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

>>> X = ‘iNeuron’

>>> def func():

Print(X)

>>> func()

**Ans: iNeuron (the assigned value for the function is X)**

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

>>> X = ‘iNeuron’

>>> def func():

X = ‘NI!’

>>> func()

>>> print(X)

**Ans : iNeuron (the assigned value for X is not changed by the def function.)**

3. What does this code print, and why?

>>> X = ‘iNeuron’

>>> def func():

X = ‘NI’

print(X)

>>> func()

>>> print(X)

**Ans : iNeuron (gives out by the first print statement)**

**iNeuron (gives out by the second print statement)**

4. What output does this code produce? Why?

>>> X = ‘iNeuron’

>>> def func():

global X

X = ‘NI’

>>> func()

>>> print(X)

**Ans : NI (the value of X changed by the statement global)**

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 : iNeuron (gives out by the function calling X)**

**'iNeuron' (as the code calling X)**

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 : it gives syntax error as there is no binding for nonlocal ‘X’ found.**