Rohini Mittal Unit-II

Python Program Flow control conditional Blocks

Flow control

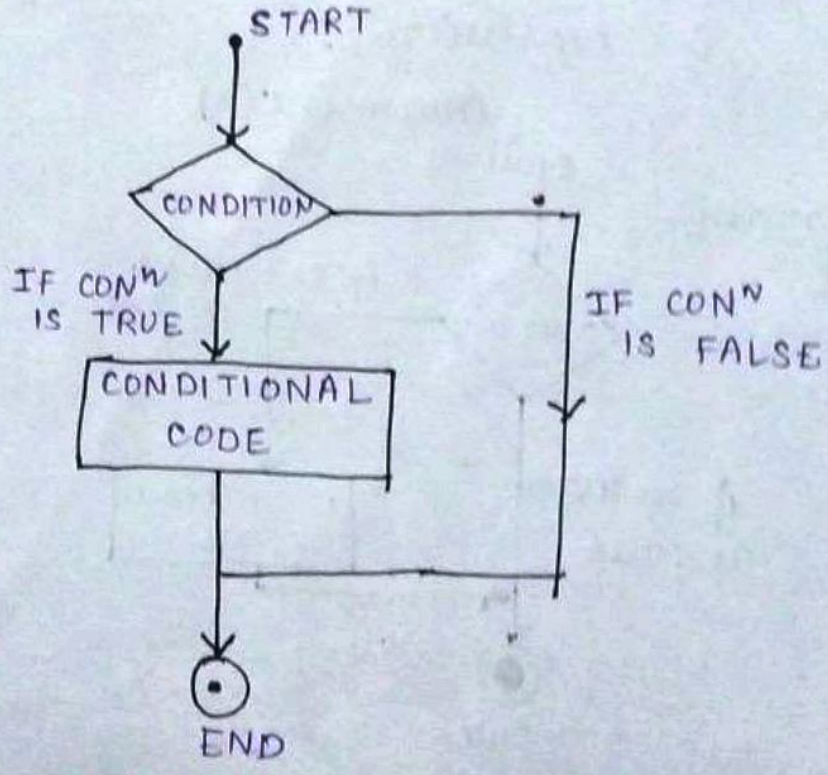
The order of execution of statement in a program is known as flow of control.

Python supports to type of control structures
- selection / Decision making (if, else, elif)
- Repetition (while, for)

Decision Making Statement

Decision making is the anticipation of conditions occuring during the execution of a program & specified actions taken according to that condition.

- Decision making statement generates boolean
  if condition is matched
  return True
  else
  return False



- Python programming lang. assumes any non-zero & non-null value as TRUE otherwise FALSE.
- Python provides following types of decision-making statements

Statement	Description
if statements	An if statement consist of a boolean expression followed by one or more statements
if ... else statement	An if statement followed by optional else statement
nested if statements	You can use one if or elif statement inside another if or else if statements.

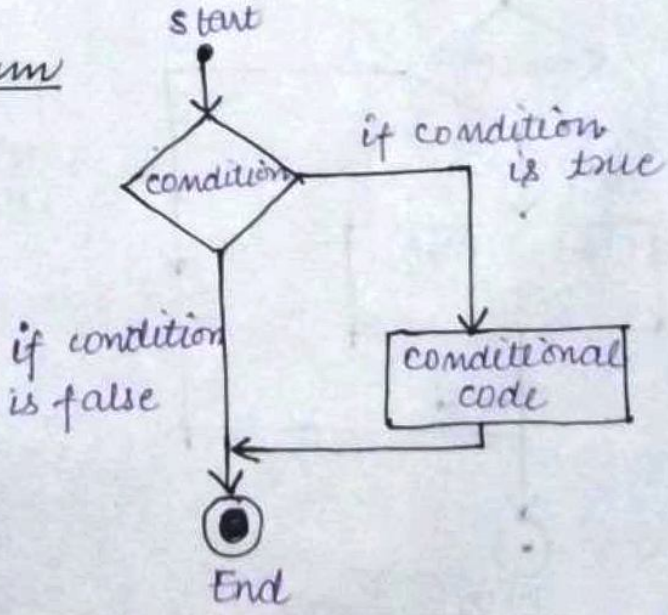
If Statements

If statement contains a boolean expression.
If the boolean expression evaluates to TRUE then block of statement inside if statement is executed

