Python Programming Structure (Contd.)

Programming Control – Flow of Control

- Sequential programming
- Conditional (Decision making)
  - If
  - If else
  - If elif else
  - Nested if
- Iterative (Looping)
  - While loop
  - For loop

Note: Indentation is an important aspect in Python. It is the space before the code line within its code block. In case of conditional statements or looping statements, we need to put at least one space on every line following the conditional or iterative block. When the block is over, the rest of the lines should be in place as the first line of the program.

If statement –

```
if condition:
        Statement
```

If else statement –

```
if condition:
        Statement 1
else:
        Statement 2
```

If Elif Else statement –

```
if condition 1:
        Statement 1
elif condition 2:
        Statement 2
else:
        Statement 3
```

Nested if statement –

```
if condition 1:
        if condition 2:
                Statement 1
        else:
                Statement 2
else:
        Statement 3
```

Practice programs:

Leap Year Check
Write a program to check whether a given year is a leap year or not.

Number Sign Classification
Take a number as input and determine if it is positive, negative, or zero.

Grading System
Input marks (0 to 100) and display the grade based on the following:

- 90-100: A
- 80-89: B
- 70-79: C
- 60-69: D
- Below 60: F

Odd or Even and Divisibility by 3
Take an integer input and check if it is even. If it is, check if it is divisible by 3.

Largest of Three Numbers
Input three numbers and determine the largest using nested if statements.

Vowel or Consonant
Accept an alphabet from the user and determine if it is a vowel or a consonant. (Assume lowercase input and check for valid alphabets only.)

Electricity Bill Calculator
Input the number of units consumed and calculate the bill using the following rules:

- First 100 units: ₹1.5/unit
- Next 100 units: ₹2.5/unit
- Beyond 200 units: ₹3/unit
  Add ₹50 as meter charge for all.

Simple Calculator
Take two numbers and an operator (+, -, *, /, %) and perform the operation. Use if-elif-else.

Check Triangle Validity
Input three sides of a triangle and check if it forms a valid triangle using the triangle inequality theorem.

Nested Login System
Write a program that asks for a username and password. If the username is correct, it checks for the password. Display appropriate messages based on the result.

Loops in Python
Loops are used to execute a block of code repeatedly as long as a condition is true.

Types of Loops in Python:
1. **for loop**
2. **while loop**
3. **Nested loops**

1. for Loop
The for loop is used to iterate over a **sequence** (like list, tuple, string, or range).
Syntax:
```
for variable in sequence:
    # code block
```
Example using range():
```
for i in range(5):
    print(i)
```
Output:
0
1
2
3
4

Example iterating over a list:
```
fruits = ['apple', 'banana', 'cherry']
for fruit in fruits:
    print(fruit)
```

For loop –

```
for var in range (initial value, ending value+1, step value):
        body

n=int(input("Enter a no"))
f=1
for a in range (1, n+1):
        f=f*a
print("factorial of ", n, " =", f)

#printing 1st 10 odd nos.
for a in range (1, 20, 2):
        print(a, end=' ') #printing in same line

#output
1 3 5 7 9 11 13 15 17 19
```

2. while Loop

The while loop continues executing as long as the **condition is True**.

Syntax:

```
while condition:
    # code block
```

Example:

```
i = 1
while i <= 5:
    print(i)
    i += 1
```

Output:

```
1
2
3
4
5
```

While loop –

```
#Initialization
while condition:
        body
        #re-initialization

n = int(input("enter a no"))
f=1
a=1
while a<=n:
        f=f*a
        a=a+1
print("factorial of ", n, " =", f)
```

Nested Loops

You can place a loop inside another loop.

