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

>>> X = 'iNeuron'

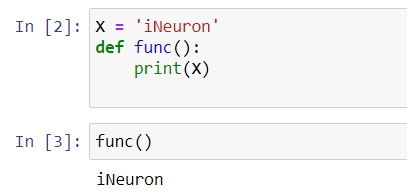
>>> def func():

print(X)

>>> func()

🡪 The result of the above code will be “iNeuron”.

The reason being in 1st line of code variable ‘X’ is assigned a string (iNeuron) which act as global variable and then the function func() is printing the variable X. So when ever we will call the function func() it will take the value of X as global variable(in this case iNeuron).



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

>>> X = 'iNeuron'

>>> def func():

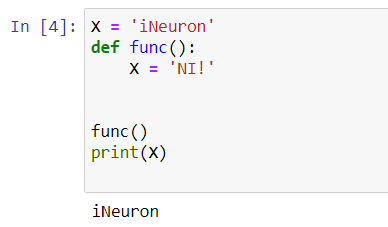
X = 'NI!'

>>> func()

>>> print(X)

🡪 The result of the above code will be “iNeuron”.

The reason being in 1st line of code we have declared variable ‘X’ and assigned it a string (iNeuron), then we have defined a function func() which assigns variable ‘X’ a string (NI!). Here the declaration of variable ‘X’ in the function is called as local variable and is applicable within the function itself. The outside declaration of variable ‘X’ is called global declaration and is valid globally. Thus when print(X) is executed it will take X = ‘iNeuron’ as value and print it, even though func() is called just before the print statement. When the func() is called it is just assigning X = ‘NI!’ but not printing it and declaration is limited in func().



3. What does this code print, and why?

>>> X = 'iNeuron'

>>> def func():

X = 'NI'

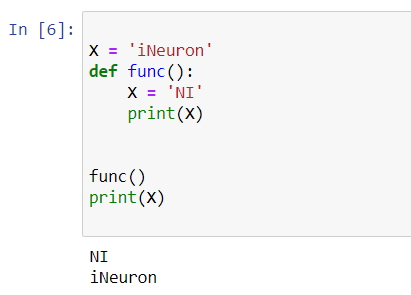
print(X)

>>> func()

>>> print(X)

🡪 The result of the above code will be “NI” on 1st line and “iNeuron” on 2nd line.

The reason being in 1st line of code we have declared variable ‘X’ and assigned it a string (iNeuron), then we have defined a function func() which assigns variable ‘X’ a string (NI) and then prints the variable ‘X’. Here the declaration of variable ‘X’ in the function is local variable and is applicable within the function itself. The outside declaration of variable ‘X’ is global declaration and is valid globally. Thus when func() is called it prints “NI” and in next line when print(X) is executed it will take X = ‘iNeuron’ as value and print it. Hence we see both as output.



4. What output does this code produce? Why?

>>> X = 'iNeuron'

>>> def func():

global X

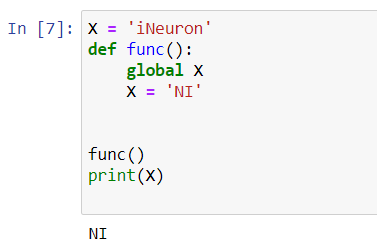
X = 'NI'

>>> func()

>>> print(X)

🡪 The result of the above code will be “NI”.

This is so because in 1st line of code we have declared variable ‘X’ and assigned it a string (iNeuron), then we have defined a function func() which 1st declares variable ‘X’ as global and then assigns ‘X’ a string (NI). Here in the function the keyword “global” is used for global declaration of variable. Thus when we call the func() it will declare X as global variable and then assign it the string ‘NI’. Hence now whenever ‘X’ is called its value will be the string ‘NI’.



5. What about this code—what’s the output, and why?

>>> X = 'iNeuron'

>>> def func():

X = 'NI'

def nested():

print(X)

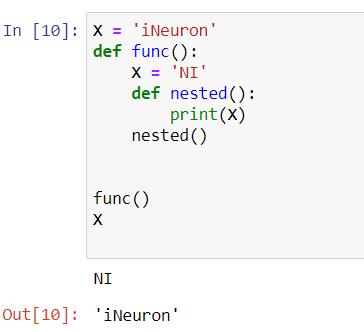
nested()

>>> func()

>>> X

🡪 The result of the above code will be “NI” and “iNeuron”.

This is so as in 1st line of code we have declared variable ‘X’ and assigned it a string (iNeuron), then we have defined a function func() which assigns variable ‘X’ a string (NI) and inside this function another function named nested() is defined which prints the variable ‘X’. Hence when we call the function func() 1st it will give result as ‘NI’ because here the value of variable ‘X’ declared in the func() is local to function thus the defined function nested() uses this ‘X’ value and prints “NI”. After that we called the variable ‘X’ as X = ‘iNeuron’ is global variable whenever we will call it will give output as “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()

🡪 The result of the above code will be “Spam”.

Because in the code we f1st defined function func() in it we declared variable X=’NI’ and then we defined another function called nested() inside the func(). In the nested() we have used nonlocal keyword for variable ‘X’. This keyword is used in nested functions for variables whose local scope is not defined. Hence X is now nonlocal variable and assigned a string ‘Spam’. When we call the function nested() this nonlocal scope is applicable to the variable ‘X’. And when we print the variable ‘X’ after calling the nested() we get result as “Spam”.

