsnamederic imports a module named areallwerpetsnamederic. This module does not exist, so the import statement will fail.

## Q12

spam.bacon()

## Q13

To save a program from crashing when it encounters an error, we can use error handling techniques to catch and handle the errors. In Python, this is achieved using try and except blocks.

## Q14

The purpose of the try clause in Python is to enclose a block of code that may potentially raise an error. It allows us to specify code that might cause an exception and provides a structured way to handle potential errors. If an error occurs within the try block, the execution of the code within the try block is immediately stopped, and the program jumps to the corresponding exception. The purpose of the except clause is to define how to handle specific types of errors that may occur within the associated try block. It allows us to catch and handle specific exceptions, providing alternative code or actions to be executed when a particular error occurs.