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

>>> X = 'iNeuron'

>>> def func():

print(X)

>>> func()

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

>>> X = 'iNeuron'

>>> def func():

X = 'NI!'

>>> func()

>>> print(X)

3. What does this code print, and why?

>>> X = 'iNeuron'

>>> def func():

X = 'NI'

print(X)

>>> func()

>>> print(X)

4. What output does this code produce? Why?

>>> X = 'iNeuron'

>>> def func():

global X

X = 'NI'

>>> func()

>>> print(X)

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

>>> X = 'iNeuron'

>>> def func():

X = 'NI'

def nested():

print(X)

nested()

>>> func()

>>> 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()

Ans: 1. The result of the code is 'iNeuron'. When the function `func()` is called, it tries to print the value of the variable `X`, which is defined outside the function. Since `X` is a global variable, it is accessible within the function, and its value is 'iNeuron'. Therefore, 'iNeuron' is printed.

2. The result of the code is 'iNeuron'. In the `func()` function, a new variable `X` is created and assigned the value 'NI!'. This variable `X` is a local variable to the function and does not affect the value of the global variable `X`. So when `func()` is called, it doesn't modify the value of the global `X`. When `print(X)` is executed outside the function, it refers to the global variable `X`, which still holds the value 'iNeuron'. Therefore, 'iNeuron' is printed.

3. The code prints:

```

NI

iNeuron

```

Inside the `func()` function, a new variable `X` is created and assigned the value 'NI'. When `print(X)` is executed within the function, it refers to this local variable `X`, so 'NI' is printed. Outside the function, when `print(X)` is executed, it refers to the global variable `X`, which still holds the value 'iNeuron'. Therefore, 'iNeuron' is printed.

4. The output of the code is 'NI'. The `func()` function uses the `global` keyword to indicate that the variable `X` being assigned is the global variable, not a local one. So when `func()` is called, the global variable `X` is modified and assigned the value 'NI'. When `print(X)` is executed, it refers to the modified global `X` variable, which now holds the value 'NI'. Therefore, 'NI' is printed.

5. The output of the code is 'NI'. Inside the `func()` function, a new variable `X` is created and assigned the value 'NI'. Then a nested function `nested()` is defined, which prints the value of `X`. When `nested()` is called inside `func()`, it prints the value of the local variable `X`, which is 'NI'. After that, the code outside the function `func()` executes `X`, which refers to the global variable `X` with the value 'iNeuron'. Therefore, 'NI' is printed followed by the value of the global `X`, which is 'iNeuron'.

6. The output of the code is 'Spam'. Inside the `func()` function, a variable `X` is defined and assigned the value 'NI'. Then a nested function `nested()` is defined, and the `nonlocal` keyword is used to indicate that `X` is a variable from the nearest enclosing scope that should be modified. So when `nested()` is called, it modifies the variable `X` from the `func()` scope and assigns it the value 'Spam'. Finally, when `print(X)` is executed within `func()`, it refers to the modified variable `X` in the `nested()` function, which now holds the value 'Spam'. Therefore, 'Spam' is printed.