Decision statement

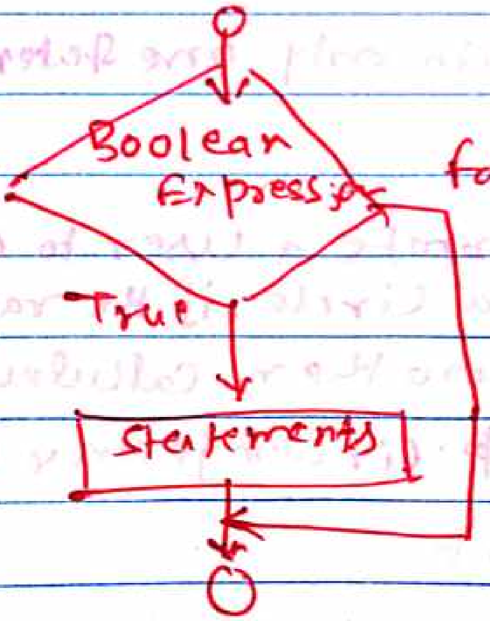
Python supports various decision making statements. [programmer decides which statement is to be executed based on a condition]

1. if statement
2. if-else statement
3. Nested if statement
4. multi-way if-elif-else statement

① If statement :→

The if statement executes a statement if a condition is true

```
if condition :              if condition:
    statement(s)                Block
```



flow chart of if statement

Note: • The statement(s) must be indented ~~ooh~~
• if more than one statement after if each statement must be indented

(P) WPP that prompts a user to enter two integer values, print the message 'Equals' if both the entered values are equal.

```
num1 = eval (input ("Enter first no:"))
num2 = eval (input ("Enter Second no:"))
if num1-num2 ==0:
        print("Both the no entered are equal")
```

Note:→ no = eval (input ("Enter the no:"))

```
[ if no > 0:              ↑ if no > 0:
      no = no * no        |  no = no * no
                           wrong I error
or
[ if no > 0: no = no * no
```

∴ if contain only one statement

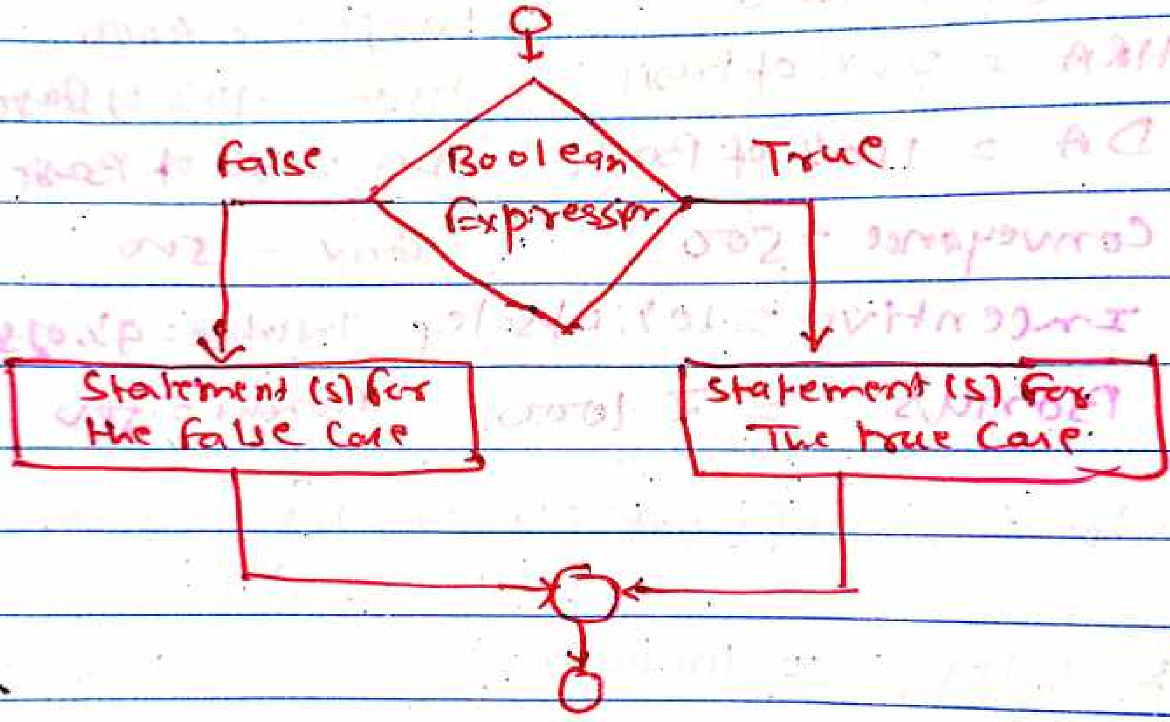
(P) WPP which promts a user to enter the radius of a circle, if the radius is greater than zero then calculate the area and ~~ϕ~~ circumference of the circle

```
from math import pi
Radius = eval (input ("Enter Radius of
                       Circle:"))
if Radius > 0:
        Area = Radius * Radius * pi
        print ("Area of circle is =", format
        (Area, ".2f"))
        print ("Circumference", 2*pi*Radius)
```

