Topic 1.1.5 – Control Statements

A program's **control flow** is the order in which the program's code executes.

The control flow of a Python program is regulated by conditional statements, loops, and function calls.

Python has *three* types of control structures:

- **Sequential** - default mode
- **Selection** - used for decisions and branching
- **Repetition** - used for looping, i.e., repeating a piece of code multiple times.

1. Sequential

Sequential statements are a set of statements whose execution process happens in a sequence. The problem with sequential statements is that if the logic has broken in any one of the lines, then the complete source code execution will break.

Example :-

```
## This is a Sequential statement
a=20
b=10
c=a-b
print("Subtraction is : ",c)
```

Output:-

```
Subtraction is :  10
```

2. Selection/Decision control statements

In Python, the selection statements are also known as *Decision control statements* or *branching statements*.

The selection statement allows a program to test several conditions and execute instructions based on which condition is true.

Some Decision Control Statements are:

- Simple if
- if-else
- nested if

- if-elif-else

Simple if: *If statements* are control flow statements that help us to run a particular code, but only when a certain condition is met or satisfied. A *simple if* only has one condition to check.

Syntax:-

```
if test expression:

   statement(s)
```

Example Program :-

```
n = 10
if n % 2 == 0:
  print("n is an even number")
```

Output:-

```
n is an even number
```

if-else: The *if-else statement* evaluates the condition and will execute the body of if if the test condition is True, but if the condition is False, then the body of else is executed.

Syntax:-

```
if test expression:

   Body of if

else:

   Body of else
```

Example Program:-

```
# Program checks if the number is positive or negative
# And displays an appropriate message

num = 3
# Try these two variations as well.
# num = -5
# num = 0
if num >= 0:
   print("Positive or Zero")
else:
   print("Negative number")
```

Output:-

```
Positive or Zero
```

The elif Statement:

The **elif** statement allows you to check multiple expressions for TRUE and execute a blockof code as soon as one of the conditions evaluates to TRUE.

Syntax

```
if expression1:
    statement(s)
elif expression2:
    statement(s)
elif expression3:
    statement(s)
else:
    statement(s)
```

Example

```
amount=int(input("Enter amount: "))

if amount<1000:
    discount=amount*0.05
    print ("Discount",discount)
elif amount<5000:
    discount=amount*0.10
    print ("Discount",discount)
else:
    discount=amount*0.15
    print ("Discount",discount)
    print ("Net payable:",amount-
discount)
```

Input and Output:-

```
  Enter amount: 600
  Discount 30.0
  Net payable: 570.0

  Enter amount: 3000
  Discount 300.0
  Net payable: 2700.0

  Enter amount: 6000
  Discount 900.0
Net payable: 5100.0
```

Nested If statement:-

There may be a situation when you want to check for another condition after a conditionresolves to true. In such a situation, you can use the nested **if** construct.

In a nested **if** construct, you can have an **if...elif...else** construct inside another **if...elif...else** construct.

Syntax

```
if expression1:
    statement(s)
    if expression2:
        statement(s)
    elif expression3:
        statement(s)
    else
        statement(s)
elif expression4:
    statement(s)
else:
    statement(s)
```

Example:-

```
num=int(input("enter number"))
  if num%2==0:
       if num%3==0:
            print ("Divisible by 3 and 2")else:
            print ("divisible by 2 not divisible by 3")
  else:
       if num%3==0:
            print ("divisible by 3 not divisible by 2")else:
print   ("not Divisible by 2 not divisible by 3")
```

Input and Output:-

```
  enter number8
 divisible by 2 not divisible by 3

 enter number15
 divisible by 3 not divisible by 2

 enter number12
 Divisible by 3 and 2

 enter number5
not Divisible by 2 not divisible by 3
```

3. Repitition Statements:-

A loop statement allows us to execute a statement or group of statements multiple times.

Python has two loop statements:

- while loop
- for loop

While Loop statement:-

The while loop in Python is used to iterate over a block of code as long as the test expression (condition) is true.

We generally use this loop when we don't know the number of times to iterate beforehand.

Syntax:-

```
while test_expression:
        statement(s)
```