Python Control Statements

In any programming language a program may execute sequentially, selectively or iteratively. Every programming language provides constructs to support Sequence, Selection and Iteration. In Python all these construct can broadly categorized in 2 categories.

A. **Conditional Control Construct**
   (Selection, Iteration)
B. **Un- Conditional Control Construct**
   (pass, break, continue, exit(), quit())

Python have following types of control statements

1. **Selection** (branching) Statement
2. **Iteration** (looping) Statement
3. **Jumping** (break / continue)Statement

Conditional Control Statements

Un Conditional Control Statements

Python Selection Statements

Python have following types of selection statements

1. if statement
2. if else statement
3. Ladder if else statement (if-elif-else)
4. Nested if statement

Python If statements

This construct of python program consist of one if condition with one block of statements. When condition becomes true then executes the block given below it.
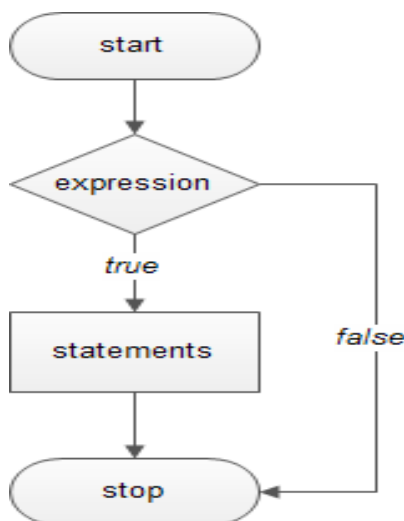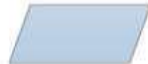
Syntax:

if (condition):

....

....

....

> **Flow Chart:** it is a graphical representation of steps an algorithm to solve a problem.

Flowchart

Symbol	Name	Function
	Start/end	An oval represents a start or end point
→	Arrows	A line is a connector that shows relationships between the representative shapes
	Input/Output	A parallelogram represents input or output
	Process	A rectangle represents a process
	Decision	A diamond indicates a decision

 Age=int(input("Enter Age: "))

If (age>=18):

       Print("You are eligible for vote")

If(age<0):

       Print("You entered Negative Number")
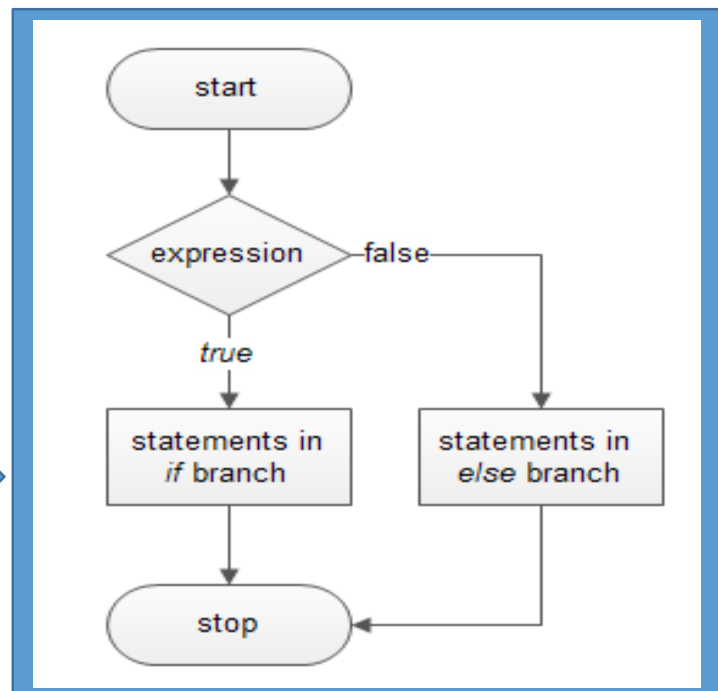
Python if - else statements

This construct of python program consist of <mark>one if condition with two blocks</mark>. When condition becomes true then executes the block given below it. If condition evaluates result as false, it will executes the block given below else.

Syntax:

if (condition):

       ………………..

else:

       ………………..