② The if-else statement :-

The if-else statement takes care of a true as well as false condition.

```
if Condition :                 |  if condition :
        statement (s)          |       if-Block
else :                         |  else :
        statement (s)          |       else_Block
```



False
Boolean Expression
True
Statement (s) for the false Case
statement (s) for The true Case

Flow chart for if-else statement

Ⓟ WAP to prompt a user to enter two numbers. Find the greater number.

```
num1 = eval(input("Enter the first no:"))
num2 = eval(input("Enter the second no:"))

if num1 > num2 :
        print(num1, " is greater than", num2)
else :
        print(num2, " is greater than", num1)
```

HW

(P) WPP to calculate the salary of a medical representative considering the sales bonus incentives offered to him are based on the total sales. if the sales exceed or Equal to ₹ 100000 follow the particulars of column 1, else follow column 2.

Column1	Column2
Basic = ₹ 4000	Basic = ₹ 4000
HRA = 20% of Basic	HRA = 10% of Basic
DA = 110% of Basic	DA = 110% of Basic
Conveyance = 500	Conv = 500
Incentive = 10% of sales	Incentive = 4% of sales
Bonus = ₹ 1000	Bonus = 500

```
sales = float(input('Enter total sales of the month'))
if sales >= 100000:
        basic = 4000
        hra = 20 * basic/100
        da = 110 * basic/100
        incentive = sales * 10/100
        bonous = 1000
        conveyance = 500
else :
        :
print("salary=", basic+hra+da+inc+bo
```

Ⓟ WPP to test whether a no is divisible by 5 and 10 or by 5 or 10

```
num= eval(input("          "))
if (num%5==0 and num%10==0):
        print(num, 'is divisible by both 5 and 10')
if (num%5==0  or num%10==):
        print(num, 'is divisible by 5 or 10')
else:
        print(num, 'is not divisible Either by 5 or 10')
```

