Conditional and Iterative Statements

As per CBSE curriculum
Class 11

Chapter- 4

By-
Neha Tyagi
PGT (CS)
KV 5 Jaipur(II Shift)
Jaipur Region

Introduction

- Generally a program executes from starting point to end point.
- Some program does not execute in order.
- As per the requirement, execution order of the program can be changed and it is also possible to execute a program repeatedly.
- Python provides control structures to manage the order of execution of a program, which are if-else, for, while and jump statements like break, continue.

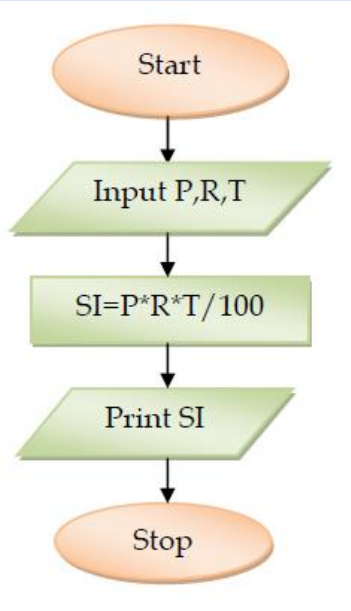
Types of statements

- In Python, statements are of 3 types-
  - » Empty Statements
    - pass
  - » Simple Statements (Single Statement)
    - name=input (“Enter your Name “)
    - print(name) etc.
  - » Compound Statements
    - <Compound Statement Header>:
      <Indented Body containing multiple simple statements/compound statements>
    - Here, Header line starts with the keyword and ends at colon (:).
    - The body consists of more than one simple Python statements or compound statements.

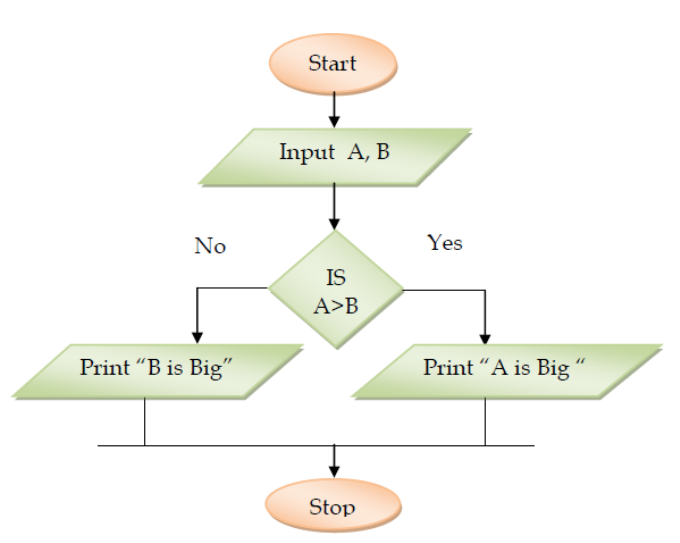
Statement Flow Control

- In a program, statements executes in sequential manner or in selective manner or in iterative manner.

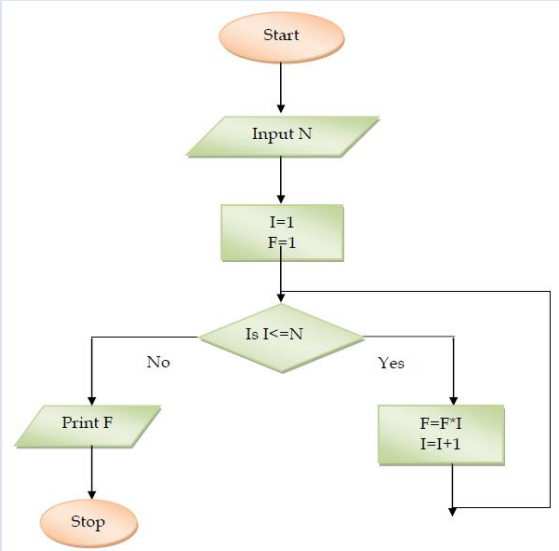
Sequential



Selective



Iterative



Program Logic Development Tool

A program has following development stages-

1. Identification of the problem
2. Analysis of problem
3. Writing Algorithm or Designing Flowchart
4. Writing Code
5. Testing and Debugging
6. Implementation
7. Maintenance

Algorithm

- A process or set of rules to be followed in problem-solving operations is an algorithm.

