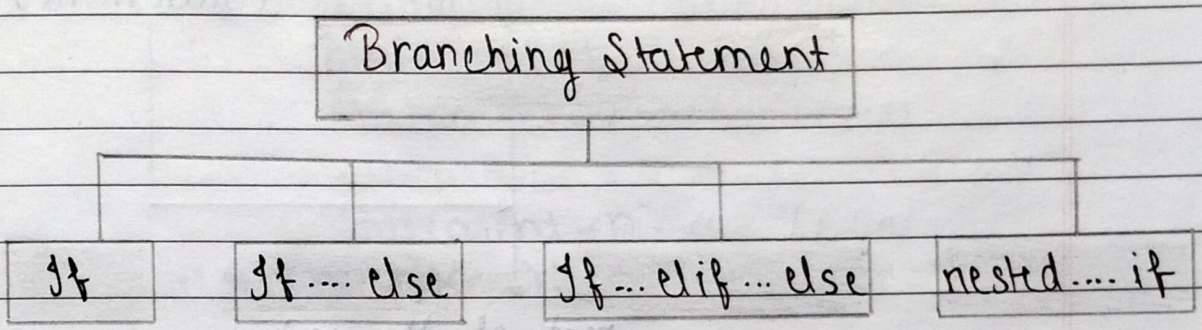
Chap-2: Selection / Branching Statement.

Selection or Conditional Branching Statement are used to execute one or more block of code when the expression is true.
There are four types of Branching Statement.



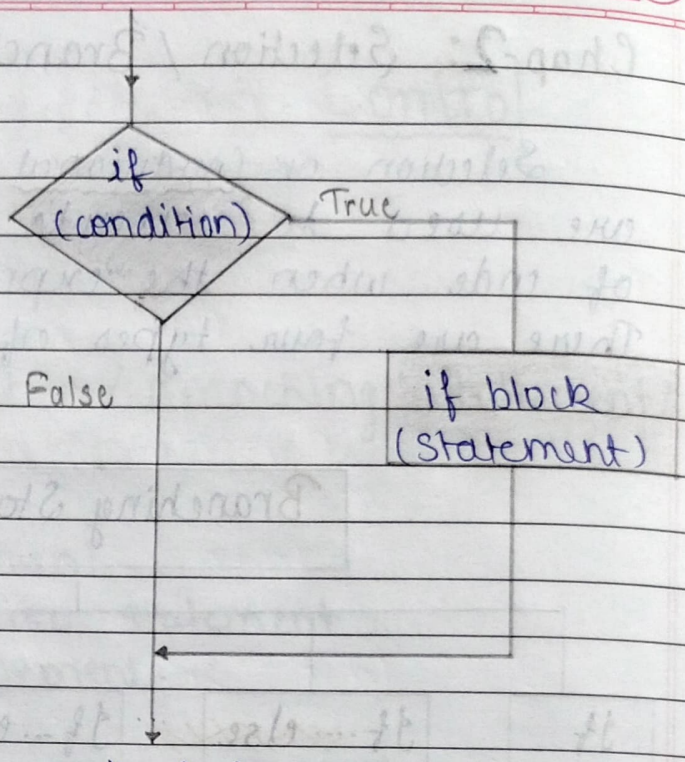
1) If statement:

The if statement is the simplest form conditional statement.

In this statement, 'If' is passed with certain condition, when execution is starts, it will check the condition of if, if the condition is true then statement under if i.e. if block is executated and if false the rest of the program is executed.

- Syntax:

```
if (condition):
    statement-1
    statement-2
    ---- Statement-n
rest of the program.
```



Example:

1)
```
if 5<10:
    print ("5 is less than 10")
print ("5 is not equal to 10")
```

2)
```
if a<0;
    print ("a is negative no.")
print ("a is positive no.")
```

2] If ---- else statement: (it is a two-way statement)
In if --- else statement an extra else block combined with if statement is also provided.

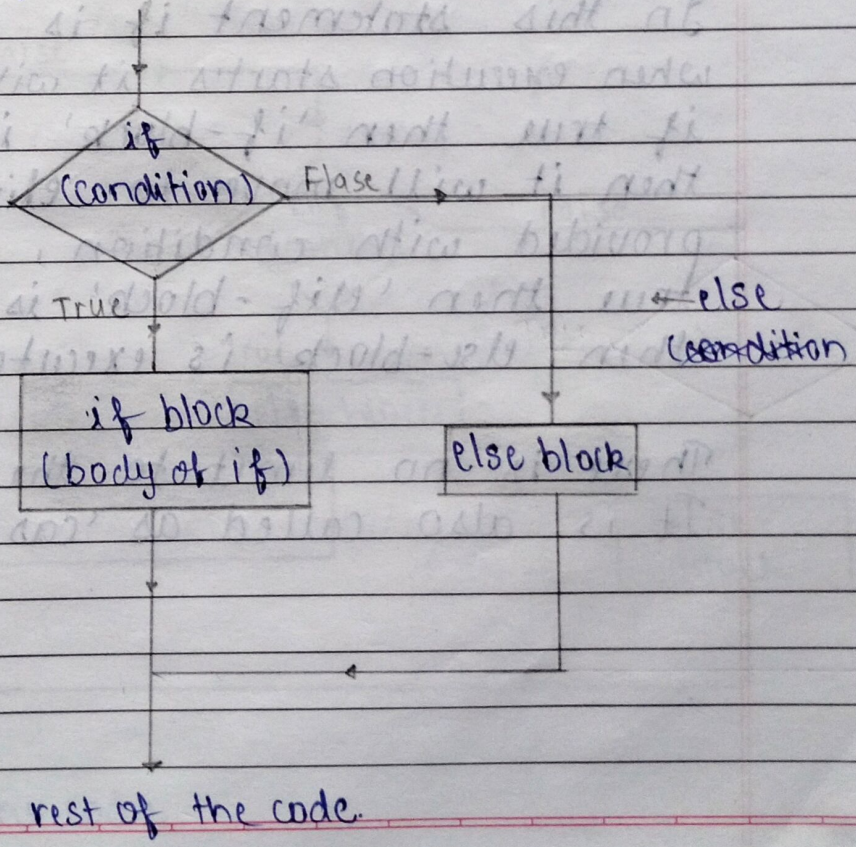
In this statement if is provided with some condition, when execution starts, it will check the condition of if, and if the condition is true then if-block is executed and if false

