# Assingnment 22 Solutions

## 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 intially looks for the variable **X** in its local scope,But since there is no local variable **X** , its returns the value of globle variable **X** ie **iNeuron**

In [2]:
```
X = 'iNeuron'
def func():
    print(X)
func()
```

```
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 cide 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 globle scope,Since here the **X** is present in the local scope.it prints the value **NI!**

In [3]:
```
X = 'iNeuron'
def func():
    X = 'NI!'
    print(X)
func()
```

```
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**

In [4]:
```
X = 'iNeuron'
def func():
    X = 'NI'
    print(X)
func()
print(X)
```

```
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**.

In [5]:
```
X = 'iNeuron'
def func():
    global X
    X = 'NI'
func()
print(X)
```

```
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 its 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**

In [6]:
```
X = 'iNeuron'
def func():
    X = 'NI'
    def nested():
        print(X)
    nested()
func()
X
```

```
NI
```

Out[6]:
```
'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()

**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**

In [8]:
```
def func():
    X = 'NI'
    def nested():
        nonlocal X
        X = 'Spam'
    nested()
    print(X)
func()
```

```
Spam
```