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

>>> X = 'iNeuron'

>>> def func():

print(X)

>>> func()

Ans: iNeuron

We can use global variable inside a func

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

>>> X = 'iNeuron'

>>> def func():

X = 'NI!'

>>> func()

>>> print(X)

Ans: iNeuron

value of global variable cant be changed in global scope by assigning it new value in local scope

3. What does this code print, and why?

>>> X = 'iNeuron'

>>> def func():

X = 'NI'

print(X)

>>> func()

>>> print(X)

Ans:NI

iNeuron

when global var is assigned a new value in local scope it gives that value and outside the func it gives global value

4. What output does this code produce? Why?

>>> X = 'iNeuron'

>>> def func():

global X

X = 'NI'

>>> func()

>>> print(X)

Ans: NI

In the local scope we have defined X as global var so value assigned to it changes 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: NI

iNeuron

when we are calling func, inside nested we are assigning new value to X so the new value gets printed but global value of X is not changing

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()

Ans: Spam

When we are calling func, X is defined as non local means it can be used in the scope of func() so Spam gets printed