then it moves to else block, further if else cannot be provided with condition. then it will check condition if it is true. the else block is executed otherwise and rest of the code is get executed.

- Syntax:

```
if (condition):
    statement-1
    statement-2
    ---  -  -  -
    statement-n
else ():
    statement-1
    statement-2
    ---  -  -  -
    statement-n
rest of the code.
```

- Flow Chart:



Example:

1)
```
n = eval(input("Enter the no. : "))
if n%2==0:
    print("no. is even")
else:
    print("no. is odd")
```

2)
```
opr = input("Enter operator : ")
if opr=='+':
    print("addition is ", a+b)
else :
    print("subtraction is", a-b)
```

3] If --- elif --- else :

It is a multiway decision statement. 'elif' is basically abbreviation of 'else if'.

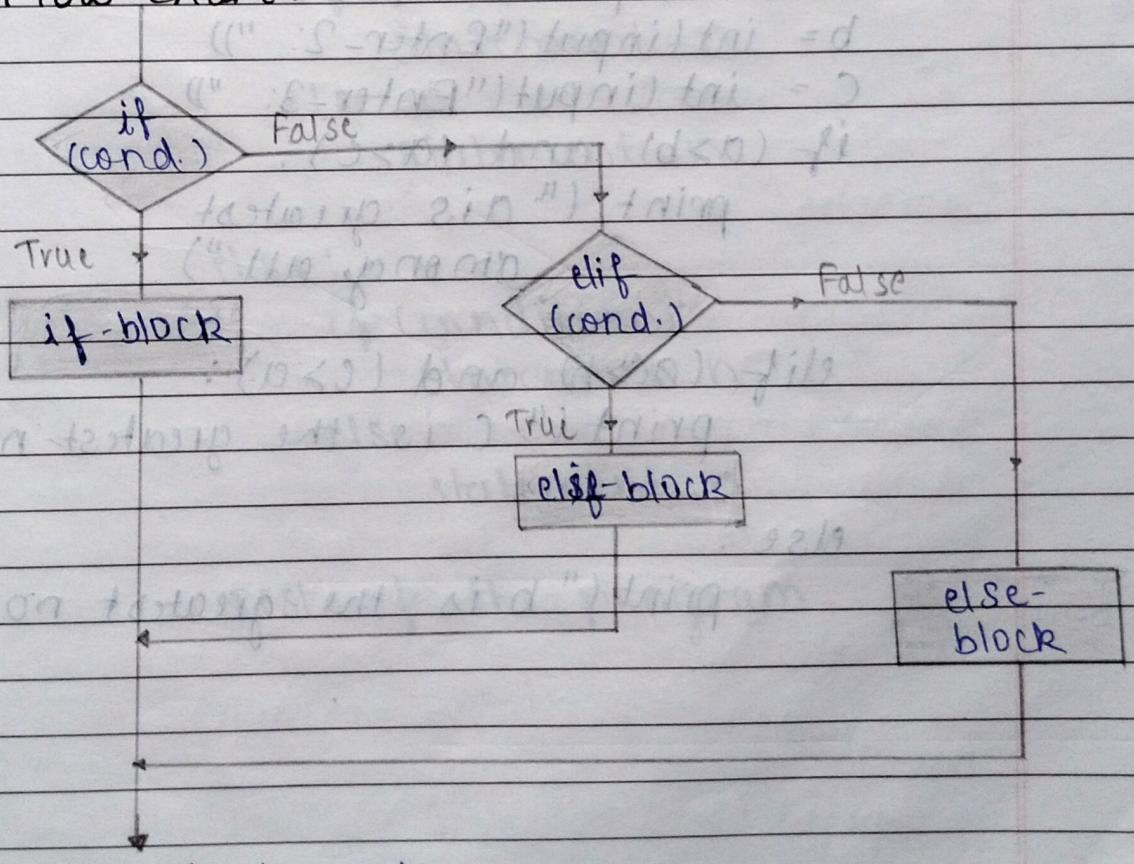
In this statement if is provided with condition when execution starts it will check the condition if true then 'if-block' is executed if false then it will move to elif which is also provided with condition, when condition is true then 'elif-block' is executed if false then else-block is executed.