syntax

```
if expression :
    statement(s)
```

Flow Diagram



IF- ELSE statement

- An if statement is combined with an else statement
- If the conditional expression in the if statement is True

  then

  block of statement inside if is executed

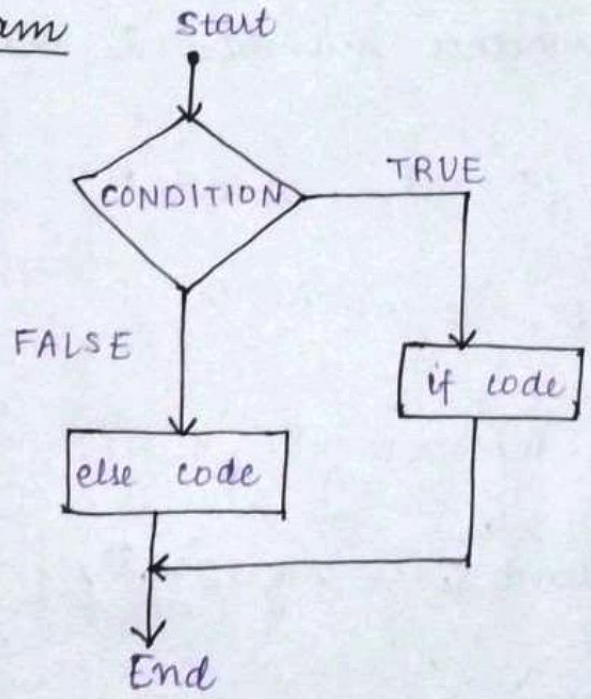
  else

  block of statement inside else is executed.

Syntax

```
if expression :
    statement(s)
else :
    statement(s)
```

Flow Diagram



ELIF Statement

elif statement allows to check multiple expression for TRUE & executes block of code as soon as one of the condition evaluates to TRUE.

Syntax

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

1. WAP to check whether number is positive, negative or 0.

```
n = int(input("Enter number : "))
if(n>0):
    print("Number is positive")
elif(n==0):
    print("Number is zero")
else:
    print("Number is negative")
```

O/P
```
Enter number: 5
Number is positive
```

2. WAP to find greatest among three numbers.

```
a = int( input (" Enter a : "))
b = int ( input (" Enter b : "))
c = int ( input (" Enter c : "))

if( a > b):
    if ( a > c):
        print ( a , " is the largest number.")
    else
        print ( c , " is the largest number. ")

else :
    if ( b > c ):
        print ( b , " is the largest number.")
    else :
        print ( c , " is the largest number .")
```

3. WAP to check whether year is leap year or not.

```
year = int( input (" Enter any year to be checked : "))
if ( year % 4 == 0 and year % 100 != 0 or
                       year % 400 == 0) :
    print( " %d is a leap year " % year)

else :
    print( " %d is not a leap year " % year)
```

o/p Enter any year to be checked : 2000
2000 is a leap year .

4. WAP to take 5 subject marks in input & display the grade

if avg of mark >= 90 Grade A

avg Marks >= 80 and avg Marks < 90 Grade B

avg Marks >= 70 and avg Marks < 80 Grade C

avg Marks >= 60 and avg Marks < 70 Grade D

avg Marks < 60 Grade E

```
s1 = int(input("Enter marks of sub 1 : "))
s2 = int(input("Enter marks of sub 2 : "))
s3 = int(input("Enter marks of sub 3 : "))
s4 = int(input("Enter marks of sub 4 : "))
s5 = int(input("Enter marks of sub 5 : "))
avg = (s1+s2+s3+s4+s5)/5
if(avg>=90):
    print("Grade A")
elif(avg>=80 and avg<90):
    print("Grade B")
elif(avg>=70 and avg<80):
    print("Grade C")
elif(avg>=60 and avg<70):
    print("Grade D")
else:
    print("Grade E")
```

