**Assignment 22**

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

**>>> X = ‘iNeuron’**

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

**print(X)**

**>>> func()**

**The code defines a variable X and assigns it the value 'iNeuron'. Then it defines a function named func() which simply prints the value of the variable X. Finally, the func() function is called.**

**When the func() function is called, it prints the value of X, which is 'iNeuron'. Therefore, the output of the code is:**

**Copy code**

**iNeuron**

**This is because the variable X is defined outside of the func() function, which means it is in the global namespace and can be accessed by any function within the same namespace. So, when func() is called, it can access the value of X and print it.**

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

**>>> X = ‘iNeuron’**

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

**X = ‘NI!’**

**>>> func()**

**>>> print(X)**

**The code defines a variable X and assigns it the value 'iNeuron'. Then it defines a function named func() which creates a new variable X inside the function and assigns it the value 'NI!'. Finally, the func() function is called and then the value of the variable X is printed outside the function.**

**When the func() function is called, it creates a new variable X which only exists within the function scope. This means that the X variable outside of the function is not affected and remains 'iNeuron'. Therefore, the output of the code is:**

**Copy code**

**iNeuron**

**This is because when print(X) is executed outside the function, it refers to the global variable X, which has not been modified by the func() function.**

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

**>>> X = ‘iNeuron’**

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

**X = ‘NI’**

**print(X)**

**>>> func()**

**>>> print(X)**

**The code defines a variable X and assigns it the value 'iNeuron'. Then it defines a function named func() which creates a new variable X inside the function and assigns it the value 'NI'. Within the func() function, the value of the X variable is printed. Finally, the func() function is called and then the value of the variable X is printed outside the function.**

**When the func() function is called, it creates a new variable X which only exists within the function scope. Therefore, when print(X) is executed inside the function, it refers to the local variable X, which has the value 'NI'. Therefore, the output of the code is:**

**Copy code**

**NI**

**iNeuron**

**This is because the first call to print(X) prints the value of the local variable X inside the func() function, which is 'NI'. The second call to print(X) prints the value of the global variable X, which has not been modified by the func() function and remains 'iNeuron'.**

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

**>>> X = ‘iNeuron’**

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

**global X**

**X = ‘NI’**

**>>> func()**

**>>> print(X)**

**The code defines a variable X and assigns it the value 'iNeuron'. Then it defines a function named func() which uses the global keyword to declare that the variable X inside the function refers to the global variable X. The function then assigns the value 'NI' to the global variable X. Finally, the func() function is called and then the value of the variable X is printed outside the function.**

**When the func() function is called, it declares that the variable X inside the function refers to the global variable X. This means that when the function assigns the value 'NI' to X, it is modifying the global variable X rather than creating a new local variable. Therefore, the output of the code is:**

**Copy code**

**NI**

**This is because when print(X) is executed outside the function, it refers to the global variable X, which has been modified by the func() function to have the value '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**

**The code defines a variable X and assigns it the value 'iNeuron'. Then it defines a function named func() which creates a new variable X inside the function and assigns it the value 'NI'. The func() function also defines a nested function named nested() which prints the value of the variable X. Finally, the func() function is called, which in turn calls the nested() function, and then the value of the variable X is printed outside the functions.**

**When the func() function is called, it creates a new variable X which only exists within the function scope. Within the nested() function, the value of the X variable is printed, but since there is no local X variable in nested(), the function looks for the nearest enclosing scope that contains a variable named X. In this case, that would be the func() function, which has a local variable X with the value 'NI'. Therefore, the output of the code is:**

**Copy code**

**NI**

**iNeuron**

**This is because the first call to print(X) prints the value of the local variable X inside the nested() function, which is 'NI'. The second call to print(X) prints the value of the global variable X, which has not been modified by any of the functions and remains '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()**

**The code defines a function named func() which creates a local variable X and assigns it the value 'NI'. The func() function also defines a nested function named nested() which uses the nonlocal keyword to declare that the variable X inside the function refers to the non-local variable X in the enclosing func() function. The nested() function then assigns the value 'Spam' to the non-local variable X. Finally, the nested() function is called from within func() and then the value of the variable X is printed.**

**When the nested() function is called, it declares that the variable X inside the function refers to the non-local variable X in the enclosing func() function. This means that when the function assigns the value 'Spam' to X, it is modifying the non-local variable X rather than creating a new local variable. Therefore, the output of the code is:**

**Copy code**

**Spam**

**This is because when print(X) is executed outside the nested() function, it refers to the local variable X inside the func() function, which has been modified by the nested() function to have the value 'Spam'. Note that the nonlocal keyword is not available in Python 2.x, and attempting to use it would result in a SyntaxError.**