Flowchart ➡️

Example-1:

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

if ( age>=18):
        print("You are eligible for vote")
else:
        print("You are not eligible for vote")
```

Example-2:

```
N=int(input("Enter Number: "))
if(n%2==0):

        print(N," is Even Number")
Else:
        print(N," is Odd Number")
```

Python Ladder if else statements (if-elif-else)

This construct of python program consist of ==more than one if condition.== When first condition evaluates result as true then executes the block given below it. If condition evaluates result as false, it transfer the control at else part to test another condition. So, it is ==multi-decision making== construct.

Syntax:

if (condition-1):

    ………………..

    ………………..

elif (condition-2):

    ………………..

    ………………..

elif (condition-3):

    ………………..

    ………………..

else:

    ………………..
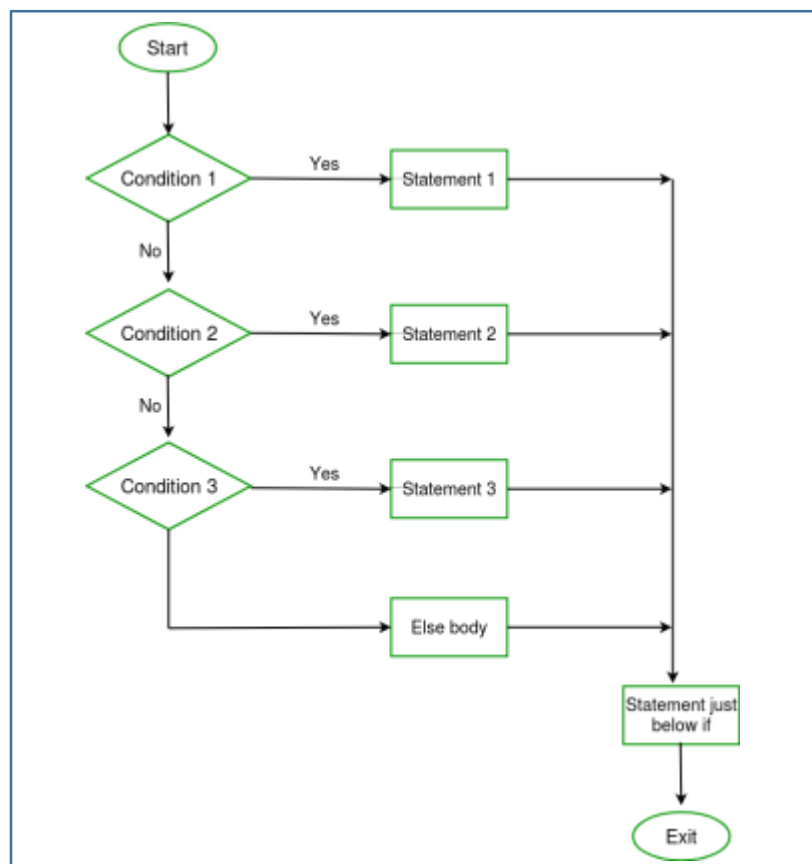
    ………………..

Example:

num=int(input("Enter Number: "))

If (num>=0):

    Print("You entered positive number")

elif (num<0):

    Print("You entered Negative number")

else:

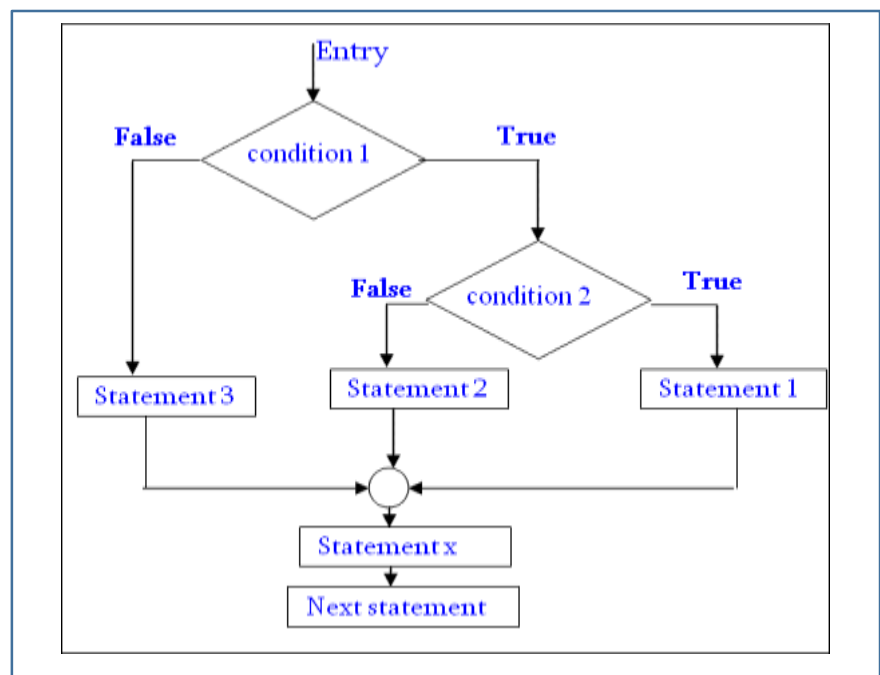   Print("You entered Zero ")

Python Nested if statements

It is the construct where one if condition take part inside of other if condition. This construct consist of more than one if condition. Block executes when condition becomes false and next condition evaluates when first condition became true.

So, it is also multi-decision making construct.

Syntax: **FlowChart**

if (condition-1):

     if (condition-2):

         ....

         ....

     else:

         ....

         ....

else:

     ....

     ....

Example:

```
num=int(input("Enter Number: "))
If ( num<=0):
        if ( num<0):
                Print("You entered Negative number")
        else:
                Print("You entered Zero ")
else:
        Print("You entered Positive number")
```

Program: find largest number out of given three numbers

```
x=int(input("Enter First Number: "))
y=int(input("Enter Second Number: "))
z=int(input("Enter Third Number: "))
if(x>y and x>z):
    largest=x
elif(y>x and y>z):
    largest=y
elif(z>x and z>y):
    largest=z
print("Larest Value in %d, %d and %d is: %d"%(x,y,z,largest))
```

Program: calculate simple interest
Formula: principle x (rate/100) x time

```
p=float(input("Enter principle amount: "))
r=float(input("Enter rate of interest: "))
t=int(input("Enter time in months: "))
si=p*r*t/100
print("Simple Interest=",si)
```

Program: calculate EMI

Input the following to arrive at your Equal Monthly Installment -EMI:

1. Loan Amount: Input the desired loan amount that you wish to avail.
2. Loan Tenure (In Years): Input the desired loan term for which you wish to avail the loan.
3. Interest Rate (% P.A.): Input interest rate.
4. $EMI=[[P*R*(1+R)^N] / [(1+R)^N{-1}]]$

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

```python
YR=float(input("Enter rate of interest P.A. : "))
T=int(input("Enter tenure(Installments) in years: "))
MR=YR/(12*100) # Monthly Rate
EMI=(P*MR*(1+MR)**T)/(((1+MR)**T)-1)
print("Principle Amount: ",P)
print("Rate of Interest(Yearly): ",YR)
print("No. of Installments: ",T)
print("EMI Amount: ",EMI)
```

Program: Sorting of three number. (Ascending and Descending)

```python
x=int(input("Enter First Number: "))
y=int(input("Enter Second Number: "))
z=int(input("Enter Third Number: "))
min=max=mid=None
if(x>=y and x>=z):
    if(y>=z):
        min,mid,max=z,y,x
    else:
        min,mid,max=y,z,x
elif(y>=x and y>=z):
    if(x>=z):
        min,mid,max=z,x,y
    else:
        min,mid,max=x,z,y
elif(z>=x and z>=y):
    if(x>=y):
        min,mid,max=y,x,z
    else:
        min,mid,max=x,y,z
print("Numbers in Ascending Order: ",min,mid,max)
print("Numbers in Descending Order: ",max,mid,min)
```

Program: Absolute Value

Absolute value of a given number is always measured as positive number. This number is the distance of given number from the 0(Zero). The input value may be integer, float or complex number in Python. The absolute value of given number may be integer or float.
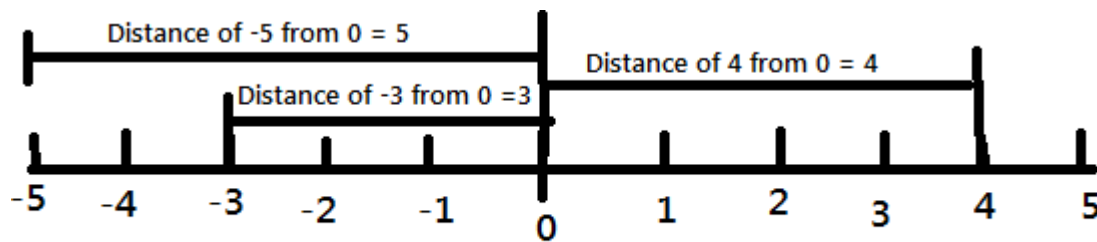
Concept of Absolute Value

(i). Absolute Value of -5 is 5   (ii) Absolute Value of -3 is 3   (iii) Absolute Value of 4 is 4

```
n=float(input("Enter a number to find absolute value: "))
print("Absolute Value using abs(): ",abs(n))
if(n-int(n)>=0 or n-int(n)<=0):   # This code is used to identify that number is float or int type.
    pass
else:
    n=int(n)
if(n<0):
    print("Absolute Value= ",n*-1)
else:
    print("Absolute Value= ",n)
```

Program: Calculate the Total selling price after levying the GST (Goods and Service Tax) as CGST and SGST on sale.

CGST (Central Govt. GST), SGST (State Govt. GST)

Sale amount	CGST Rate	SGST Rate
0-50000	5%	5%
Above 50000	18%	18%

```python
amt=float(input("Enter total Sale Amount: "))
if(amt<=50000):
    rate=5
else:
    rate=18
cgst=sgst=amt*rate/100
tot_amt=amt+cgst+sgst
print("Amount of Sale: ",amt)
print("GST rate of Sale: ",rate)
print("CGST of Sale: ",cgst)
print("SGST of Sale: ",sgst)
print("Total Payable Amount of Sale: ",tot_amt)
```

Python Iteration Statements

The iteration (Looping) constructs mean to execute the block of statements again and again depending upon the result of condition. This repetition of statements continues till condition meets True result. As soon as condition meets false result, the iteration stops.

Python supports following types of iteration statements

1. while
2. for

Four Essential parts of Looping:

i.   **Initialization of control variable**
ii.   **Condition testing with control variable**
iii. **Body of loop Construct**
iv.   **Increment / decrement in control variable**

Python while loop

The while loop is conditional construct that executes a block of statements again and again till given condition remains true. Whenever condition meets result false then loop will terminate.

Syntax:

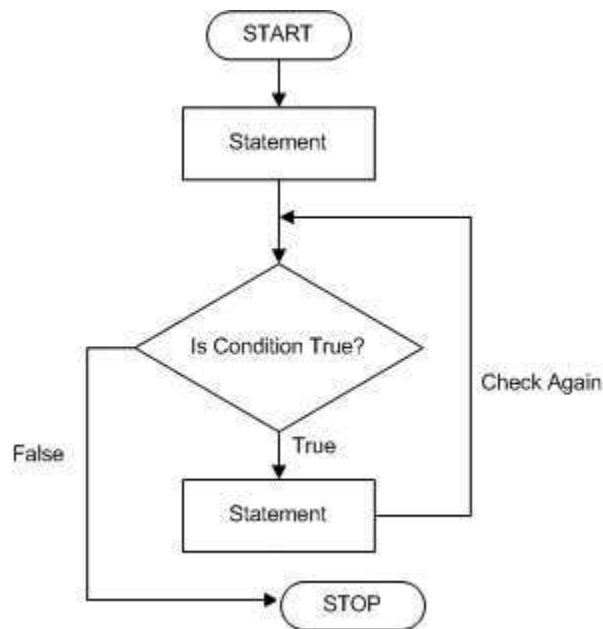Initialization of control variable

while (condition):

        ……………………..

        Updation in control variable

        ..…………………

Flowchart

Example: print 1 to 10 numbers

num=1   # initialization

while(num<=10):   # condition testing

    print(num, end=" ")

                        Body of loop

    num + = 1   # Increment

Example: Sum of 1 to 10 numbers.

num=1

sum=0

while(num<=10):

    sum + = num

    num + = 1

print("The Sum of 1- 10 numbers: ",sum)

Example: Enter per day sale amount and find average sale for a week.

Python range( ) Function

The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number. The common format of range() is as given below:
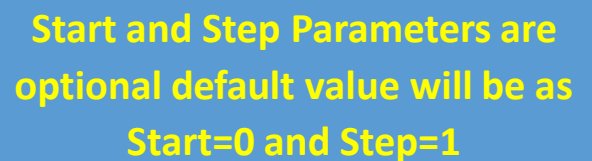
range (**start value**, **stop value**, **step value**)

Where all 3 parameters are of integer type
Start value is Lower Limit
Stop value is Upper Limit
Step value is Increment / Decrement

Start and Step Parameters are optional default value will be as Start=0 and Step=1

Note: The Lower Limit is included but Upper Limit is not included in result.

Example

range(5)	=> sequence of 0,1,2,3,4
range(2,5)	=> sequence of 2,3,4
range(1,10,2)	=> sequence of 1,3,5,7,9
range(5,0,-1)	=> sequence of 5,4,3,2,1
range(0,-5)	=> sequence of [ ] blank list (default Step is +1)
range(0,-5,-1)	=> sequence of 0, -1, -2, -3, -4
range(-5,0,1)	=> sequence of -5, -4, -3, -2, -1
range(-5,1,1)	=> sequence of -5, -4, -3, -2, -1, 0

L=list(range(1,20,2)

Print(L)   Output: [1, 3, 5, 7, 9, 11, 13, 15, 17, 19]

Python for loop

A for loop is used for iterating over a sequence (that is either a list, a tuple, a string etc.) With for loop we can execute a set of statements, and for loop can also execute once for each element in a list, tuple, set etc.

Example: print 1-10 numbers

