**Assignment 22**

Q1. 1. What is the result of the code, and explain?

>>>> X = ‘iNeuron’

>>>> def func():

print(X)

>>>> func()

Sol.

**Output:** iNeuron

Q2. What is the result of the code, and explain?

>>>> X = ‘iNeuron’

>>>> def func():

X = 39;

func()

print(X)

Sol.

X = 'iNeuron'

def func():

global X # global keyword means modify the global variable value

X = 39

func()

print(X)

**Output:** 39

Q3. What does this code print, and why?

>>>> X = ‘iNeuron’

>>>> def func():

X = ‘NI’

print(X)

>>>> func()

>>>> print(X)

Sol. The variable **X** is initially assigned the value **'iNeuron'**.

The function **func()** is defined.

Inside the function **func()**, a new local variable **X** is created and assigned the value **'NI'**.

The line **print(X)** inside the function **func()** prints the value of the local variable **X**, which is **'NI'**.

The function **func()** is called.

After the function call, the line **print(X)** outside the function **func()** prints the value of the global variable **X**, which is still **'iNeuron'**.

**Output:**

NI

iNeuron

Q4. What output does this code produce? Why?

X = ‘iNeuron’

def func():

global X

X = ‘NI’

>>>> func()

>>>> print(X)

Sol. The variable **X** is initially assigned the value **'iNeuron'**.

The function **func()** is defined.

Inside the function **func()**, the **global** keyword is used to indicate that we want to modify the global variable **X** rather than creating a new local variable with the same name.

The line **X = 'NI'** inside the function assigns the value **'NI'** to the global variable **X**.

The function **func()** is called.

After the function call, the line **print(X)** prints the value of the modified global variable **X**, which is **'NI'**.

**Output:**

NI

Q5. What about this code—what’s the output, and why?

X = ‘iNeuron’

def func():

X = ‘NI’

def nested():

print(X)

nested()

>>>> func()

>>>> X

Sol.

The variable **X** is initially assigned the value **'iNeuron'**.

The function **func()** is defined.

Inside the function **func()**, a new local variable **X** is created and assigned the value **'NI'**.

Inside **func()**, another nested function **nested()** is defined.

Inside the **nested()** function, the line **print(X)** is executed, which prints the value of the local variable **X** within the scope of **func()**. Since no local variable **X** is defined within **nested()**, Python looks for the nearest enclosing scope, which is the **func()** scope, and finds the value **'NI'**. Therefore, it prints **'NI'**.

The **nested()** function is called from within **func()**, resulting in the execution of the **print(X)** statement within **nested()**.

After the execution of **func()**, the line **print(X)** outside the function **func()** is executed, which prints the value of the global variable **X**, which is **'iNeuron'**.

**Output:**

NI

iNeuron

Q6. 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()

Sol.

The function **func()** is defined.

Inside **func()**, a local variable **X** is created and assigned the value **'NI'**.

Inside **func()**, another nested function **nested()** is defined.

Inside the **nested()** function, the **nonlocal** keyword is used to indicate that we want to modify the variable **X** in the nearest enclosing scope, which is the scope of **func()**.

The line **X = 'Spam'** inside **nested()** assigns the value **'Spam'** to the variable **X** in the scope of **func()**.

The **nested()** function is called from within **func()**, resulting in the execution of the line **X = 'Spam'**.

After the execution of **nested()**, the line **print(X)** within **func()** is executed, which prints the value of the modified variable **X**, which is **'Spam'**.

**Output:**

Spam