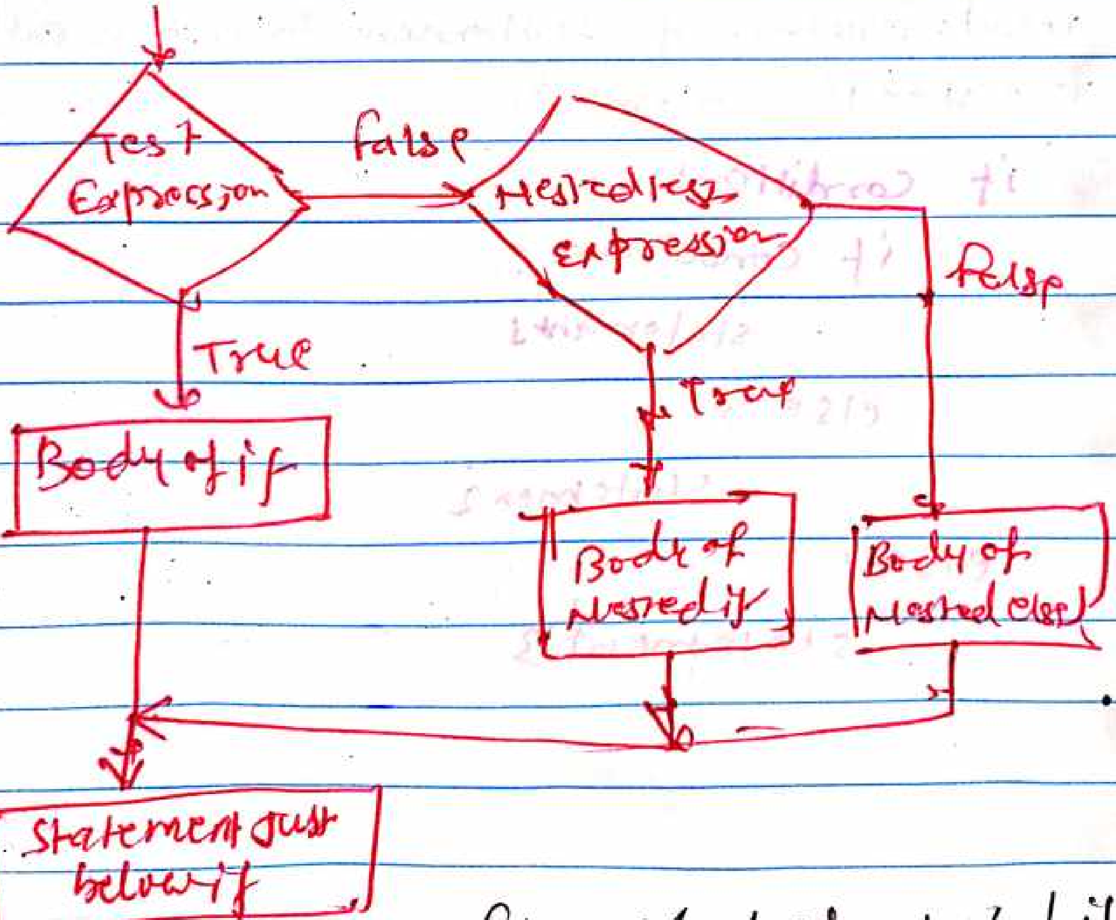
Nested if statement:-

When a programmer writes one if statement inside another if statement then it is called a nested if statement.

```
if condition1:
    if condition2:
            statement1
    else:
            Statement2
else:
        statement3
```

(P) WPP to read three nos from a user and check if the first no is greater or less than the other two nos

```
num1
num2
num3
if num1 > num2 :
        if num1 > num3 :
                print(num1, " is greater than", num2,
                        'and', num3)
  else: .
        print (num1, " is less than", num2,
                "and", num3)
  print ("End of Nested if")
```



Flow chart of Nested if

Multiway if-elif-else statements

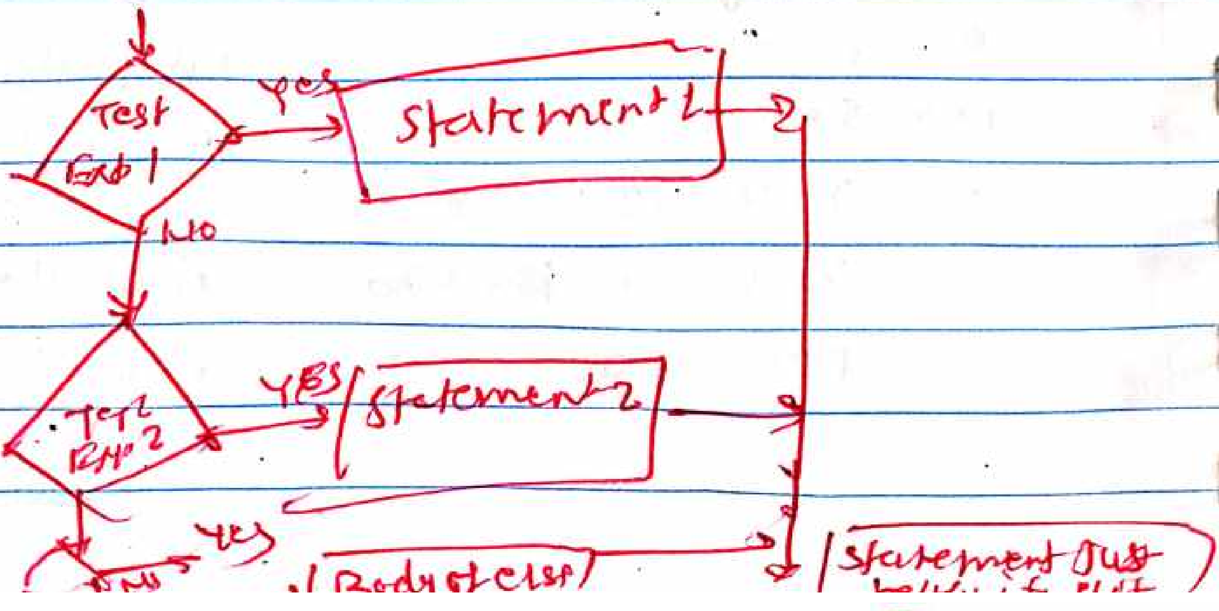
```
if condition1:
      statemen
      elif condition2:
            statement 2
      elif condition3:
            statement3
      :
      :
    elif condition n:
          statement n
    else:
          statement (s)
```

(P) WPP to prompt a user to read the marks of five different subjects. Calculate the total marks and percentage of the marks and display the message according to the range of percentage given in table

Percentage	Message
per >= 90	Distinction
per >80 && per <90	First class
per >= 70 && per <80	Second class
per >= 60 && per <70	first class
per < 60	fail

by if-else-if

```
sub1
sub2
sub3
sub4
sub5
sum = sub1+sub2+sub3+sub4+sub5
per = sum/5
if per>=90:
        print("Distinction")
else:
        if per >= 80:
                print("first class")
        else:
                if per >=70:
                        print("second class")
                else:
                        if per>=60:
                                print("pass")
                        else:
                                print("Fail")
```



by if-elif-else

```
if per>=90:
    print("Distinction")
elif per>=80:
    print("first class")
elif per>=70:
    print("second class")
elif per>=60:
    print("pass")
else:
    print("fail")
```