There is no limit to the elif statement It is also called as 'cascade' or 'ladder'

- Syntax:

```
if (condition):
    statement-1
    statement-2
    - - - - - -
    statement-n
elif (condition):
    statement-2
    statement-3
    - - - - - -
    statement-n
else:
    statement

rest of the code
```

- Flow-chart:



Example:

1)
```
Print("Enter your mark's:")
mark = m = float(input())
if m>=75:
    print("Distinction")
elif m>=60:
    print("First class")
elif m>=50:
    print("Second class")
elif m>=40:
    print("Third class")
else:
    print("Fail")
```

2) #Program to find greatest among three no.

```
a = int(input("Enter_1: "))
b = int(input("Enter_2: "))
c = int(input("Enter_3: "))
if (a>b) and (a>c):
    print("a is greatest
          among all")

elif (a<c>b) and (c>a):
    print("c is the greatest no.")

else:
    print("b is the greatest no.")
```

4] Nested if, if--else, if--elif--else:

When one Conditional statement is present in another Conditional statement then we can say that it is Nested Conditional Statement.

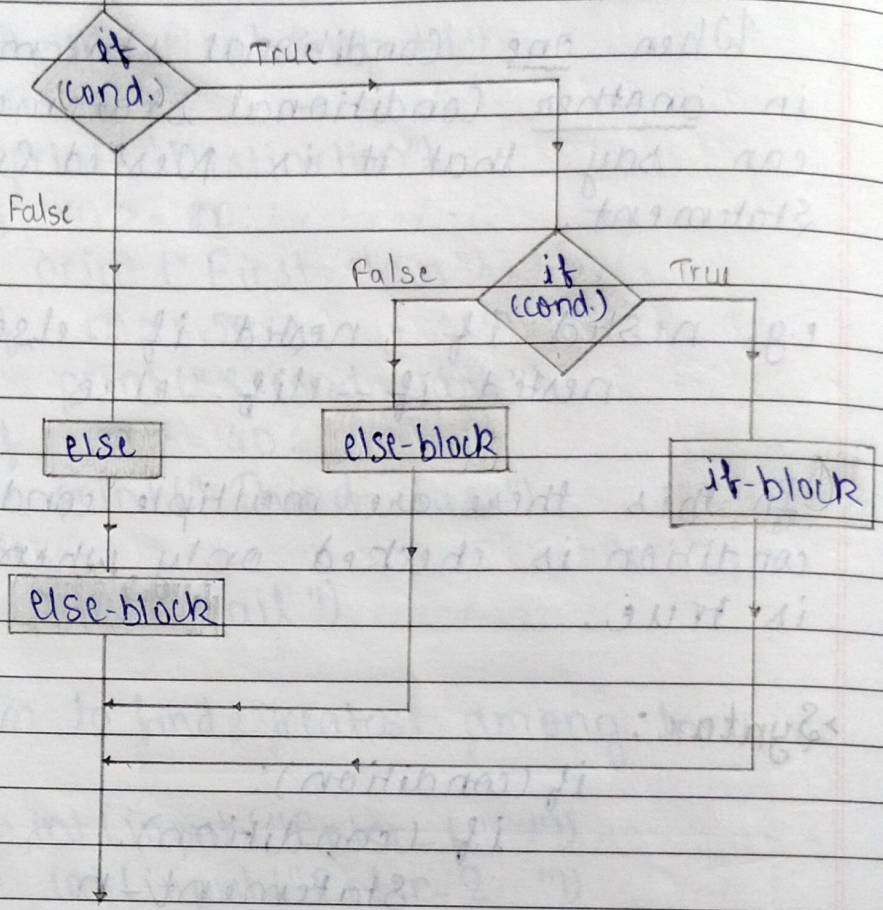
e.g. nested if, nested if--else,
nested if--elif--else

In this there are multiple conditions, inner condition is checked only when outer condition is true.

>Syntax:

```
if (condition):
    if (condition):
        Statement-n
    else:
        Statement-n

else:
    if (condition):
        Statement-n
    else:
        statement-n

rest of the program
```

Flow Chart:



Examples:

1) # program to find +ve or -ve or zero

```
a = int(input("Enter no.:"))
if a!=0:
    if a>0:
        print("no. is positive")
    else:
        print("no. is negative")
else:
    print("no. is zero")
```

2) # greatest among three.
assign three no.

```
a, b, c = 5, 7, 9
if a>b:
    if a>c:
        print ("a is the greatest")
    else:
        print ("c is the greatest")
else:#(a<b):
        if b>c:
            print ("b is the greatest")
        else:
            print("c is the greatest")
```

