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

**>>> X = 'iNeuron'**

**>>> def func():**

**print(X)**

**>>> func()**

Ans-The output of the code is 'iNeuron'. This is because the variable X is defined outside of the function, and the function is simply printing the value of X.

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

**>>> X = 'iNeuron'**

**>>> def func():**

**X = 'NI!'**

**>>> func()**

**>>> print(X)**

Ans-The code will output 'iNeuron'. When the function func() is called, it creates a new local variable 'X' with the value 'NI!', but this variable is separate from the global variable 'X'. So, when the print statement outside the function is executed, it prints the value of the global variable 'X', which is 'iNeuron'.

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

**>>> X = 'iNeuron'**

**>>> def func():**

**X = 'NI'**

**print(X)**

**>>> func()**

**>>> print(X)**

Ans-The code will print 'NI' followed by 'iNeuron'. When the function func() is called, it creates a new local variable 'X' with the value 'NI' and prints its value. However, this local variable is different from the global variable 'X'. When the print statement outside the function is executed, it prints the value of the global variable 'X', which is 'iNeuron'.

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

**>>> X = 'iNeuron'**

**>>> def func():**

**global X**

**X = 'NI'**

**>>> func()**

**>>> print(X)**

Ans-The output of the code is 'NI'. This is because the function uses the global keyword to indicate that it is modifying the global variable X, rather than creating a local variable with the same name. Therefore, when the function is called, it changes the value of the global variable X to 'NI'. The print statement outside the function then prints the new value of 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**

Ans-The output of the code is 'NI'. Inside the function, the variable X is assigned the value 'NI'. Then, a nested function is defined, which simply prints the value of X. When the nested function is called, it prints 'NI'. Finally, the outer function is called, which assigns the value 'NI' to X and then calls the nested function. Therefore, when the nested function is called, it prints 'NI'. The value of X outside the function remains unchanged, so it still has the 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()**

Ans-The code will output 'Spam'. The nonlocal keyword is used to indicate that the variable 'X' belongs to the nearest enclosing scope, which is the function func(). So, when nested() modifies the variable 'X' using the nonlocal keyword, it modifies the 'X' variable of the parent function func(). As a result, when 'X' is printed within func(), it will display the modified value 'Spam'.