#### 1. 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's because the function initially looks for the variable X in its local scope. But since there is no local variable X, its returns the value of global variable x i.e. iNeuron

I/P:

X **=** 'iNeuron'

**def** func():

print(X)

func()

O/P:

iNeuron

#### 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 prints the value NI!

I/P:

X **=** 'iNeuron'

**def** func():

X **=** 'NI!'

print(X)

func()

O/P:

NI!

#### 3. 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

I/P:

X **=** 'iNeuron'

**def** func():

X **=** 'NI'

print(X)

func()

print(X)

O/P:

NI

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. 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 access the variable in X in global scope and changes its value to NI. Hence the output of the code is NI.

I/P:

X **=** 'iNeuron'

**def** func():

**global** X

X **=** 'NI'

func()

print(X)

O/P:

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  
**Ans:**

The output of the code is NI. The reason for this output is if a function wants to access a variable, if it’s not available in its localscope. It looks for the variable in its global scope. Similarly here also function nested looks for variable X in its global scope. Hence the output of the code is NI

I/P:

X **=** 'iNeuron'

**def** func():

X **=** 'NI'

**def** nested():

print(X)

nested()

func()

X

O/P:

NI

#### 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 output of the code is Spam. The 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.

I/P:

**def** func():

X **=** 'NI'

**def** nested():

**nonlocal** X

X **=** 'Spam'

nested()

print(X)

func()

O/P:

Spam