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

>>> X = 'iNeuron'

>>> def func():

print(X)

>>> func()

It will print `iNeuron’

In last line the function called and because X is public variable, its value recognized into function and print it.

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

>>> X = 'iNeuron'

>>> def func():

X = 'NI!'

>>> func()

>>> print(X)

It will print `iNeuron’

Although the value of X changed inside the function but that X is local variable of function and it does not affect the X outside the function.

3. What does this code print, and why?

>>> X = 'iNeuron'

>>> def func():

X = 'NI'

print(X)

>>> func()

>>> print(X)

It will print `NI ` and `iNeuron’ respectively

Firsly, the function will call and the X inside it will get ‘NI’ then its value will print, after running function, next command print will run that print public X = iNeuron.

4. What output does this code produce? Why?

>>> X = 'iNeuron'

>>> def func():

global X

X = 'NI'

>>> func()

>>> print(X)

It will print `NI’

The X inside the function is global which point the X outside the function, then when we change the value the value of X will changed and the result of print will be ‘NI

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

>>> X = 'iNeuron'

>>> def func():

X = 'NI'

def nested():

print(X)

nested()

>>> func()

>>> X

Func() will print ‘NI’ because X inside the function changed, the enclosing variable can be use inside nested function in read only format.

X will show ‘iNeuron’ which remaid without change after running function

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()

It will print ‘Spam’

Because we mentioned nonlocal X then we can change the value of variable inside nested function.