### Result of the code:

python

Copy code

>>> X = 'iNeuron'

>>> def func():

print(X)

>>> func()

**Result:**

csharp

Copy code

NameError: name 'X' is not defined

**Explanation:** The code snippet has a syntax error because the print(X) statement inside the function func() is not properly indented. After fixing the indentation:

python

Copy code

>>> X = 'iNeuron'

>>> def func():

print(X)

>>> func()

**Output:**

Copy code

iNeuron

**Explanation:** The function func() prints the global variable X, which is 'iNeuron'. The global variable is accessible inside the function since it’s not shadowed by any local variable.

### 2. Result of the code:

python

Copy code

>>> X = 'iNeuron'

>>> def func():

X = 'NI!'

>>> func()

>>> print(X)

**Result:**

Copy code

iNeuron

**Explanation:** Inside func(), a local variable X is defined and it shadows the global variable X only within the scope of func(). The print(X) outside the function refers to the global variable, which remains 'iNeuron'.

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

python

Copy code

>>> X = 'iNeuron'

>>> def func():

X = 'NI'

print(X)

>>> func()

>>> print(X)

**Output:**

Copy code

NI

iNeuron

**Explanation:**

* When func() is called, it prints the local variable X within the function, which is 'NI'.
* After the function call, print(X) prints the global variable X, which remains 'iNeuron'.

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

python

Copy code

>>> X = 'iNeuron'

>>> def func():

global X

X = 'NI'

>>> func()

>>> print(X)

**Output:**

Copy code

NI

**Explanation:**

* Inside func(), the global X statement makes X refer to the global variable X. Hence, when X = 'NI' is executed within the function, it changes the global X to 'NI'.
* The print(X) after the function call prints the updated global X, which is now 'NI'.

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

python

Copy code

>>> X = 'iNeuron'

>>> def func():

X = 'NI'

def nested():

print(X)

nested()

>>> func()

>>> X

**Output:**

Copy code

NI

**Explanation:**

* Inside func(), the local variable X is set to 'NI'.
* The nested() function prints the X variable from its enclosing scope, which is 'NI'.
* The global X remains unchanged, so if you evaluate X outside of func(), it will still be 'iNeuron'. However, this result is not printed in the code snippet.

### 6. How about this code: what is its output in Python 3, and explain?

python

Copy code

>>> def func():

X = 'NI'

def nested():

nonlocal X

X = 'Spam'

nested()

print(X)

>>> func()

**Output:**

Copy code

Spam

**Explanation:**

* The nonlocal X statement in the nested() function tells Python that X refers to the variable in the nearest enclosing scope (in this case, func()), not the global scope or a new local variable.
* nested() changes the value of X to 'Spam'.
* When print(X) is called in func(), it prints the modified value 'Spam' because the nonlocal declaration allows nested() to modify X in func().