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

Ans-

Functions are advantageous in programs because they promote code reusability, modularity, and abstraction. They make your code more readable, maintainable, and organized, simplifying testing and debugging. Functions encapsulate specific tasks, making it easier to understand and modify code. They also facilitate collaboration and code reuse across projects. With functions, you can break down complex programs into smaller, manageable units, improving overall code structure and efficiency.

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

Ans- The function runs when it is called.

3. What statement creates a function?

In Python, we use the ‘def’ keyword to create the function

```

def function\_name:

pass

```

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

Ans-

A function is a named block of code that defines a specific task or behavior.

A function call is the actual invocation of a function, using its name and parentheses, to execute the code within the function.

```

# Function

def nameformat(fName):

print(f”Hello {fName} “)

# Function Call

nameformat(Pranjal)

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

Ans- There is one global scope in a Python program. Each function has its own local scope.

Local scopes can also be created within control flow blocks. Variables defined in a scope can only be accessed within that scope unless explicitly passed or returned. There are many local scopes.

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

Ans- When the function returns the local variables are getting destroyed.

7. 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 result that a function sends back to the caller after executing its tasks. It allows functions to provide output or pass information back to the code that called them. When a function is called, it can perform operations and calculations and then use the return statement to send a specific value or object back to the caller. The caller can capture this return value and use it in further expressions or assignments. So yes, it is possible to have a return value in an expression, as the returned value can be used directly in other parts of the code for further processing.

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

Ans- If a function does not have a return statement, or if the return statement is missing, the function will implicitly return None. None is a special value in Python that represents the absence of a value.

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

Ans- When you want to access or modify a global variable within a function, you need to declare it as global using the global keyword.

10. What is the data type of None?

Ans- The data type of None is called NoneType. NoneType is a built-in data type in Python that represents the absence of a value or the lack of a return value.

11. What does the sentence import areallyourpetsnamederic do?

Ans- It would raise a ModuleNotFoundError because the module named "areallyourpetsnamederic" does not exist in the Python.

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

Ans- After importing the spam module in Python, you can access the bacon() feature by using the module name followed by the function name separated by a dot.

```

Import spam

spam.bacon()

```

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

Ans- In Python we can use error-handling mechanisms such as try-except blocks. Here's how it works:

1. Identify the portion of the code that may potentially raise an error.

b. Surround that portion of the code with a try block.

c. Immediately following the try block, include an except block that specifies the type of error you want to catch.

d . Inside the except block, provide a fallback solution or alternative code to handle the error gracefully.

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

Ans- the try and except clauses work together to implement error handling and exception handling mechanisms. Here's the purpose of each clause:

A). Try Clause: The purpose of the try clause is to enclose a block of code that may potentially raise an error or an exception. The code within the try block is executed, and if an error occurs during its execution, the flow of control immediately transfers to the appropriate except block.

b). Except Clause: The purpose of the except clause is to specify the type of error or exception that you want to catch and handle. If an error occurs within the try block that matches the specified exception type, the corresponding except block is executed. Inside the except block, you can provide alternative code or error handling logic to gracefully respond to the encountered error. If no matching exception type is found, the program will terminate with an unhandled exception error.