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

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

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

**print(X)**

**>>> func()**

The output of the code will be:

iNeuron

In this code, X is a string with the value 'iNeuron'. The func function is then defined, which prints the value of X. When the function is called using func(), the value of X is printed to the console, which is 'iNeuron'.

This code demonstrates how a variable can be defined outside of a function and be used within a function. In this case, the variable X is defined outside of the func function and is accessible within the function. This is known as a global variable. If a local variable with the same name as a global variable is defined within a function, the local variable will shadow the global variable and its value will be used instead.

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

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

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

**X = 'NI!'**

**>>> func()**

**>>> print(X)**

The output of the code will be:

iNeuron

In this code, X is a string with the value 'iNeuron'. The func function is then defined, which has a local variable X with the value 'NI!'. When the function is called using func(), the local variable X is defined within the function, but it does not affect the global variable X. After the function is called, the value of the global variable X is printed to the console using print(X), which is still 'iNeuron'.

This code demonstrates how a local variable with the same name as a global variable can be defined within a function, and that it will shadow the global variable, but its value will not affect the value of the global variable. In this case, the local variable X is defined within the func function, and its value is 'NI!', but it does not affect the value of the global variable X, which remains 'iNeuron'.

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

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

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

**X = 'NI'**

**print(X)**

The output of the code will be:

NI

In this code, X is a string with the value 'iNeuron'. The func function is then defined, which has a local variable X with the value 'NI'. When the function is called using func(), the value of the local variable X is printed to the console using print(X), which is 'NI'.

This code demonstrates how a local variable with the same name as a global variable can be defined within a function, and that it will shadow the global variable. In this case, the local variable X is defined within the func function, and its value is 'NI'. When the function is called, the value of the local variable is used instead of the value of the global variable. After the function is called, the value of the global variable X remains unchanged and is still 'iNeuron'.

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

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

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

**global X**

**X = 'NI'**

**>>> func()**

**>>> print(X)**

The output of the code will be:

NI

In this code, X is a string with the value 'iNeuron'. The func function is then defined, which has a local variable X with the value 'NI'. However, the global keyword is used within the function to indicate that the X referred to in the function is the global variable X and not a local variable with the same name.

When the function is called using func(), the value of the global variable X is changed to 'NI' within the function. After the function is called, the value of the global variable X is printed to the console using print(X), which is 'NI'.

This code demonstrates how the global keyword can be used within a function to indicate that a variable is the global variable and not a local variable with the same name. In this case, the value of the global variable X is changed within the function and remains changed after the function is called.

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

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

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

**X = 'NI'**

**def nested():**

**print(X)**

**nested()**

**>>> func()**

**>>> X**

The output of the code will be:

NI

iNeuron

In this code, X is a string with the value 'iNeuron'. The func function is then defined, which has a local variable X with the value 'NI'. Within the func function, another function nested is defined, which prints the value of the variable X.

When the func function is called using func(), the nested function is defined and its contents are executed. Within the nested function, the local variable X from the parent function func is used, so the output is 'NI'.

After the func function is called, the value of the global variable X is printed to the console using X, which is 'iNeuron'.

This code demonstrates how nested functions can access variables from their parent functions. In this case, the nested function is able to access and use the local variable X from its parent function func.

**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()**

The output of this code will be:

Spam

In this code, the func function is defined, which has a local variable X with the value 'NI'. Within the func function, another function nested is defined, which uses the nonlocal keyword to modify the value of X. The nonlocal keyword allows the nested function to modify the value of a variable in the nearest enclosing scope that is not global.

When the func function is called using func(), the nested function is defined and its contents are executed. Within the nested function, the value of X is changed to 'Spam' using the nonlocal keyword.

After the nested function is executed, the value of X from the func function is printed to the console using print(X), which is 'Spam'.

This code demonstrates how the nonlocal keyword can be used to modify a variable in the nearest enclosing scope that is not global. In this case, the nested function is able to modify the value of the variable X from its parent function func.