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

Functions are advantageous to have in your programs for several reasons:

Modularity and Reusability, Abstraction, Readability, Debugging and Testing, Maintainability, Collaboration, Collaboration, Code Organization, Parameterization, Error Handling, Error Handling, Performance Optimization, Code Documentation.

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

When you define a function in your code, you are essentially creating a named block of code with a specific set of instructions. These instructions are not executed until you explicitly call the function in your program.

def my\_function():

print("This code runs when the function is called.")

my\_function()

3. What statement creates a function?

def function\_name(parameters):

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

Function:

A function is a named, reusable block of code that performs a specific task or set of tasks.

Function Call:

A function call is an action that triggers the execution of a specific function.

When you call a function, you are instructing the program to execute the code defined within that function.

# Function definition

def greet(name):

"""This function greets the user."""

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

# Function call

greet("Alice")

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

global\_variable = 42 # This is in the global scope

def my\_function():

print(global\_variable) # Accessing the global variable

my\_function() # Calling the function

global\_variable = 42 # This is in the global scope

def my\_function():

local\_variable = 10 # This is in the local scope of the function

print(global\_variable) # Accessing the global variable

print(local\_variable) # Accessing the local variable

my\_function() # Calling the function

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

def my\_function():

local\_variable = 10

print(local\_variable) # This variable is accessible within the function

my\_function() # Call the function

# After the function call returns, 'local\_variable' goes out of scope and becomes inaccessible

# Attempting to access 'local\_variable' outside of the function will result in a NameError

# print(local\_variable) # Uncommenting this line would result in a NameError

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

def add\_numbers(a, b):

result = a + b

return result # Returns the 'result' value

total = add\_numbers(5, 3) # Call the function and store the returned value in 'total'

print(total) # Prints the value returned by the function

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

def say\_hello(name):

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

result = say\_hello("Alice")

print(result)

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

global\_variable = 42 # This is a global variable

def modify\_global\_variable():

global global\_variable # Declare that we want to use the global variable

global\_variable = 99 # Modify the global variable inside the function

# Before calling the function

print(global\_variable) # Prints 42

# Call the function to modify the global variable

modify\_global\_variable()

# After calling the function

print(global\_variable) # Prints 99, as the global variable has been modified by the function

10. What is the data type of None?

In Python, None is a special value that represents the absence of a value or the lack of a meaningful result. It is often used to indicate that a variable or expression has no assigned value. None itself is a unique data type called NoneType. You can check the data type of None using the type() function:

result = None

print(type(result)) # Prints "<class 'NoneType'>"

11. What does the sentence import areallyourpetsnamederic do?

The import statement in Python is used to bring external modules or libraries into your code so that you can use their functions, classes, or variables. The module name following the import keyword should refer to an existing module or package installed on system.

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

import spam # Import the 'spam' module

# Call the 'bacon()' function from the 'spam' module

spam.bacon()

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

try:

# Code that might raise a specific exception

except SpecificException as e:

# Handle the specific exception

except AnotherException as e:

# Handle another specific exception

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

Purpose of the try Clause:

The try clause is used to enclose the block of code where you anticipate that an exception or error might occur during execution.

It allows you to specify a section of code that is potentially problematic or might raise exceptions.

The purpose of the try clause is to "try" to execute the code within it. If an exception occurs within the try block, the normal flow of the program is interrupted, and Python looks for an except block that can handle the exception.

Purpose of the except Clause:

The except clause is used to specify how to handle exceptions that may have been raised within the corresponding try block.

When an exception occurs in the try block, Python searches for an except block with a matching exception type (or a more general exception type if no exact match is found).

The purpose of the except clause is to define the actions to be taken when a specific exception is encountered, such as logging an error, displaying a message to the user, or taking corrective actions.