**Assignment22:  
Submitted By: Bhupendra Singh**

**Qus1. What is the result of the code, and explain?**

>>> X = 'iNeuron'  
>>> def func():  
print(X)  
>>> func()  
**Ans:** The Result of this code is iNeuron. it is because the function intially looks for the variable X in its local scope but since there is no local variable X, it returns the value of global variable X i.e iNeuron. Please find the output below:

X **=** 'iNeuron'

**def** func():

print(X)

func()

iNeuron

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

>>> X = 'iNeuron'  
>>> def func():  
X = 'NI!'  
>>> func()  
>>> print(X)  
**Ans:** The Result of this code is NI! because the function initially looks for the variable X in its local scope. If X is not available, then it checks for variable X in the global scope. Since here the X is present in the local scope, it directly prints the value NI! and does not look for a variable in global scope.

X **=** 'iNeuron'

**def** func():

X **=** 'NI!'

print(X)

func()

NI!

**Qus3. What does this code print, and why?**

>>> X = 'iNeuron'  
>>> def func():  
X = 'NI'  
print(X)  
>>> func()  
>>> print(X)  
**Ans:** The output of the code is NI and iNeuron. X=NI is in the local scope of the function func(). Hence the function prints the x value as NI. X = 'iNeuron' is in the global scope. Hence print(X) prints output as iNeuron

X **=** 'iNeuron'

**def** func():

X **=** 'NI'

print(X)

func()

print(X)

NI

iNeuron

**Qus4. 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. The global keyword allows a variable to be accessible in the current scope. Since we are using global keyword inside the function func, it directly accesses the variable in X in global scope and changes its value to NI. Hence the output of the code is NI as shown below.

X **=** 'iNeuron'

**def** func():

**global** X

X **=** 'NI'

func()

print(X)

NI

**Qus5. 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 and iNeuron. Output of func() is 'NI' because it has a variable X as 'NI' in its local scope whereas Output of X is 'iNeuron' because it refers to variable X that is having global scope instead of referring to a variable having a local scope in a function.

X **=** 'iNeuron'

**def** func():

X **=** 'NI'

**def** nested():

print(X)

nested()

func()

X

NI

'iNeuron'

**Qus6. 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 code is Spam. nonlocal keyword in python is used to declare a variable as not local.Hence the statement X = "Spam" is modified in the global scope. Hence the output of print(X) statement is Spam

**def** func():

X **=** 'NI'

**def** nested():

**nonlocal** X

X **=** 'Spam'

nested()

print(X)

func()

Spam