For ex-

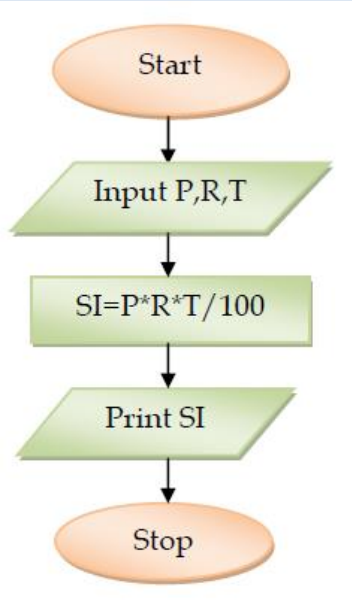
Algorithm to add two numbers is as under-

1. **Input** *First Number*
2. **Input** *Second Number*
3. **Add** *First Number* with *Second Number* and store into *Third number.*
4. **Display** *Third number*

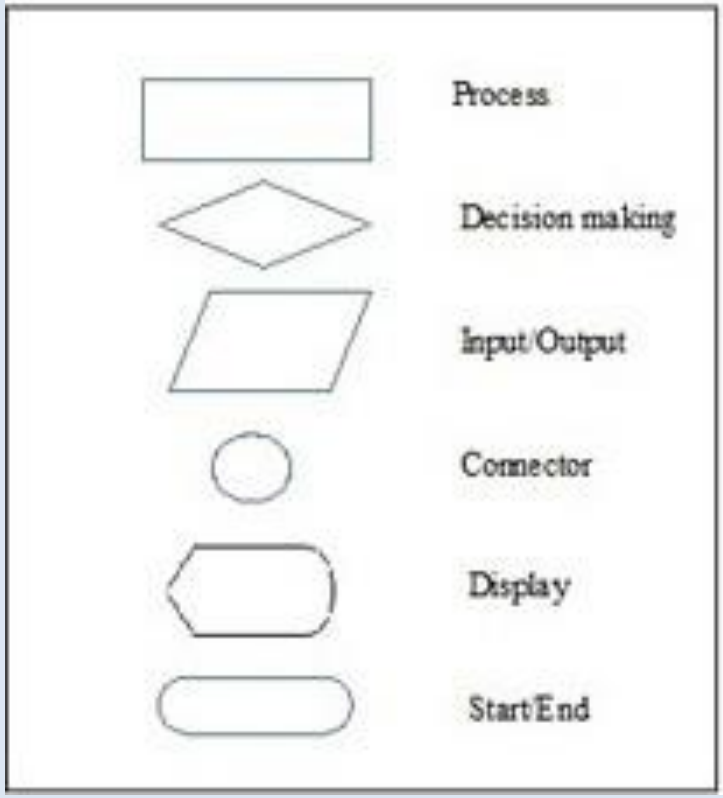
Flowcharts

- A flowchart is a graphical representation of an algorithm, workflow or process. The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows.

For ex- flowchart to calculate simple interest is as under-



Symbols to be used in Flowchart

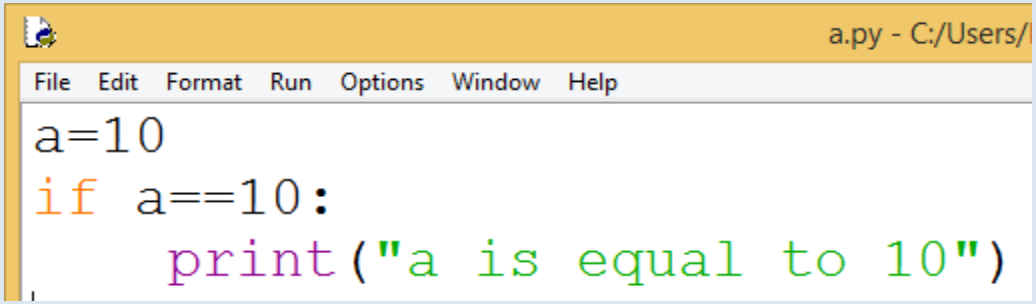


if Statement

- In Python, if statement is used to select statement for processing. If execution of a statement is to be done on the basis of a condition, if statement is to be used. Its syntax is-

```
if <condition>:
    statement(s)
```

