**Assignment\_3**

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

**Answer:**

* It reduces duplication of code.
* Complex problems can be decomposed into simple pieces.
* It improves the clarity of the code.
* It helps in reusing the code and stop us in writing the same logic again and again.
* Information (code block) can be hidden within a function.

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

**Answer:**

Code inside a function is executed only when the function is called and is not executed if the code is only defined inside a function and the function is not called.

3. What statement creates a function?

**Answer:**

Function blocks begin with the keyword **def** followed by the function name and parentheses ( ( ) )

**def** functionname( parameters ):

….

…..

return [expression]

The statement return [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

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

**Answer:**

A **function** is a named sequence of statements that performs some useful operation. Functions may or may not take parameters and may or may not produce a result.

A **function call** is a statement that executes a function. It consists of the name of the function followed by a list of arguments enclosed in parentheses.

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

**Answer:** 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.

Local scope is created at function call, not at function definition, so we can have as many different local scopes as function calls. Even the same function is called multiple times, or recursively, each call will result in a new local scope being created.

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

**Answer:**

Variables inside a function exist only within the function or local scope associated with the function call. When the function returns, the local scope is destroyed, and the variables are forgotten.

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

**Answer:**

In general, a function takes arguments (if any), performs some operations, and returns a value (or object). The value that a function returns to the caller is generally known as the function’s return value.

If we define a function with an explicit return statement that has an explicit return value, then we can use that return value in any expression:

Ex: >>> num = return\_11()

>>> num

11

>>> return\_11() \* 2

22

>>> return\_42() + 5

16

Since return\_11() returns a numeric value, you can use that value in a math expression or any other kind of expression in which the value has a logical or coherent meaning.

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

**Answer:**

If we don’t explicitly use a return value in a return statement, or if we totally omit the return statement, then Python will implicitly return a default value for you. That default return value will always be None.

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

**Answer:**

When you create a variable inside a function, that variable is local, and can only be used inside that function.To create a global variable inside a function, we can use the global keyword.

Ex:

def myfunc():

global x

10. What is the data type of None?

**Answer:**

The None keyword is used to define a null value, or no value at all.None is not the same as 0, False, or an empty string. None is a data type of its own (NoneType) and only None can be None.

11. What does the sentence import areallyourpetsnamederic do?

**Answer:**

import areallyourpetsnamederic will make the module/function “areallyourpetsnamederic” available in current code.

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

**Answer:**

from spam import bacon

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

**Answer:** Exceptions are raised when the program encounters an error during its execution. They disrupt the normal flow of the program and usually end it abruptly. To avoid this, we can catch them and handle them appropriately by using “**Exception Handling**” in the code.

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

**Answer:** Python would process all code inside the **try** and **except** statement. When it encounters an error, the control is passed to the **except** block, skipping the code in between.

Ex:

**def** addNumbers(a, b):

**try**:

**return** a **+** b

**except** Exception as e:

**return** 'Error occurred : ' **+** str(e)

print(addNumbers('', 10))

When the function is called, the try clause will run. If no exceptions are raised, the program will run as expected.But if an exception is raised in the try clause, the flow of execution will immediately jump to the except clause to handle the exception.