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

X = 'iNeuron'

def func():

print(X)

func()

**Result:**

iNeuron

**Explanation:** The function func() prints the value of X, which is 'iNeuron'. Since X is a global variable, it is accessible within the function, and Python will print 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)

**Result:**

iNeuron

**Explanation:** Inside the func() function, X is assigned the value 'NI!'. However, this assignment only affects the local variable X within the scope of the function, not the global variable X. Therefore, when print(X) is called outside the function, it prints the global X, which is 'iNeuron'.

**3. What does this code print, and why?**

X = 'iNeuron'

def func():

X = 'NI'

print(X)

func()

print(X)

**Result:**

NI

iNeuron

**Explanation:**

* Inside the func() function, X is locally assigned the value 'NI', and print(X) prints 'NI' because it refers to the local X in the function.
* After the function call, the global X remains unchanged (still 'iNeuron'), so the second print(X) prints 'iNeuron'.

**4. What output does this code produce? Why?**

X = 'iNeuron'

def func():

global X

X = 'NI'

func()

print(X)

**Result:**

NI

**Explanation:** The global keyword inside the func() function allows the function to modify the global variable X. So, when X is assigned 'NI' inside the function, it modifies the global X. After the function call, print(X) prints 'NI'.

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

X = 'iNeuron'

def func():

X = 'NI'

def nested():

print(X)

nested()

func()

print(X)

**Result:**

NI

iNeuron

**Explanation:**

* In the function func(), the local X is set to 'NI', and then nested() prints the value of X, which is 'NI' because it refers to the X in the local scope of func().
* After the function call, the global X remains unchanged (still 'iNeuron'), so the second print(X) prints '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()

**Result:**

Spam

**Explanation:**

* The nonlocal keyword allows nested() to modify the variable X from the nearest enclosing scope, which is the func() function's scope.
* Inside nested(), X is changed to 'Spam', and when print(X) is called inside func(), it prints 'Spam' because X was modified in the enclosing scope.

In Python 3, the nonlocal keyword is used to refer to variables in the nearest enclosing scope that are not global.