like -

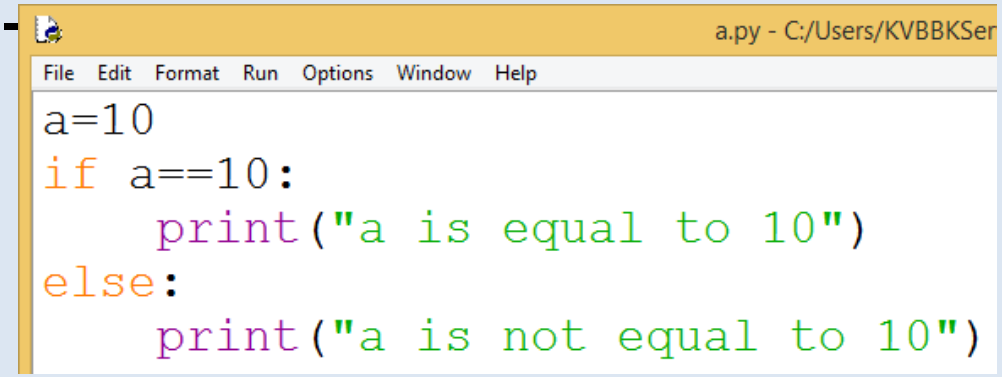


if-else Statement

- If out of two statements, it is required to select one statement for processing on the basis of a condition, if-else statement is to be used. Its syntax is-

```
if <condition>:
        statement(s) when condition is true
else:
        statement(s) when condition is false
```

like -



if-elif Statements

- If out of multiple statements, it is required to select one statement for processing on the basis of a condition, if-elif statement is to be used. Its syntax is-

```
if <condition1>:
        statement(s) when condition1 is true
elif <condition2>:
        statement(s) when condition2 is true
elif <condition3>:
        statement(s) when condition3 is true
else
```

