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

>>> X = 'iNeuron'

>>> def func():

print(X)

>>> func()

iNeuron

* The code defines a global variable X and assigns it the string value 'iNeuron'.
* It also defines a function func without any parameters.
* Inside the func function, the statement print(X) is executed, which will print the value of the global variable X, which is 'iNeuron'.
* When the function func is called with func(), it prints the value of X, resulting in iNeuron.

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

>>> X = 'iNeuron'

>>> def func():

X = 'NI!'

>>> func()

>>> print(X)

iNeuron

* The code defines a global variable X and assigns it the string value 'iNeuron'.
* It also defines a function func without any parameters.
* Inside the func function, there is an assignment statement X = 'NI!', but this assignment creates a new local variable X that shadows the global variable X.
* When the function func is called with func(), it assigns the value 'NI!' to the local variable X, but this assignment does not affect the value of the global variable X.
* After the function call, when print(X) is executed, it prints the value of the global variable X, which is still 'iNeuron'.

3. What does this code print, and why?

>>> X = 'iNeuron'

>>> def func():

X = 'NI'

print(X)

>>> func()

>>> print(X)

NI

iNeuron

* The code defines a global variable X and assigns it the string value 'iNeuron'.
* It also defines a function func without any parameters.
* Inside the func function, there is an assignment statement X = 'NI', but this assignment creates a new local variable X that shadows the global variable X.
* When the function func is called with func(), it assigns the value 'NI' to the local variable X and prints its value, which is 'NI'.
* After the function call, when print(X) is executed outside the function, it prints the value of the global variable X, which is still 'iNeuron'. The local variable X defined inside the function does not affect the value of the global variable.

4. What output does this code produce? Why?

>>> X = 'iNeuron'

>>> def func():

global X

X = 'NI'

>>> func()

>>> print(X)

NI

* The code defines a global variable X and assigns it the string value 'iNeuron'.
* The function func is defined with no parameters.
* Inside the func function, the global keyword is used to indicate that we want to use the global variable X instead of creating a new local variable with the same name.
* The assignment statement X = 'NI' inside the function assigns the value 'NI' to the global variable X.
* When the function func is called with func(), it modifies the global variable X to have the value 'NI'.
* After the function call, when print(X) is executed outside the function, it prints the value of the global variable X, which has been updated to '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

NI

'iNeuron'

* The code defines a global variable X and assigns it the string value 'iNeuron'.
* The function func is defined without any parameters.
* Inside the func function, a nested function nested is defined.
* The nested function nested uses the value of X from the enclosing scope, which is the local variable X of the func function.
* When the func function is called with func(), it executes the nested function nested.
* The nested function nested prints the value of X, which is the local variable X of the func function, having the value 'NI'.
* After the function call, when X is printed outside the function, it refers to the global variable X, which retains its original value '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()

Syntax error

* The code defines a function func without any parameters.
* Inside the func function, a nested function nested is defined.
* The nested function nested attempts to use the nonlocal keyword to refer to a variable X in an outer scope.
* However, there is no outer scope containing the variable X for the nonlocal keyword to bind to.
* As a result, a SyntaxError is raised indicating that there is no binding for the nonlocal variable 'X'.
* The code execution is halted at this point, and no output is produced.