3) # nested if, even-odd

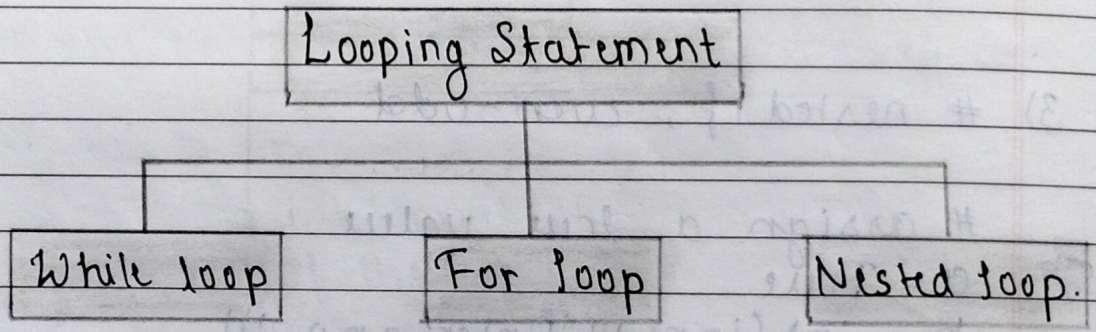
```
# assign a true value
a = True
b = int(input("Enter a no."))
# check the condition.
if a == True:
        if b%2 == 0:
            print ("no. is even")
        else:
            print("no. is odd")
print ("End of Program")
```

Chap. 3. Looping / Iterative Statements

Looping or Iteration is a technique that allows to execute a set of statement or a particular blocks of code repeatedly till the condition is true.

- Iteration: Reapeated execution of a set of statements or block of code is called Iteration.

There are three-types of Looping Statements



1] While Loop:

It is a simplest form of looping statement. It is used to iterate or repeat block of codes unless and untill the condition become False.

In this first initial value is declared, while is passed with certain condition, when execution starts it will check the condition if True then while-block is executed, this process

repeates, untill the condition is True.

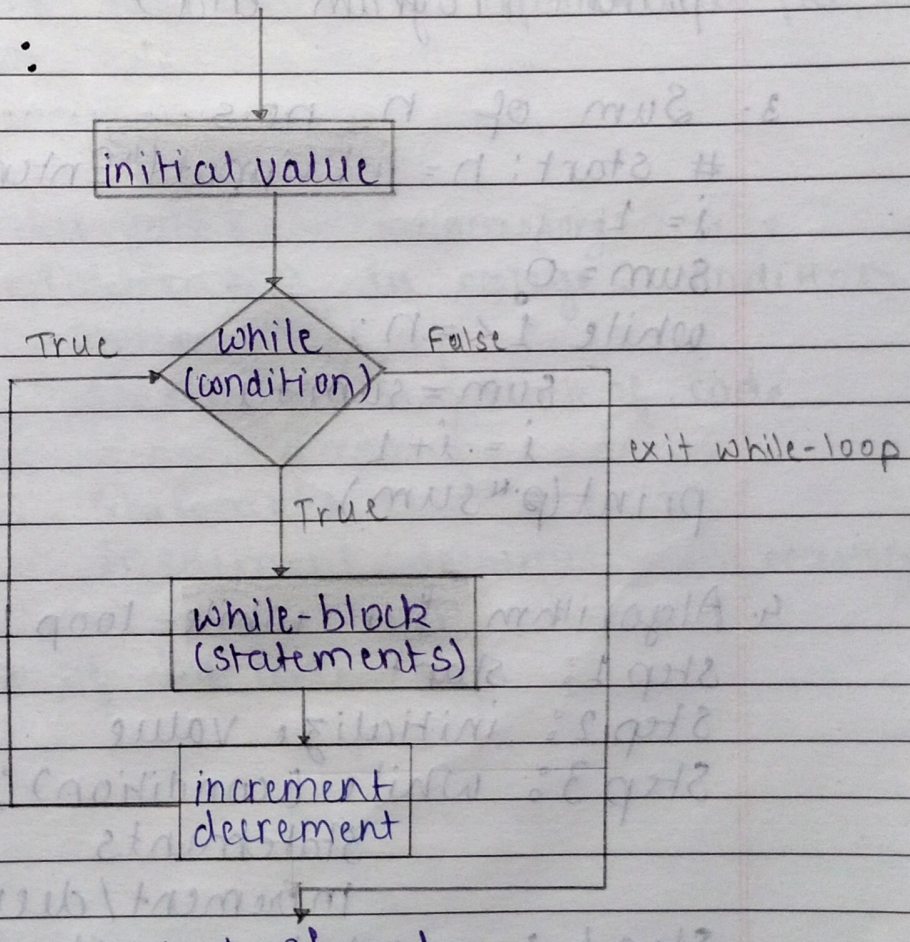
If condition is False, then it will exit the loop and 'rest of code' is executed.

- Syntax:

```
initial value
while (condition/expression):
    statement-1
    statement-2
    -- -- --
    statement-n
    (increment/decrement)
rest of code.
```

(block of code)

- Flow chart:



Example:

1. Program to print "Hello" five times.

```
#Start
i = 1
While i<=5:
    print("Hello")
print("End")
```

2. Program to print 1 to 10 digits

```
# Start
i = 0
While i<=10:
    print(i)
    i = i+1
print("Program end")
```

3. Sum of n no.s.

```
# start: n = int(input("Enter no. "))
i = 1
Sum = 0
while i<=n:
    sum = sum + i
    i = i+1
print("sum")
```

4. Algorithm for while-loop

Step 1: start
Step 2: initialize value
Step 3: while (condition):
        Statements
        increment/decrement
Step 4: print the result
Step 5: stop.

2] For Loop:

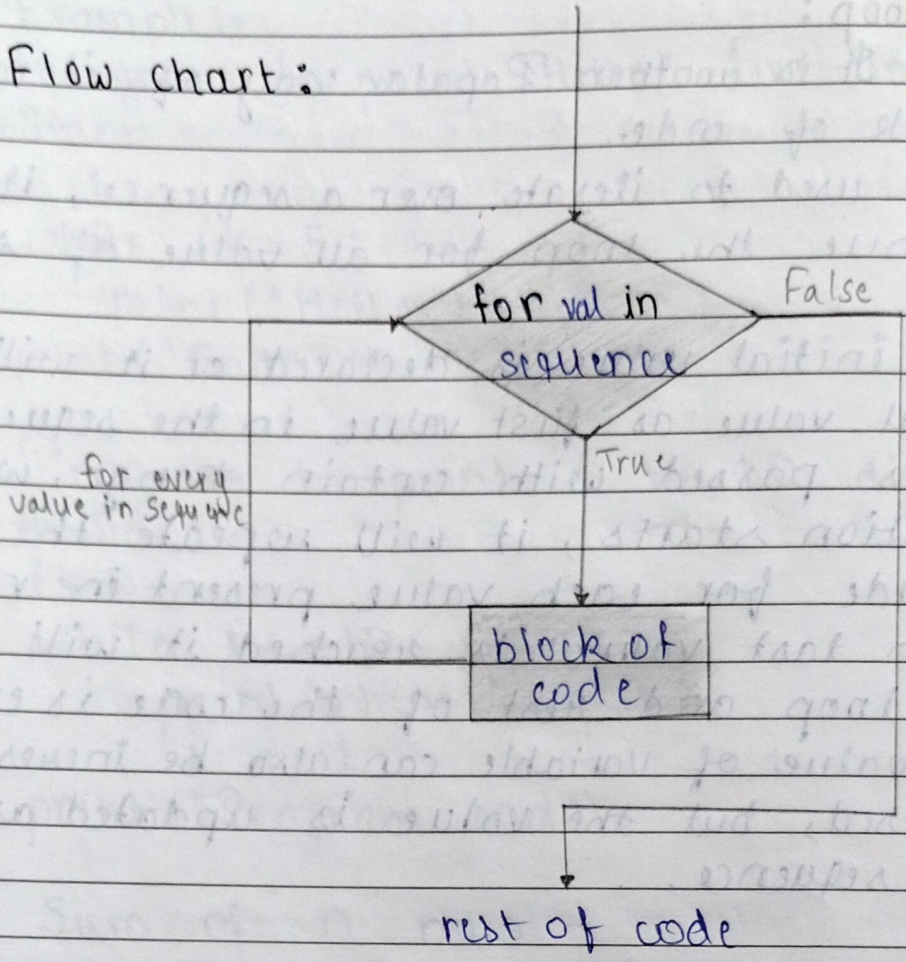
It is another Popular way of iterating a block of code.
It is used to iterate over a sequence, it will continue the loop for all value of sequence.

First initial value is declared or it will take initial value as first value in the sequence.
for is passed with certain range, when execution starts, it will repeate the block of code for each value present in range.
When last value is reached it will exit the loop and rest of the code is executed.
The value of variable can also be incresed or decresed, but the value is updated as per the sequence.

- Syntax:

```
initial value  — (not necessory)
for (variable) in range: condition
        Statement-1
        Statement-2
        ---  ---        } block of code
        Statement-n
        increment / decrement - not necessory
rest of the code
```

- Flow chart:



- Range function:

By using range function we can generate a particular sequence of number.

Syntax: range (start, stop, step-size)
(starting value) (ending value)

where, start = starting / initial value (0 by default)
stop = ending / last value
step-size = fixed increment or decrement.

Start Stop Stepsize

e.g. range (0, 10) will generate 10 no. 0 to 9

range (10) also generate 10 no. 0 to 9
(0 is taken by default)

range (1, 10) generate 9 no. 1 to 9

range (2, 10) generate 2 to 9 no. i.e. 8 no.
>>> last digit is always not considered in range, n-1 is considered.

range (1, 10, 2) will generate 1, 3, 5, 7, 9
there will be difference of 2 between each digit.

range (1, 10, 3) will generate 1, 4, 7, no.
difference of 3 between each digit.

- Example of For-loop.

1) # program to find sum of n. no.

```
n = int(input("Enter number : "))
sum = 0
for i in range(0, n+1):
    sum = sum+i
print(sum)
```

2) # program to print 'hello' 10 times.

```
for i in (0, 10):
    print("Hello")
print("end")
```

3) # Program to calculate factorial of n no.

```
n = int(input("Enter number : "))
fac = 1
for i in range(1, n+1):
    fac = fac * i
print(fac)
```

4) # Program to find sum of 1 to 10 no.

```
sum = 0
for i in range(1, 11):
    sum = sum + i
print(sum)
```

5) # program to display multiplication table.

```
n = int(input("Enter number : "))

for i in range(1, 11):
    table = n * i
    print(n, "x", i, "=", table)
print("end")
```

- Controling a loop: loop is controlled by two factors condition and counter.

