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

X = 'iNeuron'

def func():

print(X)

func()

Output : iNeuron

X = 'iNeuron' defines a global variable X with the value 'iNeuron'.

func() is defined to print the value of X.

Inside func(), when print(X) is executed, it accesses the global variable X and prints its value 'iNeuron'.

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

X = 'iNeuron';

def func():

X = 'NI';

func()

print(X)

Output : iNeuron

The global variable X with the value 'iNeuron' is unaffected by the assignment to a local variable X inside the function func().

3. What does this code print, and why?

X = 'iNeuron';

def func():

X = ‘NI’

print(X)

func()

print(X)

Output : NI

iNeuron

Inside func(), the local variable X shadows (or hides) the global variable X with the same name. print(X) inside func() prints 'NI', and print(X) outside func() prints 'iNeuron', demonstrating that the global variable X retains its initial value after the function call.

4. What output does this code produce? Why?

X = 'iNeuron'

def func():

global X

X = 'NI'

func()

print(X)

Output : NI

By using global X inside func(), the function func() modifies the global variable X instead of creating a new local variable.

After calling func(), print(X) confirms that the global variable X has been updated to '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

Output : 'iNeuron'

This function defines a local variable X and assigns the value "NI" to it. However, this local variable only exists within the scope of the func function and doesn't affect the global variable 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()

Output: Spam

The nonlocal keyword allows the nested() function to modify the variable X in the enclosing func() scope.