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

Functions are self-contained block of code to perform frequently related operation. Following are the advantages:

* Reducing duplication of code
* Decomposing complex problems into simpler pieces
* Improving clarity of the code
* Reuse of code
* Information hiding

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

The code in a function runs only when it is called. The function definition starts with the def keyword, name of the function, optional parameters and the function body. It may or may not return any value. The default return vale is None

Therefore, a function is specified in the function definition and runs when it is called.

**3. What statement creates a function?**

The statement containing the keyword def create the function. The syntax of the function is as follows:

def <function name> (list of parameter(s)):

statement(s)

return <optional return values(s) >

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

By function we mean function definition and by function call we mean invoking / executing the function.

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

There’s only one global Python scope per program execution. This scope remains in existence until the program terminates and all its names are forgotten. Otherwise, the next time when we run the program, the names would remember their values from the previous run.

Internally, Python turns our program’s main script into a module called \_\_main\_\_ to hold the main program’s execution. The namespace of this module is the main global scope of our program.

The **local scope** or function scope is a Python scope created at function calls. Every time we call a function, we are also creating a new local scope. On the other hand, we can think of each def statement and lambda expression as a blueprint for new local scopes. These local scopes will come into existence whenever we call the function at hand. Thus we can have as many number of local scopes in a program which will be equal to the number of functions defined in the program.

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

The scope of variables in the functions is restricted within the function itself and hence upon return of function call, it is no longer accessible to scope of the calling function.

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

The importance of return statement is to pass values from the function to the calling program. In python we may pass multiple return values. The calling program may use variables or placeholders to access / ignore certain values returned by the functions

Yes we may pass an expression to a return statement. The expression is evaluated first and then the value is returned from the called function. Ex.

|  |  |
| --- | --- |
| **Code** | **Output** |
| def fun(a ,b ):  return (a + b)  fun( 2,3 ) | 5 |

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

None

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

By using the global keyword

Ex.

|  |  |
| --- | --- |
| **Code** | **Output** |
| a = 10  def func1():  global a  a = 20  print("Inside func1 a = ", a)  func1()  print("Outside func1 a = ", a) | Inside func1 a = 20  Outside func1 a = 20 |

**10. What is the data type of None?**

NoneType

**11. What does the sentence import areallyourpetsnamederic do?**

It imports the module named areallyourpetsnamederic.

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

spam.bacon()

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

In order to save a program from crashing, we can use exception handling mechanism available with python. Here we can write the code which causes crashing into the try block and can handle it (once the exception causes) in the except block.

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

The try and except block in Python is used to catch and handle exceptions respectively. Python executes code following the try statement as a “normal” part of the program. The code that follows the except statement is the program’s response to any exceptions in the preceding try clause.

|  |  |
| --- | --- |
| **Code** | **Output** |
| a = 2.5  b = 0  try:  c = a/b  print(c)  except:  print("Exception : ") | Exception |