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

>>> X = 'iNeuron'

>>> def func():

print(X)

>>> func()

A1. Prints ‘iNeuron’ in the console.

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

>>> X = 'iNeuron'

>>> def func():

X = 'NI!'

>>> func()

>>> print(X)

A2. Prints ‘iNeuron’ in the console because the function is not returning anything and the ‘iNeuron’ string is assigned to X in the global scope.

3. What does this code print, and why?

>>> X = 'iNeuron'

>>> def func():

X = 'NI'

print(X)

>>> func()

>>> print(X)

A3. Output will be:

NI

iNeuron

This is because the function will print the X value in local scope and then the X value in global scope will be printed.

4. What output does this code produce? Why?

>>> X = 'iNeuron'

>>> def func():

global X

X = 'NI'

>>> func()

>>> print(X)

A4. Output: ‘NI’

Function is changing the X value in the global scope.

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

>>> X = 'iNeuron'

>>> def func():

X = 'NI'

def nested():

print(X)

nested()

>>> func()

>>> X

A5. Output: iNeuron

Nested() function uses the X in global scope and prints the same.

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()