```
for num in range(1,11,1):
        print(num, end=" ")
```

Output: 1 2 3 4 5 6 7 8 9 10

Example: print 10-1 numbers

```
for num in range(10,0,-1):
          print(num, end="  ")
```

Output: 10 9 8 7 6 5 4 3 2 1

Print each element in a fruit list:
```
fruits = ["mango", "apple", "grapes", "cherry"]
for x in fruits:
  print(x)
```

output:
mango
apple
grapes
cherry

```
for x in "TIGER":
  print(x)
```
output:

T
I
G
E
R

Membership Operators:

The "in" and "not in" are membership operators. These operators check either given value is available in sequence or not. The "in" operator returns Boolean True result if value exist in sequence otherwise returns Boolean False.

The "not in" operator also returns Boolean True / False result but it works opposite to "in" operator.

else in for Loop

The else keyword in for loop specifies a block of code to be executed when the loop is finished:

```python
for x in range(4):
  print(x, end=" ")
else:
  print("\nFinally finished!")
```

output: 0 1 2 3
        Finally finished!

Nested Loops

A nested loop is a loop inside another loop.

```python
city = ["Jaipur", "Delhi", "Mumbai"]
fruits = ["apple", "mango", "cherry"]
for x in city:
  for y in fruits:
    print(x, ":",y)
```

output:

Jaipur : apple
Jaipur : mango
Jaipur : cherry
Delhi : apple
Delhi : mango
Delhi : cherry
Mumbai : apple
Mumbai : mango
Mumbai : cherry

Un- Conditional Control Construct
(pass, break, continue, exit(), quit())

pass Statement (Empty Statement)

The pass statement do nothing, but it used to complete the syntax of programming concept. Pass is useful in the situation where user does not requires any action but syntax requires a statement. The Python compiler encounters pass statement then it do nothing but transfer the control in flow of execution.

```python
a=int(input("Enter first Number: "))
b=int(input("Enter Second Number: "))
if(b==0):
    pass
else:
    print("a/b=",a/b)


for x in [0, 1, 2]:
    pass
```

Jumping Statements

break Statement

The jump- break statement enables to skip over a part of code that used in loop even if the loop condition remains true. It terminates to that loop in which it lies. The execution continues from the statement which find out of loop terminated by break.

```
n=1
while(n<=5):
   print("n=",n)
   k=1
   while(k<=5):
     if(k==3):
        break
     print("k=",k, end=" ")
     k+=1
   n+=1
   print()
```

```
Output:
n= 1
k= 1 k= 2
n= 2
k= 1 k= 2
n= 3
k= 1 k= 2
n= 4
k= 1 k= 2
n= 5
k= 1 k= 2
```

Exit the loop when x is "banana":

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
    if x == "banana":
        break
    print(x)
```

output: apple

Continue Statement

Continue statement is also a **jump statement. With the help of continue statement, some of statements in loop, skipped over and starts the next iteration. It forcefully stop the current iteration and transfer the flow of control at the loop controlling condition.**

```
i = 0
while i <=10:
    i+=1
    if (i%2==1):
        continue
    print(I, end=" ")
```
output: 2 4 6 8 10


```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
    if x == "banana":
        continue
    print(x)
```

output:

apple
cherry