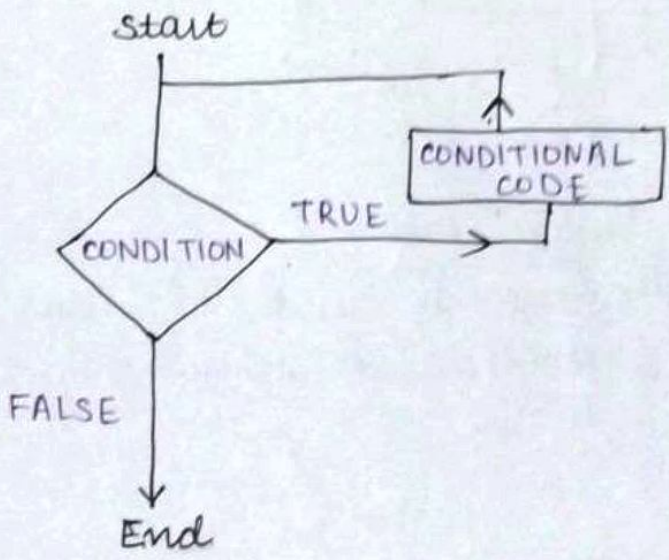
op
```
Enter marks of sub1 : 50
Enter marks of sub 2 : 60
Enter marks of sub3 : 50
Enter marks of sub4 : 60
Enter marks of sub 5 : 50
Grade E
```

Different Types of loops in Python

In general, statements are executed sequentially. The first statement is executed first followed by the second and so on.

There may be a situation when you need to execute block of code several numbers of time.

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



(i) while loop

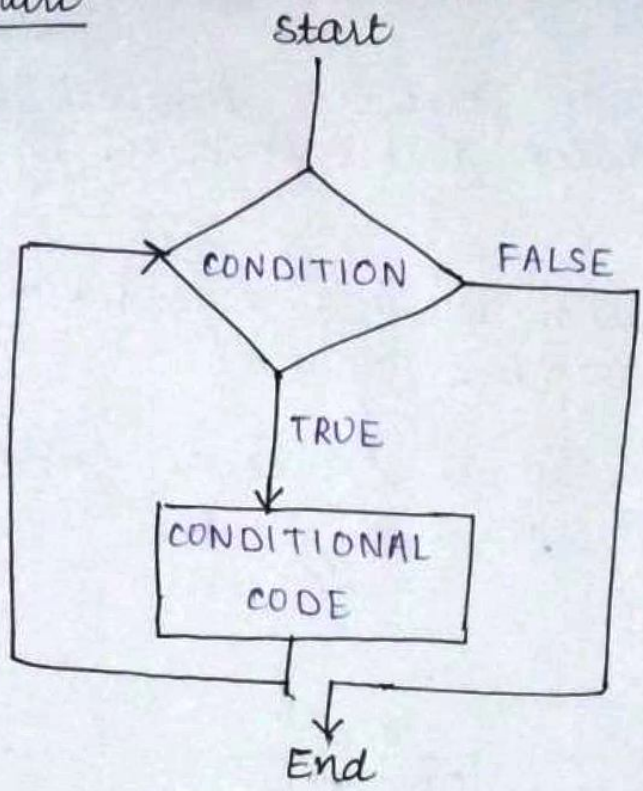
while loop statement in python programming language repeatedly executes a target statement as long as given condition is true

Syntax

```
while expression :
    statement(s)
```

The loop iterates while the condition is TRUE.
When the condition become FALSE loop terminates & control passes to the next immediate line.

Flow chart



using 'else' with 'while' loop

If the else statement is used with the while loop, else statement will executed when condition becomes false.

Example

```
count = 2
while count > 0 :
        print ("Inside while")
        count -= 1

else :
        print ("Exiting the while loop")
```

O/P

```
Inside while
Inside while
Exiting the while loop
```

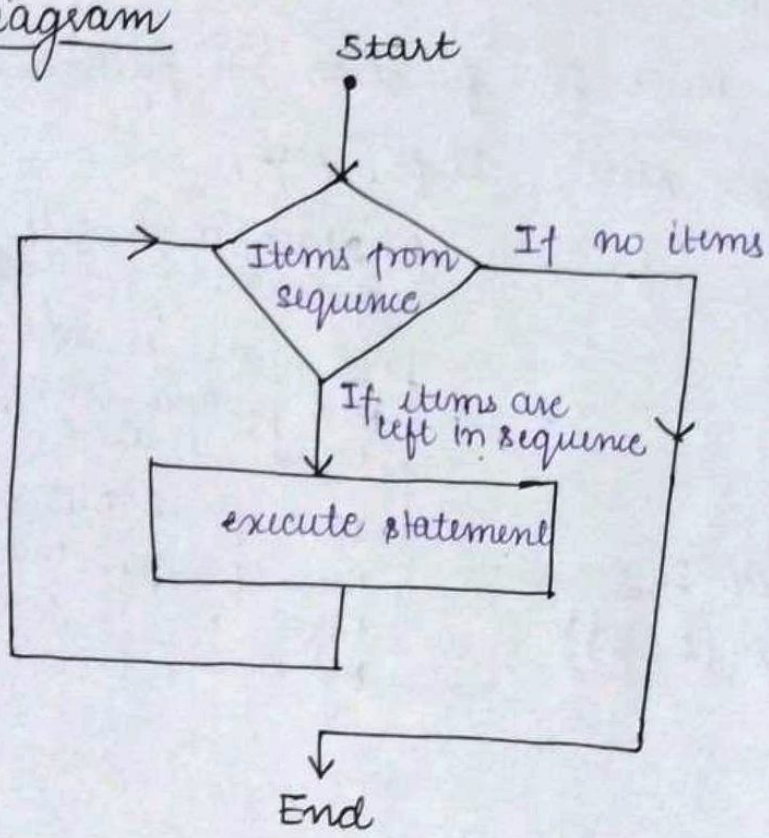
for loop statements

The for statement in Python has ability to iterate over the items of any sequence such as a list or a string.

syntax

```
for <iterating_variable> in sequence :
        statements)
```

Flow Diagram



Ex → print the characters in string 'PYTHON' using for loop.

```
for letter in 'PYTHON' :
        print (letter)
```

O/P

```
P
Y
T
H
O
N
```

Ex print sequence using for loop.

```
for num in [10,20,30,40,50]:
    print(num)
```

O/P
```
10
20
30
40
50
```

The range() fun^n

The range is a built-in function in python.

Syntax

range(start, stop, step)

Examples

```
>>> range(5)
range(0,5)
>>> list(range(5))
[0,1,2,3,4]

# default step value is 1
    >>> list(range(1,5))
    [1,2,3,4]

# step value is 2
    >>> list(range(1,6,2))
    [1,3,5]
```

- start and step are optional in range function
- if start is not given then python interpreter takes it as 0.
- if step is not given then python interpreter takes it as incremented by 1.

'else' with 'for' loop

else statement will be executed after for loop body

Ex
```
for i in [10,20,30]:
    print(i)
else:
    print("Exiting the for loop")
```

O/P
```
10
20
30
Exiting the for loop
```

Nested Loop

We can use nested looping in python. We can use one or more loop inside any another while or for loop.

Loop control statements

Loop control statements change the execution from its normal sequence. When the execution leaves a scope, all automatic objects that were created in that scope are destroyed.

Python supports the following control statements

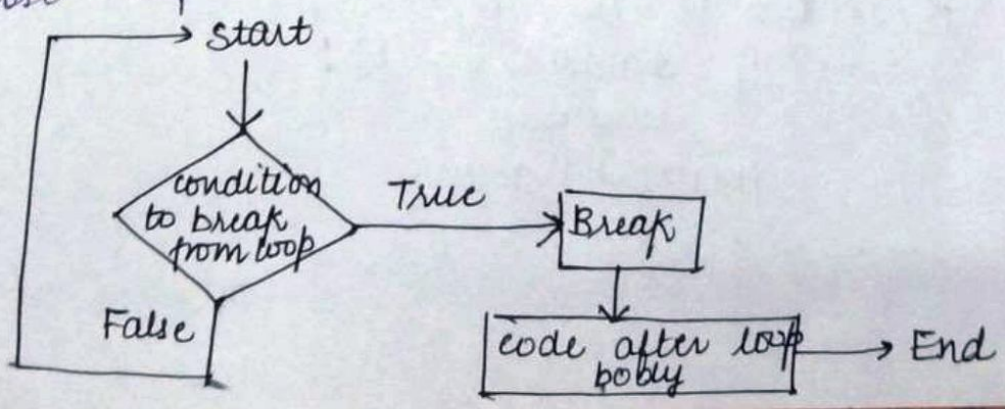
(i) Break statement

(ii) continue statement

(iii) pass statement

(i) Break statement

- Break allows us to exit a loop when an external condition is met.
- Normal program execution resumes the next statement
- we can use a 'break statement' with both for & while loops.
- In a nested loop, break will stop execution of innermost loop.



