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

X = 'iNeuron'

def func():

print(X)

func()

```
X = 'iNeuron'
def func():
    print(X)
func()
```

```
iNeuron
```

Answer: the result of this code is "iNeuron' because it is function look for varible x,but there is no local varaible x,it return the value of variable x.

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

X = 'iNeuron'

def func():

X = 'NI!'

func() print(X)

Answer: The Result of this code is NI! because the function looks for the variable X in its local scope. If X is not available, then it checks for variable X in the global scope. 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!
```

3. What does this code print, and why?

```
        X = 'iNeuron'
        def func():
X = 'NI'
print(X)
        func()
        print(X)
```

Answer: The result of this code is iNeuron, NI. x='NI' is the local code of the function func(). print x values as NI .. x='iNeuron' is the globle scope . print(x) - - print output as 'iNeuron'

```
x='iNeuron'
def fuc():
  x="NI"
print(x)
func()
print()
```

```
iNeuron
NI!
```

4. What output does this code produce? Why?

```
        X = 'iNeuron'
        def func():
global X
X = 'NI'
        func()
        print(X)
```

Answer: The result is this code is 'NI'.because it is global keyword allows a variable 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.

```
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

Answer: 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'
```

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()
```

Answer: 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
```