1) Condition controlled loop:
When we do not know for how many no. of times the loop will execute then we use condition controlled loop.
It is also called as Sentinel controlled loop or indefinite loop.
It is normally used with while loop.

2) Counter Controlled loop: (range controlled)
When we know how many no. of time the loop will be executed then we use Count-or Controlled loop.
It is also called as definite loop.
It is normally used with for loop., it can also be used with while loop.

3] Nested-loop:

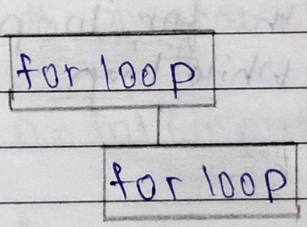
Nested loop are the loop structure in which one loop is present inside the outer loop.

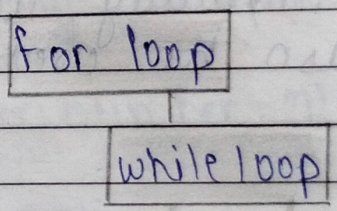
When execution start it execute the outer loop first and then enters the inner loop and then execute the inner loop completely and then switch back the outer loop until the condition becomes false or range got terminated. The inner-loop get executed multiple no. of time.

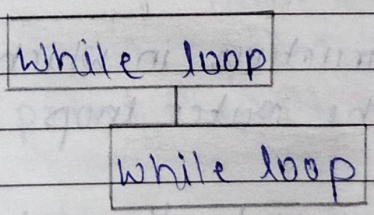
- Syntax:

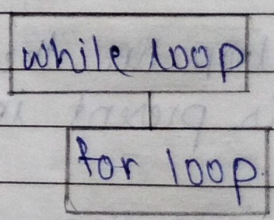
```
initial value
(outer loop):
    statement-1
    [(inner loop):
        statement-1
        statement-2
        - - - -
        statement-n
        , increment/decrement]
    statement-n
    increment/decrement

    rest of the code
```

Note: we can have four conditions:

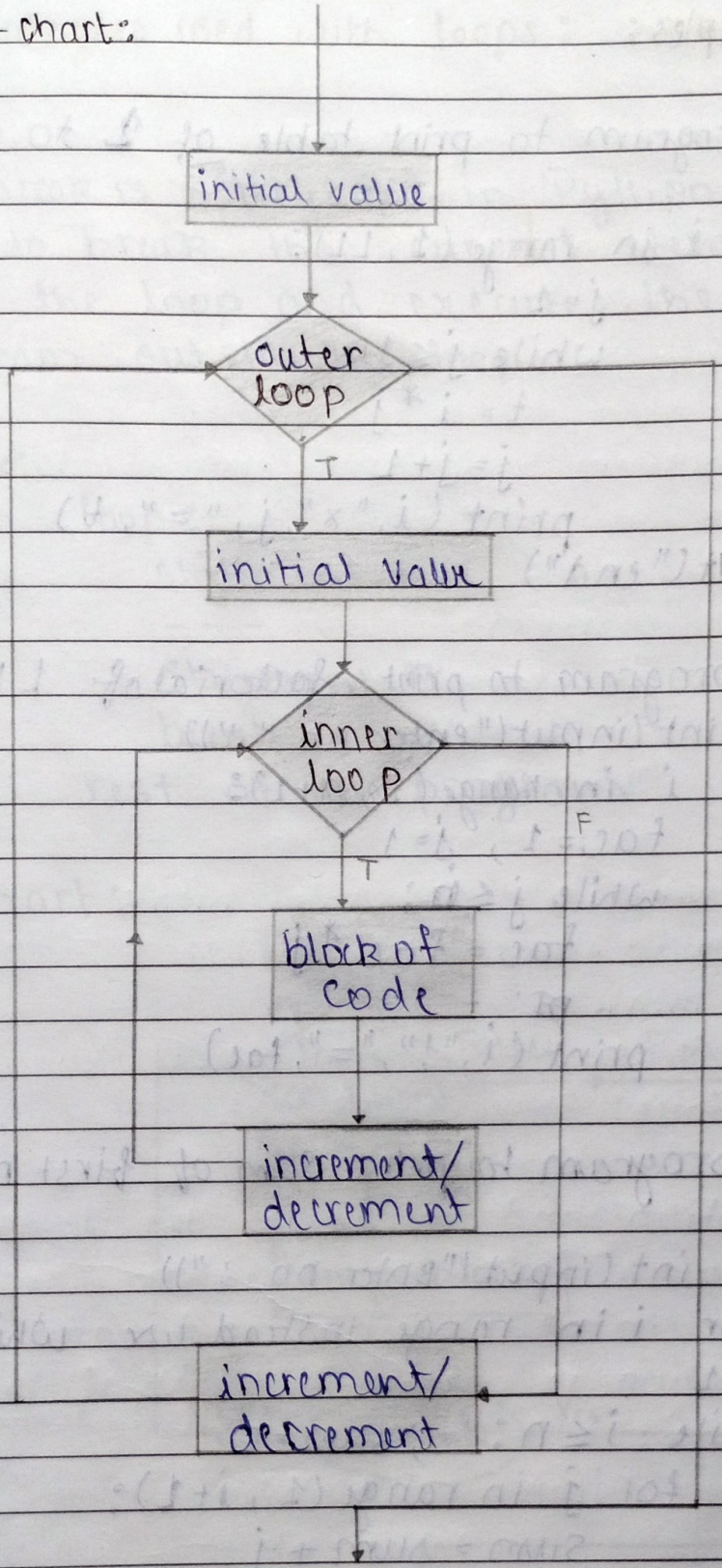








- Flow-chart:


initial value
outer loop
T
initial value
F
inner loop
F
T
block of code
increment/ decrement
increment/ decrement
rest of the code.

- Examples:

1) # Program to print table of 1 to 10

```
for i in range(1, 11):
    j = 1
    while j <= 10:
        t = i * j
        j = j + 1
    print(i, "x", j, "=", t)
print("end")
```

2) # program to print factorial of 1 to n no.

```
n = int(input("enter no. : "))
for i in range(1, n+1):
    fac = 1, j = 1
    while j <= n:
        fac = fac * j
    print(i, "!", "=", fac)
```

3) # program to print sum of first n. no. individual

```
n = int(input("enter no. : "))
# for i in range instead use while.
i = 1
while i <= n:  ; sum = 0
    for j in range(1, i+1):
        sum = sum + j
    print("Σ", i, "=", sum)
print("end")
```

Terminalogies used with loops:

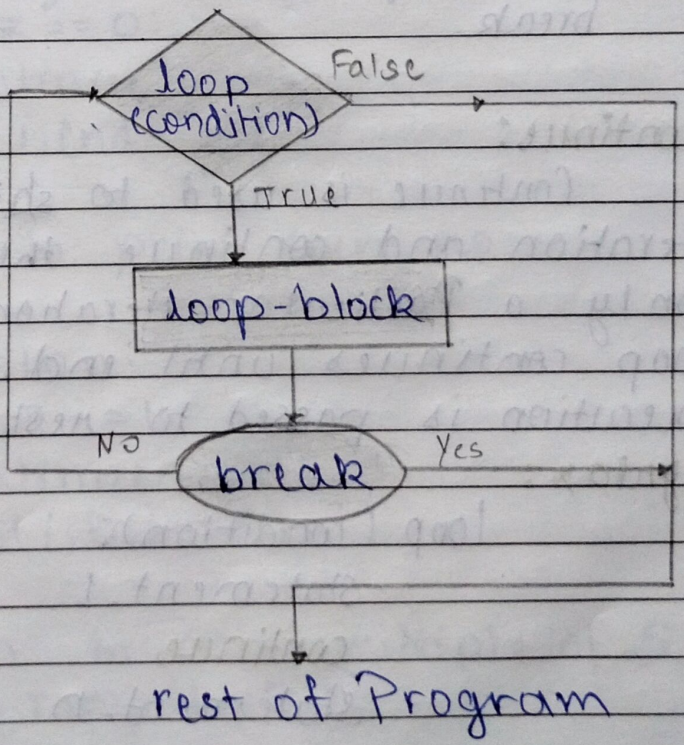
1) Break:

Break is a keyword in Python, which is used to break the current iteration and exist the loop and execute the remaining program outside the loop.

Syntax:

```
loop:
    statement-1
    ---
    Statement-n
    break
rest of the program
```

Flowchart:



Examples:

Q1. # Program to demonstrate break.

```
for i in range(1,10):
    print(i)
    break
```

→ 1

2) # Program to demonstrate break.

```
i = 5
while i<=20:
    print(i)
    if i == 20:
        break
    i = i+1
```

3) # Program to demonstrate break

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

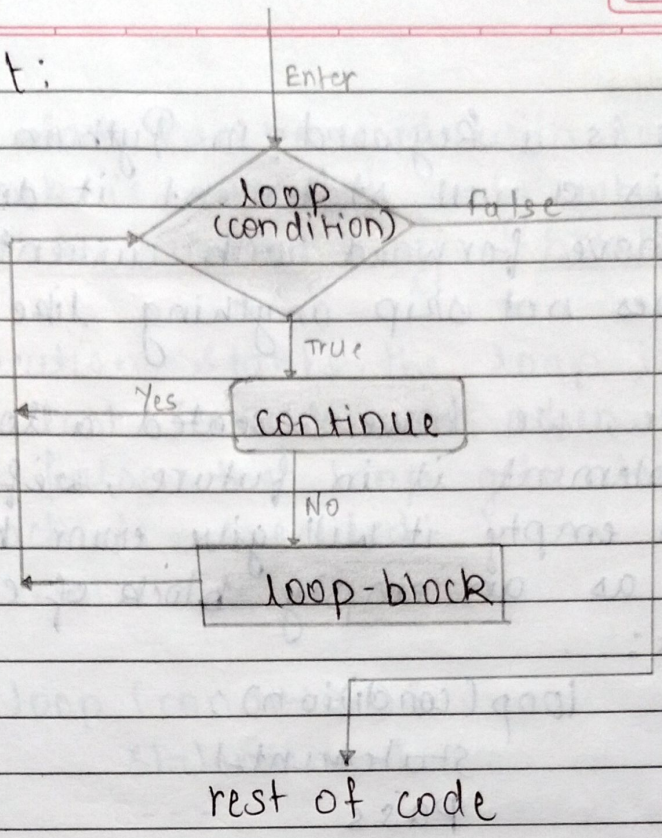
2) Continue:

Continue is used to skip a Particular iteration and continue the loop. It will skip only a Particular iteration only, and then loop continues until end, and then execution is passed to rest of the program.