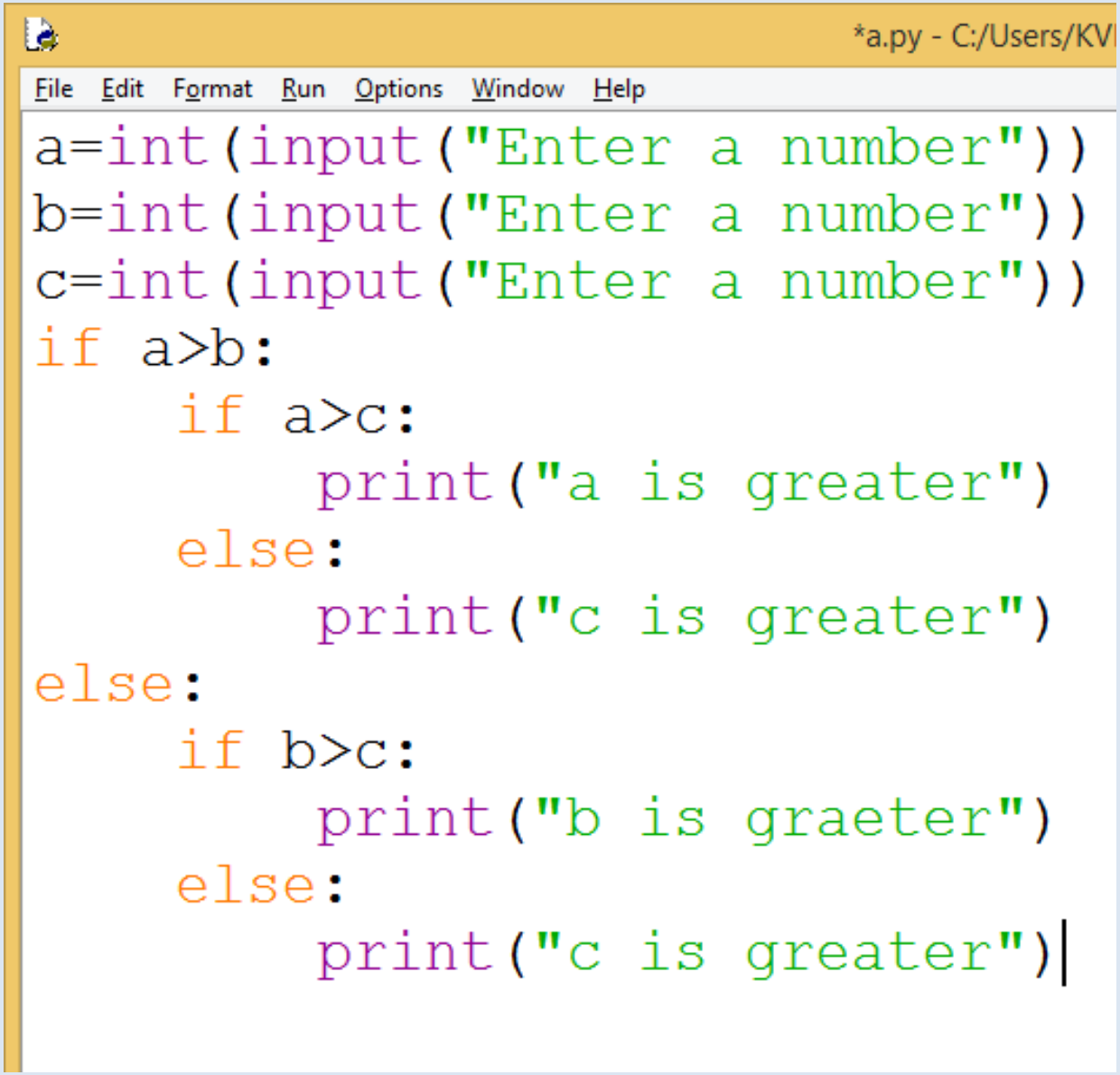
like -

a.py - C:/Users/KVBBKServer

File Edit Format Run Options Window Help

```
a=int(input("Enter a number"))
if a==10:
    print("a is equal to 10")
elif a==20:
    print("a is equal to 20")
elif a==30:
    print("a is equal to 30")
else:
    print("Again Give the number")
```

Nested If -else

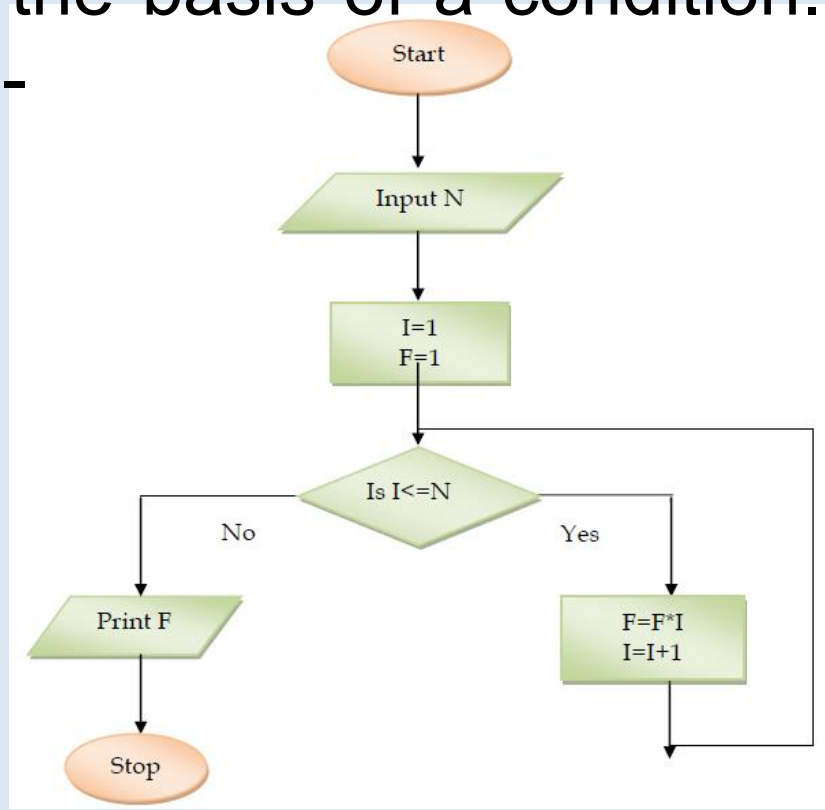


Loop/ Repetition/ Iteration

These control structures are used for repeated execution of statement(s) on the basis of a condition.

Loop has 3 main components-

1. Start (initialization of loop)
2. Step (moving forward in loop)
3. Stop (ending of loop)



Python has following loops-

– for loop
– while loop

range () Function

- **In Python**, an important function is range( ). its syntax is-

range (<lower limit>,<upper limit>)

If we write - range (0,5)

Then a list will be created with the values [0,1,2,3,4] i.e. from lower limit to the value one less than ending limit.

range (0,10,2) will have the list [0,2,4,6,8].

range (5,0,-1) will have the list [5,4,3,2,1].

in and not in operator

- *in* operator-

  3 in [1,2,3,4] will return ***True***.

  5 in [1,2,3,4] will return ***False***.

