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

>>> X = 'iNeuron'

>>> def func():

print(X)

>>> func()

Ans. The code defines a variable X with the value 'iNeuron'. It then defines a function func() that prints the value of X. When the function is called using func(), it will print the value of X, which is 'iNeuron'. Therefore, the output of the code will be:

iNeuron

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

>>> X = 'iNeuron'

>>> def func():

X = 'NI!'

>>> func()

>>> print(X)

Ans. The code defines a variable X with the value 'iNeuron'. It then defines a function func() that has a local variable X with the value 'NI!'. When the function func() is called using func(), it only changes the local variable X within the function scope, and not the global variable X outside the function. Therefore, the global variable X remains unchanged with the value 'iNeuron'.

After calling the function, it then prints the value of the global variable X, which is still 'iNeuron'. So, the output of the code will be

iNeuron

3. What does this code print, and why?

>>> X = 'iNeuron'

>>> def func():

X = 'NI'

print(X)

>>> func()

>>> print(X)

Ans. The code defines a variable X with the value 'iNeuron'. It then defines a function func() that has a local variable X with the value 'NI'. When the function func() is called using func(), it prints the value of the local variable X within the function scope, which is 'NI'.

After calling the function, it then prints the value of the global variable X, which remains unchanged with the value 'iNeuron'. The local variable X inside the function does not affect the value of the global variable X. So, the output of the code will be:

NI

iNeuron

4. What output does this code produce? Why?

>>> X = 'iNeuron'

>>> def func():

global X

X = 'NI'

>>> func()

>>> print(X)

Ans.The code defines a variable X with the value 'iNeuron'. It then defines a function func() that uses the global keyword to indicate that the variable X inside the function should refer to the global variable X.

When the function func() is called using func(), it modifies the value of the global variable X to 'NI'. So, after calling the function, the value of the global variable X is changed to 'NI'. When print(X) is executed after the function call, it will print the modified value of the global variable X, which is 'NI'.

So, the output of the code will be:

NI

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. The code defines a variable X with the value 'iNeuron'. It then defines a function func() that has a local variable X with the value 'NI'. Inside the func() function, there is another nested function nested() that prints the value of the variable X.

When the function func() is called using func(), it sets the local variable X to 'NI', but this local variable is separate from the global variable X and does not affect it. Then, it calls the nested function nested(), which prints the value of the local variable X within its scope, which is 'NI'.

After executing the func() function, it does not change the value of the global variable X. So, when X is printed outside the function, it will still have the global value 'iNeuron'.

NI

iNeuron

The output of the code will be:

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. The code defines a function func() that has a local variable X with the value 'NI'. Inside the func() function, there is another nested function nested() that uses the nonlocal keyword to indicate that the variable X inside nested() should refer to the X in the nearest enclosing scope, which is the func() function.

When func() is called, it sets its local variable X to 'NI' and then calls the nested() function. The nested() function changes the value of the variable X to 'Spam', but this variable is the one defined in the enclosing func() function, not in the global scope.

func() function prints the value of its local variable X, which is 'Spam'.

So, the output of the code will be;

Spam