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

>>> X = 'iNeuron'

>>> def func():

print(X)

>>> func()

Ans: result is iNeuron. The variable X is not a local variable , its global. As the function directly calls thsis variable its value prints.

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

>>> X = 'iNeuron'

>>> def func():

X = 'NI!'

>>> func()

>>> print(X)

Ans: Here another variable X is delclared and assigned a value inside the function. The X outside the funtion is global and the X in func() is a local variable.So here the X in the function and out of the function are different locations.

3. What does this code print, and why?

>>> X = 'iNeuron'

>>> def func():

X = 'NI'

print(X)

>>> func()

>>> print(X)

Ans: The output of func() is NI and the output of print(X) is iNeuron. The print statement in function prints its local variable X and Second line print(X) prints the global variable X.

4. What output does this code produce? Why?

>>> X = 'iNeuron'

>>> def func():

global X

X = 'NI'

>>> func()

>>> print(X)

Ans: The output is NI. Where ever we declare a variable as global then the variable is accessible from anywhere in the program and its value will be same untill and unless we change the value globally.

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

>>> X = 'iNeuron'

>>> def func():

X = 'NI'

def nested():

print(X)

nested()

>>> func()

>>> X

Ans: The output is 'iNeuron'. The scope of variable X in func is local so even if we change the value of X in func, it won't reflect outsidee the function. As we call the variale X outside the function , it will print the globally assigned value of X only.

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()