(P) WPP to prompt a user to enter a day of the week. if the entered day of the week is between 1 and 7 then display the respective name of the day.

```
Day = eval(input("Enter the day of week"))
if Day ==1:
    print("Its monday")
elif Day ==2:
    print(" Its tuesday")
 ,
 ,
 ,
 ;
else:
    print("Sorry!!! week contains only
                  7 days")
```

(P) WPP that prompts a user to enter two different numbers. Perform basic arithmetic operations based on the choices.

```
num1
num2
Choice = int (input ("please Enter the choice:"))
if choice ==1:
        print ("Addition of", num1, "and", num2,
               "is:", num1+num2)
 elif choice ==2:
        print ("substraction of",                    )
  elif choice ==3:
        print ("multi
   elif choice == 4:
        print ("Division of
   else:
        print ("Sorry!!! invalid choice")
```

H.W

Finding the number of Days in month

(1) Prompt the month from the user
(2) Check if the entered month is 2. febuary if so then go to step 3, else go to step 4
(3) if the entered month is 2 then check if the year is a leap year. If it is a leap year then store num days = 29, else num days = 28

(4) If the entered month is one of the following from the list (1,3,5,7,8,12) then store num days = 31. or if the entered month from the list (4,6,9,11) then store num_days = 30. If the entered month is different from the range (1 to 12) then display message "Invalid month"

(5) if the input is valid then display the message as "there are N number of days in the month M"

```
month = int (input ("months"))
if month ==2:
    year = int (input ("Enter year"))
    if (year%4=0) and (not (year%100==)
                    or (year%400==0):
            print ("num_days = 29")
    else:
            print ("num_days = 28")
elif month in (1,3,5,7,8,10,12):
        print ("num_days = 31")
elif month in (4,6,9,11):
        print ("num_days = 30")
else:
      print ("Please enter valid month")
```

Loop control statement

The While Loop :→

The while loop is a loop control statement in python and frequently used in programming for repeated Execution of statement(s) in a loop.

→ It Executes a sequence of Statements repeatedly as long as a Condition remains true.

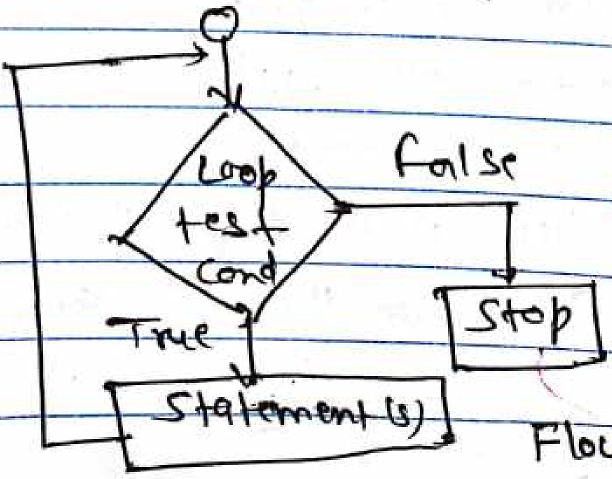
```
# While test_condition:
        # Loop Body
        Statement (s)
```

Detail: Reserved keyword while begins with the while statement

→ Test condition is a Boolean expression.

→ The Colon (:) must follow the test Condition

→ The Statement (s) within the while loop will be Executed till the condition is true



Flowchart of while loop

(P-1) WPP to print the no's from one to five using while loop :-

```
# Count = 0
while count <=5 :
        print (" Count =", Count)
        Count = count +1
```

(P-2) WPP to add 10 consecutive no's starting from 1 using the while loop

```
count = 0
sum = 0
while count <= 10 :
        sum = sum + count
        Count = Count + 1
print (" sum of first 10 no's =", sum )
```

(P-3) WPP to find the sum of the digits of a given no

```
num = int ( input (" Enter the no"))
sum = 0
rem = 0
while num > 0 :
        rem = num % 10
        num = num //10
        sum = sum + rem
print (" sum of the digits of an entered
        no"-, num, " is =", sum )
```

(P-1) WPP to print the no's from one to five using while loop :-

```
# Count = 0
while count <= 5 :
        print (" Count =" ; Count )
        Count = count +1
```

(P-2) WPP to add 10 consecutive no's starting from 1 using the while loop

```
count = 0
sum = 0
while count <= 10 :
        sum = sum + count
        Count = Count + 1
print ( " sum of first 10 no's = ", sum )
```

(P-3) WPP to find the sum of the digits of a given no

```
num = int ( input (" Enter the no" ))
sum=0
rem =0
while num >0 :
        rem = num % 10
        num = num //10
        sum = sum + rem
print ( " sum of the digits of an entered
        no "- , num, " is =" , sum )
```

(P-) WPP to display the reverse of the number

```
num= int(input(" Enter the no"))
rev=0
While num >0 :
        rem= num% 10
        num =num//10
        ~~Sum~~ = ~~Sum~~ 10+ rem
print (" Reverse of a entered no", num, " is:",rev)
```

(P) WPP to print the sum of the no's from 1 to 20 that are divisible by 5 using the while loop.

```
Count =1
Sum = 0
While Count <=20 :
        if Count % 5==0 :
                Sum = Sum+Count
        Count = count+1
print (" The Sum of Numbers from 1 to 20
        divisible by 5 is:", Sum )
```

(P) WPP to print factorial of a given no

```
Num = int (input ("Enter the no:"))
fact = 1
ans =1
while fact <=num:
        ans = ans *fact
        fact = fact+1
```

* An Armstrong no is a no which is equal to sum of the cube of its digit.

(P) WPP to check whether the no entered is an Armstrong no or not.

```
num = int(input("Please Enter the no: "))
sum=0
x=num
while num > 0 :
        d = num%10
        num = num//10
        sum= sum+ (d*d*d)
if (~~num~~ x == sum) :
        print ("The number", x, "is Armstrong no")
else :
        print ("The number", x, "is not Armstrong no")
```

(*) The inbuilt range() Function :→

- it generate a list of integers.
- range (begin, end, step)

range(5)	→ [0,1,2,3,4]
range(1,5)	→ [1,2,3,4]
range(1,10,2)	→ [1,3,5,7,9]
range (5,0,-1)	→ [5,4,3,2,1]
range(5,0,-2)	→ [5,3,1]
range(-4,4,2)	→ [-4, [illegible], -2, [illegible] 2, [illegible]]
range (0,1)	→ [0]
range(1,1)	→ Empty
range (0)	→ Empty

The For Loop:→

→ The for loop is a python statement which repeats a group of statements for a specified no of times.

```
for var in range(m,n):
        print(var)
```

(P-) use for loop to print no's from 1 to 5.

```
for i in range(1,6):
        print(i)
```

(P-) Display Capital letters from A to Z

```
for i in range (65,91,1):
        print (chr(i),end=" ")
```

(P) to Print no's from 1 to 10 in the reverse order

```
for i in range (10,0,-1):
        Print (i, end=' ')
```

(P-) wpp to print squares of the first five no's

```
for i in range (1,6):
        square = i*i
        print ("square of ",i," is:",square)
```

(P-) WPP to print even no's from 0 to 10 and find their sum.

```
sum=0
print ("Even nos from 0 to 10 are as follows")
for in inrange (0,11,1):
        if i%2==0:
            print (i)
            sum=sum+i
print ("sum of even nos from 0 to 10 is=",sum)
```

(P-) WPP to calculate the sum of numbers from 1 to 20 which are not divisible 2,3 or 5.

```
sum =0
for i in range (1,20):
        if (i %2==0 or i%3==0 or i%5 ==0):
            print (" ")
        else:
            print (i)
            sum= sum+i
    print (" sum of Even no's from 1 to 10 is =",sum)
```

(P-) WPP that prompts a user to enter four numbers and find the greatest no among the four no's Entered.

```
Num1 = int (input ("Enter the First no:"))
Num2 =
Num3 =
Num4 =
Sum = Num1+Num2+Num3+Num4
 for i in range (Sum):
        if i==num1 or i== num2 or i==num3
           or i==num4 :
            large(i)
  Print (large)
```

summation of 1 to given no

(P-) WPP to generate a triangular number.

```
num = int (input (" Enter the no:"))
T_Num = 0          (1, Num+1):
for i in range (Num, 0, -1):
        T_Num = T_Num+i
Print (T_Num)
```

Q- WPP to print Fibonacci series up to 8.

```
n1=0
n2=1
n= int(input("n:"))
print(n1)
for i in range(n+1):
        n3 = n1+n2
        n1=n2
        n2=n3
        print(n3)
```

Nested Loops:→

→ The for and while loop statements can be nested in the same manner in which the if statements are nested.

→ Loops within the loops or when one loop is inserted completely within another loop, then it is called Nested loop.

(P-) Write PP to display multiplication tables from 1 to 5.

```
for i in range (1,11,1):
        for j in range (1, 6,1):
                print ( i*j , end=" ")
        print( )
```

(P-)

```
. . . . .
. . . .
. . .
. .
.
```

```
for i in range (1,6):
        for j in range (6,i,-1):
                print ("*", end=" ")
        print ( )
```

```
.
. .
. . .
. . . .
. . . . .
```

```
for i in range (1, 6):
        for j in range (0,i):
                print("*", end=" ")
        print( )
```

(P-)

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

```
num=1
for i in range (1,6):
    num=num+1
    for j in range (1, num):
        print (j, end=" ")
    print( )
```

and
(+1)

(P-)

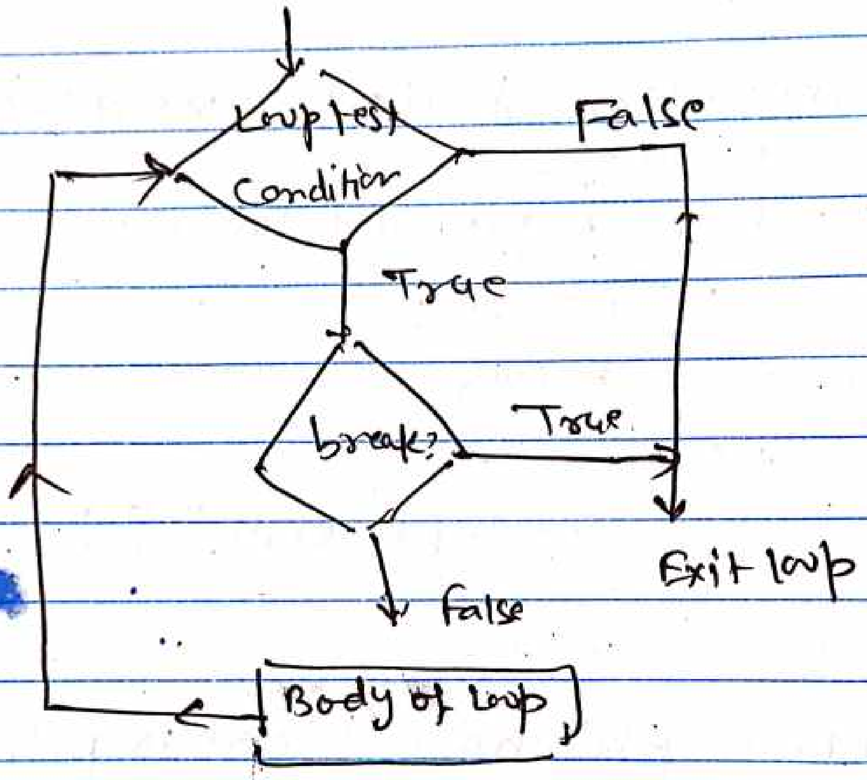
```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

```
num=6
for i in range (1,6):
    num=num-1
    for j in range (1,num):
        print (j ,end=" ")
    print( )
```

The Break Statement:→

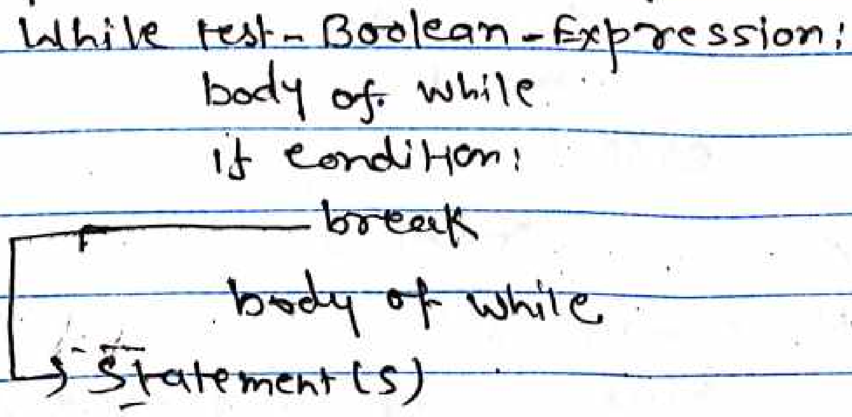
→ The keyword break allows a programmer to terminate a loop.

→ When the break statement is encountered inside a loop, the loop is immediately terminated and the program control automatically goes to the first statement.



Flow Chart of break statement

Working of break in while loop:→



```
While test-Boolean-Expression:
        body of while
        if condition:
            break
        body of while
Statement(s)
```

Working of break in for Loop :->

