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

```
>>> X = 'iNeuron'

>>> def func():

print(X)

>>> func()
```

Ans:

it will print the value of x which is iNeuron

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

```
>>> X = 'iNeuron'

>>> def func():

X = 'NI!'

>>> func()
```

Ans:There wont be anyoutput as we are not returning anything and also not priniting anything as well

```
>>> print(X)
```

Ans:Output will be iNeuron

3. What does this code print, and why?

```
>>> X = 'iNeuron'

>>> def func():

X = 'NI'

print(X)
```

>>> func()

Ans:This willl give output as the NI

>>> print(X)

Ans :Now we get the output as the iNeuron

4. What output does this code produce? Why?

>>> X = 'iNeuron'

>>> def func():

global X

X = 'NI'

>>> func()

Ans:Once we call func here it wont print or return anything but it will update the global variable

>>> print(X)

Ans:here we get the output as the NI as we have updated the gloable variable by calling the function previously

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:We get the output as 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:

Indentation is missing in question .It looks like below

```
>>> def super():
...     x='outside'
...     def nested():
...         nonlocal x
...         x='spam'
...     nested()
...     print(x)
```

If we change the value of a nonlocal variable, the changes appear in the local variable as well

Nonlocal variables are used in nested functions whose local scope is not defined. This means that the variable can be neither in the local nor the global scope.

So here the output will be Spam