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

The main advantage of writing functions is reusability of code. We can write a piece of code within a function and call it as many numbers of times as needed without having to rewrite the entire code every time.

Another advantage is that the program is better organized and legible when the code is broken into smaller programs.

Some examples of in-built python functions are print(), append(), pop(), extend() etc.

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

The code in the function runs when the function is called. To call a function we write the name of the function and pass the required parameters.

Ex: print(“Hello World”)

3. What statement creates a function?

Syntax for function creation is:

def FunctionName():

statement1

Function body

statement2

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

A function is a block of code that performs a certain operation and returns the result of that operation. It may or may not take parameters. A function call is used to pass control to the function.

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

**Global scope:** It contains the names that are defined at the top level of the program.  
The names or variables that we define in the global scope are accessible to the entire program. **There is one global scope per python program execution. The scope remains until the program is terminated.**

**Local scope:** It is the body of the python function. It contains variables or names that we define inside the function and these variables are available only within the body of the function. **Each function call will result in local scope being created, so that there are as many local scopes as function calls.**

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

When the function call returns the variables in the local scope are **destroyed.**

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

In python the **return** statement is used inside the function to return the function’s result back to the caller. Syntax is the return keyword followed by the value it is returning. The returning value can be a numeric value (int, float), collections (list, dict, set etc.) or any python object.

The return value is optional. We can also use the return statement without any value or we can exclude the entire return statement from the function. In both the cases the return value will be None. It is possible to use the returned value in any expression like a mathematical expression.

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

In this case the return value of such a function will be None which is equivalent to null value or no value at all. Data type of None is NoneType.

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

We can use the global keyword to create a global variable inside a function.

10. What is the data type of None?

Data type of None is **NoneType**

11. What does the sentence import areallyourpetsnamederic do?

The statement **import areallyourpetsnamederic** lets us incorporate the features of the module areallyourpetsnamederic into our program. For example, if the module areallyourpetsnamederic has a function named sayHello(name) and if we wish to call this function from our python program, we can do so by incorporating the module areallyourpetsnamederic in our program.

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

To call bacon() function in our program, we import spam module first and then access bacon() function as spam.bacon()

Import spam

spam.bacon()

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

Whenever we encounter an error or an exception in the program, to save our programme from crashing, we can perform exception handling. This is done by incorporating try, except and finally (optional) block in our program.

The try block allows us to test the block of code for errors and except block lets us handle those errors.

try:

x = int(input(“Enter the number”))

y = 100/x

except ZeroDivisionError:

print(“Denominator cannot be zero”)

except ValueError:

print(“Please enter correct int value”)

finally:

print(“Finally block is optional. It usually contains imp code like DB connection closing code”)

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

Try block allows us to test our block of code for runtime errors or exception. The except block allows us to handle those errors.

Example:

try:

x = int(input(“Enter the number”))

y = 100/x

except ZeroDivisionError:

print(“Denominator cannot be zero”)

except ValueError:

print(“Please enter correct int value”)

finally:

print(“Finally block is optional. It usually contains imp code like DB connection closing code”)

The error handling is done as follows:

1. When the code inside the try block is successfully executed, the except clauses are bypassed and finally block is executed.

2. If the code inside the try block encounters an error, the except block corresponding to the error encountered is executed. If the code fails due to ValueError, then code in the except ValueError block is executed. If the code fails due to ZeroDivisionError, then code in the except ZeroDivisionError block is executed.

3. Regardless of error or no error in the code, the finally block always executes. IT should contain important code that must be executed whether an exception occurs or not. For example, code to close DB connection.