**ANSWERS**

**Q1. What is the result of the code, and explain?**

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

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

**print(X)**

**>>> func()**

**Ans.** The output of the above code is “**iNeuron”.** In this code, a global variable **X** is assigned the value **'iNeuron'**. Then, a function **func()** is defined, which prints the value of the global variable **X**. When the function **func()** is called, it prints the value of **X**, which is **'iNeuron'**.

**Q2. What is the result of the code, and explain?**

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

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

**X = 'NI!'**

**>>> func()**

**>>> print(X)**

**Ans.** The output of the above code is **“iNeuron”**. In this code, a global variable X is assigned the value 'iNeuron'. Then, a function func() is defined, within which a local variable X is assigned the value 'NI!'. When the function func() is called, it creates a local variable X but doesn't affect the value of the global X. When print(X) is executed outside the function, it prints the value of the global X, which remains 'iNeuron'.

**Q3. What does this code print, and why?**

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

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

**X = 'NI'**

**print(X)**

**>>> func()**

**>>> print(X)**

**Ans.** Please find below the output of the above code:

**NI**

**iNeuron**

In this code, a global variable X is assigned the value 'iNeuron'. Then, a function func() is defined, within which a local variable X is assigned the value 'NI' and printed. When the function func() is called, it prints the local X, which is 'NI'. However, the global variable X remains unchanged, so when print(X) is executed outside the function, it prints the value of the global X, which is 'iNeuron'.

**Q4. What output does this code produce? Why?**

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

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

**global X**

**X = 'NI'**

**>>> func()**

**>>> print(X)**

**Ans.** The output of the above mentioned code is **“NI”**. In this code, a global variable X is assigned the value 'iNeuron'. Then, a function func() is defined, within which the global keyword is used to indicate that the function should use the global variable X. When the function func() is called, it modifies the global X to 'NI'. As a result, when print(X) is executed outside the function, it prints the modified value of the global X, which is 'NI'.

**Q5. What about this code—what’s the output, and why?**

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

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

**X = 'NI'**

**def nested():**

**print(X)**

**nested()**

**>>> func()**

**>>> X**

**Ans.** Please find below the output of the above code:

**NI**

**iNeuron**

In this code, a global variable X is assigned the value 'iNeuron'. Then, a function func() is defined, within which a local variable X is assigned the value 'NI'. Inside the func() function, another function nested() is defined, which prints the value of the local variable X. When nested() is called within func(), it prints the local X, which is 'NI'. However, the global variable X remains unchanged, so when print(X) is executed outside the function, it prints the value of the global X, which is 'iNeuron'.

**Q6. 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 output of the above mentioned code is **“Spam”**. In this code, a function func() is defined. Inside func(), a local variable X is assigned the value 'NI'. Then, another function nested() is defined within func(), and the nonlocal keyword is used to indicate that the inner function should modify the nearest enclosing variable named X, which is the one in the func() function. When nested() is called, it modifies the local X in the func() function to 'Spam'. As a result, when print(X) is executed within func(), it prints the modified value of the local X, which is 'Spam'.