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

>>> X = 'iNeuron'

>>> def func():

print(X)

>>> func()

Ans1

Output

iNeuron

Explanation:

At first, the variable X is assigned a string value of 'iNeuron'.

Then, a function func() is defined, which when called will print the value of X.

When func() is called, it simply prints the value of X which is 'iNeuron'.

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

>>> X = 'iNeuron'

>>> def func():

X = 'NI!'

>>> func()

>>> print(X)

Ans2

Output

iNeuron

Explanation:

Initially, the variable X is assigned the string value of 'iNeuron'.

Then, the function func() is defined which assigns the string value of 'NI!' to a local variable also named X inside the function.

When the func() function is called, it only modifies the local variable X and does not affect the global variable X.

After the function call, the print() statement prints the global variable X which has not been modified and still holds the value of 'iNeuron'. So, the output will be 'iNeuron'.

3. What does this code print, and why?

>>> X = 'iNeuron'

>>> def func():

X = 'NI'

print(X)

>>> func()

>>> print(X)

Ans3

Output

NI

iNeuron

Explanation:

At first, the variable X is assigned the string value of 'iNeuron'.

Then, the function func() is defined which assigns the string value of 'NI' to a local variable X inside the function and prints it using the print() statement.

When the func() function is called, it modifies the local variable X to 'NI' and prints it. So, the first output will be 'NI'.

After the function call, the print() statement prints the global variable X which has not been modified and still holds the value of 'iNeuron'. So, the second output will be 'iNeuron'.

4. What output does this code produce? Why?

>>> X = 'iNeuron'

>>> def func():

global X

X = 'NI'

>>> func()

>>> print(X)

Ans4

Output

NI

Explanation:

At first, the variable X is assigned the string value of 'iNeuron'.

Then, the function func() is defined which declares X as a global variable using the global keyword and assigns the string value of 'NI' to it.

When the func() function is called, it modifies the global variable X to 'NI'.

After the function call, the print() statement prints the value of the global variable X, which has been modified inside the function to 'NI'. So, the output 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

Ans5

Output

NI

iNeuron

Explanation

First, a global variable X is defined and assigned the value 'iNeuron'.

Then, a function func is defined without any input parameters.

Inside func, there is another function nested defined which doesn't take any input parameters either.

A local variable X is defined inside func and assigned the string value 'NI'. This variable X is only accessible within the scope of the func function.

Inside nested, the value of X is printed using the print statement. Since there is no X variable defined within nested, Python looks for the nearest enclosing scope that defines X, which is the func function. Therefore, the value of X printed will be 'NI'.

nested is called inside func, so when func is called later on, nested is also executed and the value of X is printed inside nested.

Finally, outside of any function, the value of the global variable X is printed using print(X), which will output 'iNeuron'.

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()

Ans6

Output

iNeuron

iNeuron

Explanation

First, a function func is defined without any input parameters.

Inside func, there is another function nested defined which doesn't take any input parameters either.

Then, X is defined inside func and assigned the string value 'NI'.

Inside nested, nonlocal X is used to indicate that the X being referred to is the one defined in the enclosing function func.