```
for var in sequence :
        body of for
        if condition :
    ┌──────── break
    │   body for for
    └──> statement(s).
```

(P-) wpp to demonstrate the use of the break statement.

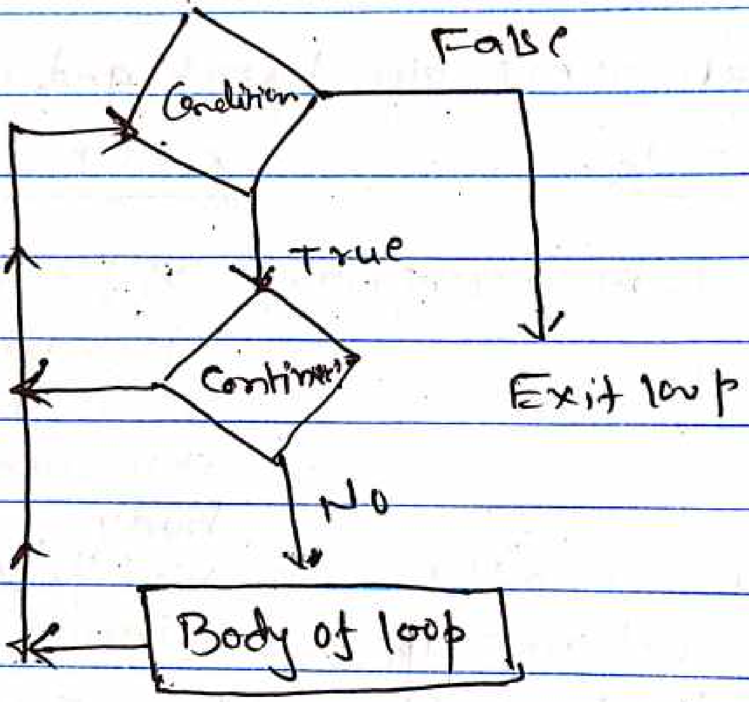
```
for i in range (1,100,1):
      if i==11 :
            break
      else :
            print(i, end=" ")
```

(P-) check if the no entered is prime or not

```
no = int (input ("Enter the no"))
for i in range (2,n):
      if no%i==0 :
            count = 0
            break
      else :
            count=1
if count==1 :
      print (no, " is prime no'')
else,
      prin( no, " is not prime
```

The continue statement :->

When continue is encountered within a loop, the remaining statements within the body are skipped but the loop condition is checked to see if the loop should continue or Exit.



Flowchart of Continue statement

```
While test-boolean-Expression:
        body of While
        if condition:
            Continue
        body of While
            Statement (s)
```

```
For var in Sequence:
        body of for
        if condition:
              continue
        body of for
              statement(s)
```

__Note:__ Difference b/w break and continue

__Break__	__continue__
Exits from current block or loop	skips the current iteration and also skips the remaining statements within the body
control passes to the next statement	control passes at the beginning of the loop.
Terminates the loop	Never terminates the loop.

(P-) Read the string "Hello world" from the user. Make sure use of continue keyword and Remove space.

```
S= "Hello world"
for i in ~~nn~~ S:
      if i == " ":
            continue
      print(i, end="")
```

Python Pass statement :-

- The Python Pass statement is a null statement.
- But the difference b/w Pass and Comment is that comment is ignored by the Interpreter where as Pass is not ignored.

→ **What is Pass statement :-**

- When the user does not know what Code to write, so user simply places a pass at that line. Sometimes, the pass is used when the user does not want any code to Execute. ~~So Users Can simply place a pass where Empty~~ code is not allowed, like in loops, function definitions, class definitions, or in if statements. So using a pass statement user avoids this error.

Why Python needs "pass"

If we do not use Pass or simply enter a comment or a blank here, we will Receive an Indentation Error message.

```
n = 26
if n > 26 :
        # Write code .
print ( " xyz" )          # Indentation Error

  def function ( ) :
      pass
```

(P-)
```
n=10
for i in range(n):
    #
    #
    pass
```

(P-)
```
a=10
b=20
if a<b :
    pass
else :
    print ("b<a")
```

(P)
```
li = ['a','b','c','d']
for i in li :
    if (i=='a') :
        pass
    else :
        print(i)
```

Functions

With the help of functions, an entire Program can be divided into small independent modules (each small module is called a function). This improves the code's readability as well as the flow of execution as small modules can be managed easily.

Syntax:- def funcName (parameters): ← called formal parameters.

statement(s) {body}

→ if a function contain more then one parameter the all the parameters are separated by commas.

(P-) WPP to create function having a name Display. print the message "welcome to Python programming" inside the function.

```
def Display():
        Print("welcome to Python Prg")
Display()
```

(P-) WPP to prompt the name of a user and print the welcome message "Dear name_of_user welcome to Python Programming!!!"

```
def msg():
        str1=input("En Name")
        Print("Dear", str1, "welcome to python prog")
msg()
```

(P-) WPP to add the sum of digits 1 to 25, 50 to 76. and 90 to 101 using function.

```
def sum (x, y):
      s=0
      for i in range (x, y+1):
            s=s+i
      print (s)
   sum (1, 25)
   sum (50, 76)
   sum (90, 100)
```

Parameters and Arguments in a function:→
while parameters are defined by names that appear in the functions definition, arguments are values actually passed to a function when it calling it. Thus parameters defined what type of arguments a function can accept.
Exp.-

(P-) WPP to find the maximum of two nos.