Syntax:

```
loop (condition):
    statement-1
    continue
    statement-n

rest of the Program.
```

Flowchart:



Example:

1) # Program to print odd no.

```
for i in range(1, 101):
    if i%2 == 0:
        continue
    print(i, end = " ")
```

2) # Program to print even no.

```
i = 1
while i <= 100:
    if i%2 == 1
        continue.
    print(i, end = ' ')
```

3) # Programs to print multiple of 5.

```
for i in range(1, 101):
    if i%5 == 0
        print i
    else: continue.
```

3) Pass:

Pass is a keyword in Python.
Pass is a null statement it does nothing, just move forward with current iteration.
It does not skip anything like break & continue.

Suppose, we have created a loop but we want to implement it in future, if body of loop is kept empty it will give error hence Pass is used as an empty block of code.

Syntax:

```
loop(condition):
    statement-1
    pass
    statement-2
rest of code.
```

Example:

1)
```
# program to print 1 to 10 digits
for i in range(1,11):
    print(i, end=" ")
    pass
    print("Sihn")
```

2)
```
# program to find factorial.
fac=1; n=int(input())
for i in range(1, n+1):
    fac=fac*i
    Pass
print(fac)
```

4) Else:

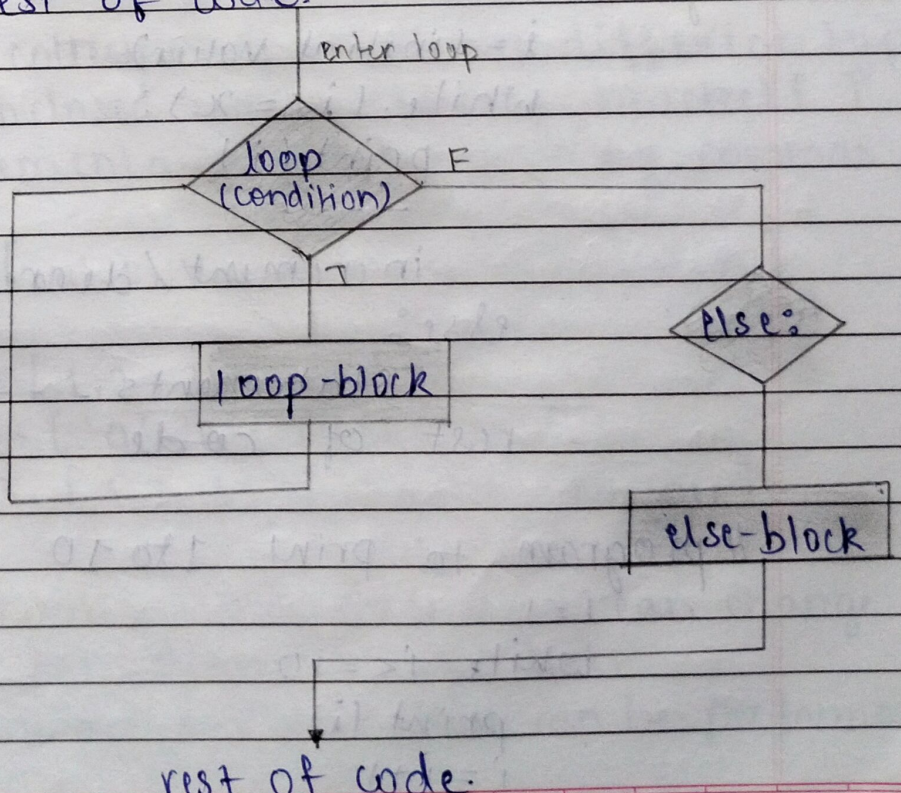
Else block can be used with looping statements. Else block is executed only when the loop is not terminated by break statement.

When execution starts, the loop is get executed on the last iteration or when the condition become false the loop gets terminated and else-block is executed.

Syntax:

```
loop (condition):
    Statement-1
    -----
    Statement-n
else:
    statements      // else-block
    --
rest of code.
```

Flow chart:



Examples

1) // else with for-loop:

Syntax:

```
for var in range():
    body of loop
    ----
else:
    statement
rest of code
```

// print 10 digits:

```
for i in range(1, 11):
    print(i)
else:
    print("Yeah! done")
```

2) // else with while loop.

Syntax:

```
i = (initial value)
while (i<=x):
    print(i)
    ----
    increment/decrement
else:
    statements ---
rest of code
```

// program to print 1 to 10 digit

```
i=1
while i<=10
    print(i)
    i=i+1
else: print("done") → rest of code
```