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

>>> X = 'iNeuron'

>>> def func():

print(X)

>>> func()

**Ans:** iNeuron is the result of this code because we have declared the iNeuron in variable X and then called same variable in function so that it prints the iNeuron.

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

>>> X = 'iNeuron'

>>> def func():

X = 'NI!'

>>> func()

>>> print(X)

**Ans:** iNeuron is the result of this code because it prints the global variable as print(X) is called outside of defined function.

1. What does this code print, and why?

>>> X = 'iNeuron'

>>> def func():

X = 'NI'

print(X)

>>> func()

>>> print(X)

**Ans:** Here answer is NI & iNeuron both as each variable is called for print. NI result is coming for print which is inside of the function as a local variable and iNeuron is coming for print which is outside of the function as a global variable.

1. What output does this code produce? Why?

>>> X = 'iNeuron'

>>> def func():

global X

X = 'NI'

>>> func()

>>> print(X)

**Ans:** Here the result is NI because in function X is declared as a global variable and hence on print statement it showing the result for global variable.

1. What about this code—what is the output, and why?

>>> X = 'iNeuron'

>>> def func():

X = 'NI'

def nested ():

print(X)

nested ()

>>> func()

>>> X

**Ans:** Process finished with exit code 0 getting this as output.

1. 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:** Process finished with exit code 0 getting this as output.