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

>>> X = 'iNeuron'

>>> def func():

print(X)

>>> func()

# iNeuron is the result of the code because the func() because the function references a global variable in the enclosing module (because it is not assigned in the function, it is considered global

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

>>> X = 'iNeuron'

>>> def func():

X = 'NI!'

>>> func()

>>> print(X)

iNeuron because X is always be iNeuron because assigning the variable inside the function makes it a local and effectively hides the global of the same name. The print statement finds the variable unchanged in the global (module) scope.

3. What does this code print, and why?

>>> X = 'iNeuron'

>>> def func():

X = 'NI'

print(X)

>>> func()

>>> print(X)

It prints 'NI' on one line and 'iNeuron' on another, because the reference to the variable within the function finds the assigned local and the reference in the print statement finds the global.

4. What output does this code produce? Why?

>>> X = 'iNeuron'

>>> def func():

global X

X = 'NI'

>>> func()

>>> print(X)

It prints 'NI' because the global declaration forces the variable assigned inside the function to refer to the variable in the enclosing 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

The output in this case is again 'NI' on one line and 'iNeuron' on another, becausethe print statement in the nested function finds the name in the enclosing function's local scope, and the display at the end finds the variable in the global scope

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()

This example prints 'Spam', because the nonlocal statement means that the assignment to x inside the nested function changes x in the enclosing function's local scope. Without this statement, this assignment would classify x as local to the nested function, making it a different variable; the code would then print 'NI' instead