Assignment-22

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

```
>>> X = ‘iNeuron’
>>> def func():
print(X)


>>> func()
```

Ans. The function func() can access the global variable X defined outside the function. When the function is called, it prints the value of the global X, which is 'iNeuron'.

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

```
>>> X = ‘iNeuron’
>>> def func():
X = ‘NI’
Print(X)


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

Ans. Although the function func() creates a local variable X and assigns it 'NI!', this doesn't affect the global X. When we print X after calling the function, we get the unchanged global value 'iNeuron'.

3. What does this code print, and why?

```
>>> X = ‘iNeuron’
>>> def func():
X = ‘NI’
print(X)
>>> func()
>>> print(X)
```

Ans. This code prints:

‘NI’

‘iNeuron’

Explanation: The function creates a local X and prints it ('NI'). The global X remains unchanged, so the second print statement outputs 'iNeuron'.

4. What output does this code produce? Why?

```
>>> X = ‘iNeuron’
>>> def func():
global X
X = ‘NI’

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

Ans. By using the global keyword, the function explicitly tells Python to use the global X variable. When the function assigns 'NI' to X, it modifies the global variable. Therefore, printing X after calling the function shows the new value '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. When func() is called, it creates a local variable X with value 'NI'. The nested function can access this local variable from its enclosing scope. So when nested() prints X, it prints 'NI'. The global X remains 'iNeuron' but is not accessed or printed.

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. This code will raise a SyntaxError Explanation: The nonlocal statement is trying to reference X from an outer scope, but X is defined in the same function where nonlocal is used. nonlocal is used to reference variables from outer (but not global) scopes when nested functions want to modify them. Since there's no X in an outer (non-global) scope, this results in a syntax error.