**Assignment22**

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

>>> X = 'iNeuron'

>>> def func():

print(X)

>>> func()

A)

The code defines a variable X with the string value 'iNeuron'. Then, it defines a function func that prints the value of X. Finally, it calls the func function.

When the func function is called, it prints the value of X, which is 'iNeuron'.

So, the result of the code is:

iNeuron

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

>>> X = 'iNeuron'

>>> def func():

X = 'NI!'

>>> func()

>>> print(X)

1. Initially, a variable X is defined in the global scope with the string value 'iNeuron'.
2. Then, a function func is defined. Inside this function, there's a local variable X assigned the value 'NI!'. However, this local variable X is separate from the global variable X defined outside the function. It's a local variable that only exists within the scope of the function.
3. When the func function is called, it assigns the value 'NI!' to its local variable X. But since this local variable X is only accessible within the function's scope, it doesn't affect the global variable X.
4. After calling func(), when print(X) is executed outside the function, it prints the value of the global variable X, which remains 'iNeuron' because the local variable X inside func() doesn't affect the global variable X.
5. 3. What does this code print, and why?
6. >>> X = 'iNeuron'
7. >>> def func():
8. X = 'NI'
9. print(X)
10. >>> func()
11. >>> print(X)
12. NI
13. INeuron

Initially, a variable X is defined in the global scope with the string value 'iNeuron'.

Then, a function func is defined. Inside this function, there's a local variable X assigned the value 'NI'. This local variable X is separate from the global variable X defined outside the function.

When the func function is called, it prints the value of its local variable X, which is 'NI'.

After calling func(), when print(X) is executed outside the function, it prints the value of the global variable X, which remains 'iNeuron'. The local variable X defined within the function func doesn't affect the global variable X.

4. What output does this code produce? Why?

>>> X = 'iNeuron'

>>> def func():

global X

X = 'NI'

>>> func()

>>> print(X)

1. The output is NI

Initially, a variable X is defined in the global scope with the string value 'iNeuron'.

Then, a function func is defined. Inside this function, the global keyword is used to declare that X refers to the global variable X defined outside the function.

When the func function is called, it assigns the value 'NI' to the global variable X.

After calling func(), when print(X) is executed, it prints the value of the modified global variable X, which is now 'NI'. The global keyword inside the func function ensures that the assignment to X inside the function affects the global variable 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

1. This code will print iNeuron

Initially, a variable X is defined in the global scope with the string value 'iNeuron'.

Then, a function func is defined. Inside this function, another function nested is defined. Within the nested function, there is a print statement that prints the value of X. However, X is not defined locally within nested, so Python looks for it in the enclosing scope, which is the func function.

When func() is called, it calls the nested() function.

Inside nested(), it prints the value of X, which is 'NI' because X is defined as 'NI' in the scope of the func function.

After calling func(), when X is evaluated in the global scope (X on its own), it prints the value of the global variable X, which remains 'iNeuron'. The local variable X defined within the func function doesn't affect the global variable 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()

1. This code will produce an error in Python 3. The error will be:

SyntaxError: no binding for nonlocal 'X' found

In Python 3, the nonlocal keyword is used to indicate that a variable is not local to the current function scope but is in an outer (enclosing) scope. However, in this code snippet, there is no variable X in an outer scope of the nested function. The X variable is defined only in the func function. Therefore, when nonlocal X is encountered inside the nested function, Python raises a SyntaxError because it can't find a binding for X in an outer scope.

To fix this error, nonlocal X should refer to a variable X that is defined in an outer scope of the nested function. If X is supposed to be a local variable within the func function and modified within the nested function, X should be declared as nonlocal within the func function, not the nested function.