1. What is the result of the code, and explain?

>>> X = 'iNeuron'

>>> def func():

print(X)

>>> func()

**Output:** iNeuron

**Reason:** Variables that are created outside of a function (as in all of the examples above) are known as global variables. Global variables can be used by everyone, both inside of functions and outside.Here, X=”iNeuron”, is a Global Variable. Hence it can be used inside the function func()

2. What is the result of the code, and explain?

>>> X = 'iNeuron'

>>> def func():

X = 'NI!'

>>> func()

>>> print(X)

**Output:** NI!

**Reason:** If you create a variable with the same name inside a function, this variable will be local, and can only be used inside the function. The global variable with the same name will remain as it was, global and with the original value.

3. What does this code print, and why?

>>> X = 'iNeuron'

>>> def func():

X = 'NI'

print(X)

>>> func()

>>> print(X)

**Output:** NI!

iNeuron

**Reason:** The first time the code prints X, it is inside the function hence the local variable value is printed. The second time the code prints X, it is outside the function hence the global variable value is printed.

4. What output does this code produce? Why?

>>> X = 'iNeuron'

>>> def func():

global X

X = 'NI'

>>> func()

>>> print(X)

**Output:** NI

**Reason:** Normally, when we 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. The global keyword can also be used to change a global variable inside a function.

5. What about this code—what’s the output, and why?

>>> X = 'iNeuron'

>>> def func():

X = 'NI'

def nested():

print(X)

nested()

>>> func()

>>> X

**Output:** iNeuron

‘iNeuron’

**Reason:** When function nested() is called at first, it simply prints the global variable value of X which is ‘iNeuron’.

When function func() is called next, X is given the value of “NI”. But the scope of this assigned value only stays inside the function func(). That is the reason why when we print X outside any of the functions, it takes the value of the global variable X=”iNeuron”.

6. How about this code: what is its output in Python 3, and explain?

>>> def func():

X = 'NI'

def nested():

nonlocal X

X = 'Spam'

nested()

print(X)

>>> func()

**Output:** Spam

**Reason:** Nonlocal variables are used in nested functions whose local scope is not defined. This means that the variable can be neither in the local nor the global scope.

If the keyword “nonlocal” was not used in the above program then, it would give “NI” as output.

In the above code, there is a nested nested() function. We use nonlocal keywords to create a nonlocal variable. The nested() function is defined in the scope of another function func().