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

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

In [1]:

```python
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)
```

In [2]:

```python
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)
```

In [3]:

```python
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)
```

In [4]:

```
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
```

In [5]:

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

```
NI
```

Out[5]:

```
'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()
```

In [6]:

```
def func():
    X = 'NI'
    def nested():
        nonlocal X
        X = 'Spam'
    nested()
    print(X)
func()
```

```
Spam
```

In [ ]: