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

Answer –

- Code reusability - We can write the repetitive lines of code inside a function. Whenever that set of code is required we can call the function and get our job done.
- Better code organization - We can write a related set of operations inside a particular function. This way the code looks much more organized and clean.
- Testing and debugging- With the use of functions it becomes easy to debug the program and problem identification becomes little easy. Testing also becomes easy as we can test individual functions instead of testing the complete program flow.
- Easy to read - We can write a related set of operations inside a function and we can give it a meaning name in english. This way a person reading a program understands what the lines of code are doing by just looking at the function name.

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

Answer – Code in a function is run when we call the function. When we define a function, we are just specifying what it's intended to do.

Q 3. What statement creates a function?

Answer – *def* keyword is used to create a function

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

Answer –

Function - A function is a sequence of statements that take some input (in the form of parameters) and performs some operations and completes the defined task or returns some value. A function has a proper name which gives a glimpse of what the line of code would be doing.

Function Call - Function call is the time when we feel the need of function and we invoke it by giving it proper inputs (if required)

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

Answer –

Global Scopes -

- **Global scope** - A variable with global scope can be utilized anywhere inside the program. We can use that variable in any of the functions and sub functions.
- **Built in scope** - A variable with built in scope is the one, which is defined in one of the modules that we are loading in our program. Example - We import pi from math module and we never define its value. But, its value in build in module and the name pi cannot be used as an identifier in the program.

  from math import pi
  print(pi)

Local scopes -

- **Local scope** - A variable has a local scope and it can be only used inside a defined function or its sub function.
- **Enclosing scope -** A variable defined in the sub function with keyword *nonlocal*. If we change the nonlocal variable as defined in the sub function, then changes are reflected in the parent function.

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

Answer – The variable is removed from the RAM and it cannot be used further.

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

Answer – return value is basically the output of the function. When we call a function we can assign the value to a variable and use it later in the program. Yes, it is possible to have return value in an expression, we can use the return value of function in an expression and get the desired result..

Q8. If a function does not have a return statement, what is the return value of a call to that function?

Answer – When there is no return statement, the function simply executes the line of codes inside the function and the program flow returns to the statement from where the function was called.

If the function call was made from an expression then program execution will end up in error as there is no value to compute upon.

```
def funs():
  x=1

y = funs() +4  # this line throws error, as funs is not returning any value to compute upon
print(y)
```

Q9. How do you make a function variable refer to the global variable?

Answer – To refer to the global variable inside the function we need to make use of the *global* keyword. By using *global* keyword we can make changes to that variable inside the function and the changes would reflect outside the function as well.

Example -

```
x = 10  # Global variable
def my_function():
    global x  # Declare x as a global variable inside the function
    x = 5
    print(x)# Modify the global variable

print(x)       # Output: 10 (before calling the function)
my_function()
print(x)       # Output: 5 (after calling the function)
```

Q10. What is the data type of None?

Answer – data type of None is *NoneType.* When assigned to a variable it indicates that the variable has nothing inside it. It not similar to an empty string(‘’) or 0()

Q11. What does the sentence import areallyourpetsnamederic do?

Answer – The code *import areallyourpetsnamederic* will load the *areallyourpetsnamederic* Module into the memory. *areallyourpetsnamederic* can be an external module/library. After import, we can start using the methods/functions defined inside the *areallyourpetsnamederic* Module.

Q12. If you had a bacon() feature in a spam module, what would you call it after importing spam?

Answer –

```
import spam
spam.bacon()
```

Q13. What can you do to save a programme from crashing if it encounters an error?

Answer – We can use Try Except finally to deal with the errors. The statements inside the try block will be checked for any errors. The except statement will be executed when any error occurs in the try block. *Finally* block is used to execute the statements that need to be run irrespective of the occurrence of error in try block.

```
Try:
   # Code that may raise an exception

except:
   # Handle the specific exception

finally:
   # Code that will always run
```

Q14. What is the purpose of the try clause? What is the purpose of the except clause?

Answer – Try clause checks for the errors in the statements written under try clause. The except clause is used to run a specific set of statements when a particular error occurs in try clause.