```
def printmax (num1, num2):
      if num1 > num2 :
            print (num1)
      elif num2 > num1 :
            print(num2)
      else:
            print ('Equal')
printmax (20, 10)
```

(P-) WPP to Print Factorial

```
def fact1(num):
    fact=1
    for i in range (1, num+1):
        fact = fact * i
    print (fact)
```

# fact_1 (10)

OR
```
number = int (input ("Enter the no:"))
fact_1 (number)
```

Position Arguments :→

→ Parameters are assigned by default according to their position.

→ The first argument in the call statement is assigned to the first parameter listed in the function definition.

→ Similarly the second argument in the call statement is assigned to the second parameter listed in the function's definition an so on.

exp:
```
def Display (Name, age):
        Print ("Name =", Name, "age =", age)
Display ( "John", 25)
Display ( 25, "John")
Display ( 25 )          # Error
                        missing one req pos, argu
```

Keyword Arguments:→

- A programmer can pass a Keyword argument to a function by using it's corresponding parameter name rather than it's position.

```
def Display (Name, age):
        print ("Name=", name, "age=", age)
Display (age=25, Name = "John")
```

Precautions for using Keyword arguments.

Exp
```
def Display (num1, num2):
   Display (40, num2=10) ✓
   Display (num2=10, 40) ✗
```

① * Positional argument Cannot follow a Keyword argument.

② * A programmer Cannot duplicate an argument by specifying it as both, a positional argument and a keyword argument.

Exp:
```
      def Display (num1, num2):
      Display (40, num1 = 40)
```

Parameters with Default values :-

```
def greet (name, msg="welcome to Python!"):
    print ("Hellow", name, msg)
greet ("sachin")
# greet ("Bill Gates", "How are you")
```

→ New argument value overwrites the default parameter value.

```
# def greet (msg="welcome to Python", name)
                                # Error
```

→ once we have default value to a parameter, all the parameters to it's right must also have default values.

(P-) WPP to calculate the area of a circle using the formula:

A = pi * r * r

```
def area_circle( Pi=3.14, radius=1):
    area = Pi * radius * radius
    print (area)
area_circle ( )
area_circle (5)
```

O/P ?

```
def dis_values (a, b=10, c=20) ;
        print (~~"a, b, c~~ "a=", a , "b=", b, "c=", c)
dis_values ( 15)
dis_values (50, b=30 )
dis_values ( c=80, a=25, b=35)
```

O/P :

```
a = 15 , b=10 , c=20
a = 50   b=30   c=20
a = 25   b=35   c=80
```

The return Statement

The return statement is used to return a value from the function.

→ It is also used to return from a function. break out the function..

(P-) WPP to return the minimum of two no's

```
def minimum (a,b) :
        if a < b :
                return a
        elif b<a :
                return b
        else :
                return "Both the numbers are
                                Equal"
print (minimum (100, 85 ))
```

(P-) Distance $= \sqrt{(x_2-x_1)^2 + (y_2-y_1)^2}$

(import math)

```
def calc_Distance (x1 y1, x2, y2):
        Distance = ~~math~~.( (x2-x1)**2 + (y2-y1)**2)
                                           ** 0.5

        return Distance,
print (calc_Distance (4A,2,2))
```

(P-) For a quadratic Eq^n in the form of ax^2+bx+c, the discriminant D, is b^2-4ac write a function to compute the discriminant D, the returns the following output depending on the discriminant D.

if D>0 → The eq^n has two real roots
if D<0 → the eq^n has two comp root
if D=0 → the eq^n has two real and equal

```
def quad_D (a,b,c):
     D = b*b-4*a*c
     if D > 0
          return " The eq^n has two real roots"
     elif d <0:
          return " the eq^n has two Complex"
     else:
          return "The eq^n has one real root"
print( quad_D(1,2,8))
```

Returning multiple values :→

(P-) WPP to use a function Calc_arith_op(num1, num2) to calculate and return at once the result of arithmetic operations such as addition and substraction.

```
def cal_arith_op(num1,num2):
        return num1+num2, num1-num2
print ( cal_arith_op( 10,20))
```

Assign Returned multiple values to variable (s)

(P-) WPP to return multiple values from a function.

```
def compute(num1):
        ~~print~~return num1*num1, num1*num1*num1
square, cube = compute(~~num1~~ 4)
print (square, cube)
```

Recursive function :->

there might be a situation where a function needs to invoke itself.

→ Recursive means that a function is repetitively called by itself.

→ function is said to be recursive if a statement within the body of the function calls itself.

P- Calculate the factorial of a number

```
def factorial (n):
        if n=0 :
            return 1
        return n * factorial (n-1).
    print (factorial (5))
```

The Lambda function [anonymous function]

→ such kind of functions are not bound to a name. They only have a code to Execute that which is associated with them.

Name = lambda(variables) : Code

```
def func(x):
    return x*x*x
print(func(3))
```

→ Define lambda func.

```
cube = lambda x : x*x*x
print(cube(2))
```

↳ Call lambda function.

HW WPP to calculate compound interest for principle amount as ₹ 10,000 at rate of interest as 5% and number of years the amount is deposited as 7 Years.