Example:

```
for i in range(3):
    for j in range(2):
        print(f"i={i}, j={j}")
```

Example programs:

1. Print numbers from 1 to 10

```
for i in range(1, 11):
    print(i)
```

2. Print even numbers from 1 to 20

```
for i in range(1, 21):
    if i % 2 == 0:
        print(i)
```

3. Print the multiplication table of a number

```
num = int(input("Enter a number: "))
for i in range(1, 11):
    print(f"{num} x {i} = {num * i}")
```

4. Find the sum of first 10 natural numbers

```
total = 0
for i in range(1, 11):
    total += i
print("Sum:", total)
```

5. Print characters of a string one by one

```
text = "Python"
for char in text:
    print(char)
```

6. Count digits in a number (using a while loop)

```
num = int(input("Enter a number: "))
count = 0
while num > 0:
    count += 1
    num //= 10
print("Number of digits:", count)
```

7. Reverse a number using a while loop

```
num = int(input("Enter a number: "))
rev = 0
while num > 0:
    digit = num % 10
    rev = rev * 10 + digit
    num //= 10
print("Reversed number:", rev)
```

8. Print all elements in a list

```
colors = ["red", "green", "blue"]
for color in colors:
    print(color)
```

9. Print square of numbers from 1 to 5

```
for i in range(1, 6):
    print(f"{i} squared is {i**2}")
```

10. Simple countdown using a while loop

```
count = 5
while count > 0:
    print(count)
    count -= 1
```

Jump statements in Loops

Let us study some example codes given below:

Case 1:

```
For I in range (1,10):
        Print (I)
Print(”Loop ends”)
```

1 2 3 4 5 6 7 8 9
Loop ends

Case 2:

```
For I in range (1,10):
        If I==5:
                Break
        Print(I)
Print(”Loop ends”)
```

1 2 3 4
Loop ends

Case 3:

```
For I in range (1,10):
        If I==5:
                Continue
        Print(I)
Print(”Loop ends”)
```

1 2 3 4 6 7 8 9
Loop ends

In the above three codes, we can see some differences in the output, though for each loop, the set iteration was 1 to 9 i.e. nine times. In case 1, we found that the loop completes all 9 iteration but in case 2 and case 3, either the number of iterations is not the same as set earlier or the print statement executed does not match the number of iterations. Why does this happen?

The reason is due to the following two statements –

1. Break statement – it ends the loop prematurely and places the control out of the loop
2. Continue statement – it skips one iteration in the loop and places the control at the beginning of the loop

In Python, break and continue are control flow statements used inside loops to alter their normal behaviour.

break Statement - Immediately exits the loop **completely**, regardless of the loop condition. So, the break statement ends the loop as soon as it is encountered, even if the loop condition is True.

Example:

```
for i in range(1, 10):
    if i == 5:
        break  # Loop stops completely when i is 5
    print(i)
```

Output:

```
1
2
3
4
```

continue Statement - Skips the **current iteration** and moves to the **next iteration** of the loop. When you want to **skip** some values but continue looping, the continue statement is used.

Example:

```
for i in range(1, 6):
    if i == 3:
        continue  # Skips printing 3
    print(i)
```

Output:

```
1
2
4
5
```

Some more examples:

Using the break statement in the Prime Check program

```
num = 29
is_prime = True
for i in range(2, int(num**0.5) + 1):
    if num % i == 0:
        is_prime = False
        break  # Not a prime number; no need to check further
if is_prime:
    print(num, “is a prime number.")
else:
    print(num, “is not a prime number.")
```

Using the continue statement to Skip Even Numbers in a List

```
numbers = [10, 15, 22, 33, 42, 55]
for num in numbers:
   if num % 2 == 0:
      continue  # Skip even numbers
   print("Odd number: ” , num)
```

Output:

Odd number: 15
Odd number: 33
Odd number: 55

Example programs to show the use of the break statement

```
N=int(input(“Enter a no.”))
if N==1:
        print(“Non-prime”)
else:
        for a in range(2,N/2+1):
                if N%a==0:
                        print(“Non-prime”)
                        break
        if a>N/2:
                print(“Prime”)
```

```
N=int(input(“Enter a no.”))
C=0
While(N>0):
        D=N%10
        N=N//10
        if D==0:
                C=1
                break
if C==0:
        print(“Not a Duck no”)
else:
        print(“Duck no”)
```

****