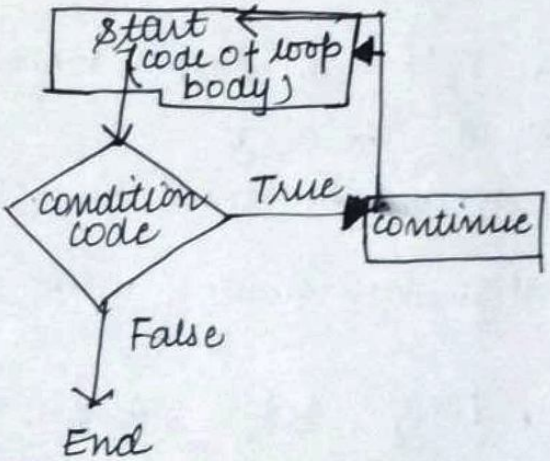
Example

```
for i in range(10):
    print(i)
    if (i==2):
        break
```

o/p
0
1
2

(ii) continue statement

- The continue statement in Python returns the control to the beginning of the current loop.
- When continue encounters loop starts next iteration without executing the remaining statements.
- continue statement can be used in while & for loop.
- In nested loops, continue will stop execution of current iteration & will execute the next one.



Example

```
for character in 'Python':
    if character == 't':
        continue
    print(character)
```

o/p
P
y
h
o
n

(iii) pass statement

- It is used when a statement is required syntactically but you do not want any command or code to execute.
- The pass statement is a null operation, nothing happens when it executes

Example

```
for character in 'python':
    if character == 'y':
        pass          # do nothing
        print('Inside pass block')
    print(character)
```

O/P

```
P
Inside pass block
y
t
h
o
n
```

Programs

1. WAP to print even numbers.

```
n = int(input("Enter end index : "))
for i in range(2, n+1):
    if (i%2 == 0):
        print(i, end = " ")
```

O/P

```
Enter end index : 10
2 4 6 8 10
```

2. WAP to print odd numbers

```
n = int( input (" Enter end index :"))
for i in range (1, n+1):
    if ( i%2 != 0):
        print( i , end = " ")
```

o/p
```
Enter end index : 10
1 3 5 7 9
```

3. WAP to find factorial of a number

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

print("factorial of %d is %d " %(n , fact))
```

o/p
```
Enter any number : 6
factorial of 6 is 720
```

4. WAP to calculate length of string without using in-built function.

```
str1 = input (" Enter any string:")
count = 0
for i in str1 :
    count += 1
print(" length of string is %d " %count)
```

o/p
```
Enter any string : python
length of string is 6
```

5. WAP to count number of digits in a number.

```
n = int(input("Enter any number :"))
m = n
count = 0
while(m) != 0):
        count += 1
        m //= 10
print("There are %d digits in %d" %(count,n))
```

O/P Enter any number : 1234
There are 4 digits in 1234

6. WAP to add digits in a numbers.

```
n = int(input("Enter any number : "))
m = n
sum = 0
while(m != 0):
        remain = m% 10
        sum += remain
        m //= 10
print("sum is %d " % sum)
```

O/P Enter any number : 123
sum is 6

7. WAP to check whether number is palindrome or not.

```
n = int(input("Enter any number : "))
m = n
reverse = 0
while(m!=0):
    remain = m%10
    reverse = reverse * 10 + remain
    m //= 10
if(reverse == n):
    print("Palindrome")
else:
    print("Not Palindrome")
```

O/P Enter any number : 121
Palindrome

8. WAP to print this pattern

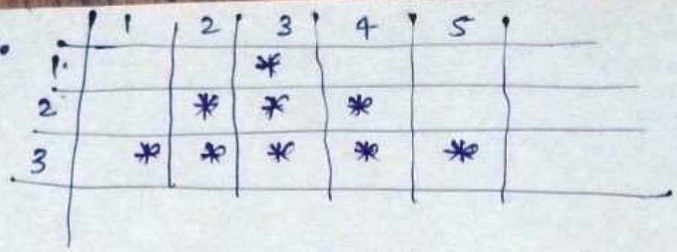
```
*
* *
* * *
```

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

O/P Enter number of lines: 3

```
*
* *
* * *
```

8.



If we have 3 rows

the column = (3) * 2 - 1 => 6 - 1 = 5

Rows

Row → for i in range(1, row+1):

column → for j in range(1, row*2 -1)+1)

Row	column
1	3
2	2, 3, 4
3	1, 2, 3, 4, 5

Row 1, print at column 3

Row 2, print at column 2,3,4

Row 3, print at column 1,2,3,4,5

if (i+j-1 >= row) start condition

if (j-i+1 <= row) end condition