– *not in* operator-

  5 not in [1,2,3,4] will return ***True.***

```
for a in [1,2,3]:
    print(a)
    print(a*a)
```

Table of a number by For loop

Syntax of For Loop

```
for <var> in <sequence>:
        <statements to repeat>
```

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

Output

```
Enter a number10
10 x 1 = 10
10 x 2 = 20
10 x 3 = 30
10 x 4 = 40
10 x 5 = 50
10 x 6 = 60
10 x 7 = 70
10 x 8 = 80
10 x 9 = 90
10 x 10 = 100
```

Table of a number by while loop

Syntax of While Loop

While <LogicalExpression>:
<loop body with increment
or decrement>

```
n=int(input("Enter a number"))
c=1
while c<11:
    print(n,"x",c,"=",c*n)
    c=c+1
```

Start

Stop

Step

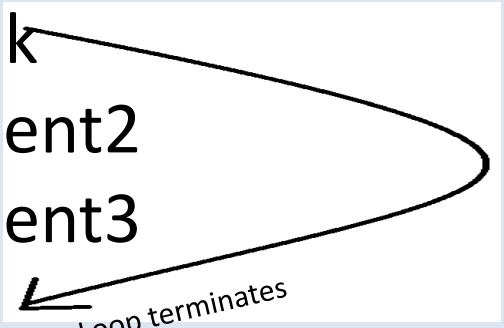
Output

```
Enter a number5
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
```

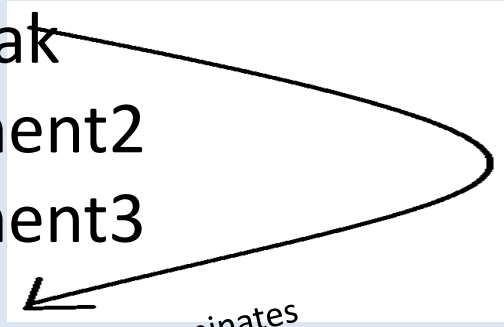
Jump Statements

break Statement

```
while <test-condition>:
        statement1
        if <condition>:
            break
        statement2
        statement3
Statement4
statement5
```




```
for <var> in <sequence>:
        statement1
        if <condition>:
            break
        statement2
        statement3
Statement4
statement5
```



Jump Statements

break Statement

```
n=int(input("Enter a number"))
c=1
while c<11:
    if c==5:
        break
    print(n,"x",c,"=",c*n)
    c=c+1
```

Output

```
Enter a number4
4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
```

```
n=int(input("Enter a number"))
c=1
for c in range(1,11):
    if c==5:
        break
    print(n,"x",c,"=",c*n)
```

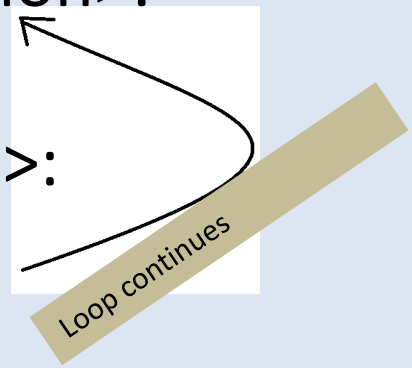
Output

```
Enter a number5
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
>>>
```

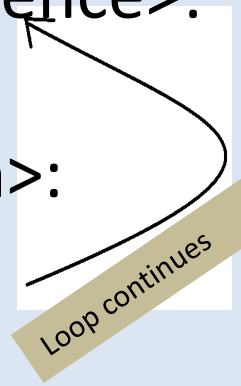
Jump Statements

continue Statement

```
while <test-condition>:
        statement1
        if <condition>:
            continue
        statement2
        statement3
Statement4
statement5
```




```
for <var> in <sequence>:
        statement1
        if <condition>:
            continue
        statement2
        statement3
Statement4
statement5
```



Jump Statements

continue Statement

```
n=int(input("Enter a number"))
for c in range(1,11):
    if c==5:
        continue
    print(n,"x",c,"=",c*n)
```

```
n=int(input("Enter a number"))
c=0
while c<11:
    c=c+1
    if c==5:
        continue
    print(n,"x",c,"=",c*n)
```

Output of both the programs

```
Enter a number5
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
5 x 11 = 55
>>>
```

Nested Loop

```
n=int(input("Enter the number"))
for r in range(1,n+1):
    for c in range(1,r+1):
        print("*", end="")
    print("")
```

OUTPUT

```
Enter the number5
*
**
***
****
*****
```

Assignments

1. WAP to find greatest among three numbers.
2. WAP to print the result on the basis of marks entered of a student.
3. WAP to print counting up to n.
4. WAP to print even numbers up to n.
5. WAP to print odd numbers up to n.
6. WAP to print Fibonacci series.
7. WAP to calculate x^n.
8. WAP to calculate n!.
9. WAP to print different patterns.

Thank you

- Please follow us on our blog-

www.pythontrends.wordpress.com