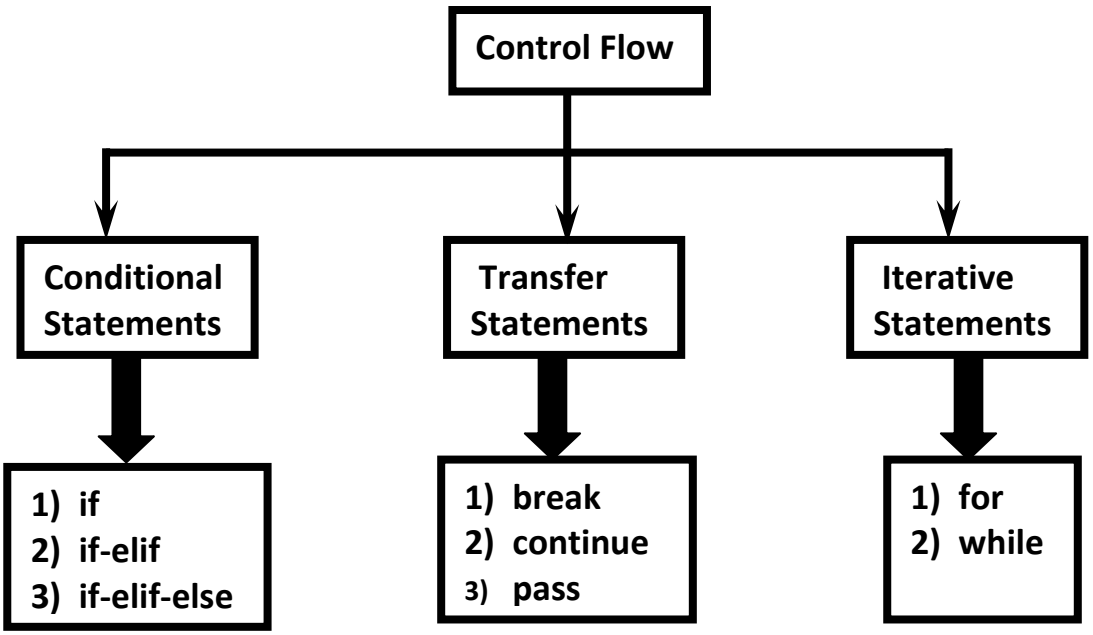
Flow Control

Flow control describes the order in which statements will be executed at runtime.



I. Conditional Statements

1) if

if condition : statement

or

if condition :
statement-1
statement-2
statement-3

If condition is true then statements will be executed.

Eg:

```
1)  name=input("Enter Name:")
2)  if name=="durga" :
3)      print("Hello Durga Good Morning")
4)  print("How are you!!!")
5)
6)  D:\Python_classes>py test.py
7)  Enter Name:durga
8)  Hello Durga Good Morning
9)  How are you!!!
10)
11) D:\Python_classes>py test.py
12) Enter Name:Ravi
13) How are you!!!
```

2) if-else:

if condition :
**   Action-1**
else :
**   Action-2**

if condition is true then Action-1 will be executed otherwise Action-2 will be executed.

Eg:

```
1)  name=input("Enter Name:")
2)  if name=="durga" :
3)      print("Hello Durga Good Morning")
4)  else:
5)     print("Hello Guest Good Moring")
6)  print("How are you!!!")
7)
8)  D:\Python_classes>py test.py
9)  Enter Name:durga
10) Hello Durga Good Morning
11) How are you!!!
12)
13) D:\Python_classes>py test.py
14) Enter Name:Ravi
15) Hello Guest Good Moring
16) How are you!!!
```

3) if-elif-else:

Syntax:

```
if condition1:
   Action-1
elif condition2:
   Action-2
elif condition3:
   Action-3
 elif condition4:
   Action-4
   ...
else:
   Default Action
```

Based condition the corresponding action will be executed.

Eg:

```
1)  brand=input("Enter Your Favourite Brand:")
2)  if brand=="RC" :
3)      print("It is childrens brand")
4)  elif brand=="KF":
5)     print("It is not that much kick")
6)  elif brand=="FO":
7)     print("Buy one get Free One")
8)  else :
9)     print("Other Brands are not recommended")
10)
11)
12) D:\Python_classes>py test.py
13) Enter Your Favourite Brand:RC
14) It is childrens brand
15)
16) D:\Python_classes>py test.py
17) Enter Your Favourite Brand:KF
18) It is not that much kick
19)
20) D:\Python_classes>py test.py
21) Enter Your Favourite Brand:KALYANI
22) Other Brands are not recommended
```

Note:

1. else part is always optional
   Hence the following are various possible syntaxes.
   1. if
   2. if - else
   3. if-elif-else
   4.if-elif

2. There is no switch statement in Python

Q. Write a program to find biggest of given 2 numbers from the commad prompt?

```
1)  n1=int(input("Enter First Number:"))
2)  n2=int(input("Enter Second Number:"))
3)  if n1>n2:
4)      print("Biggest Number is:",n1)
5)  else :
6)      print("Biggest Number is:",n2)
7)
8)   D:\Python_classes>py test.py
9)  Enter First Number:10
10) Enter Second Number:20
11) Biggest Number is: 20
```

Q. Write a program to find biggest of given 3 numbers from the commad prompt?

```
1)  n1=int(input("Enter First Number:"))
2)  n2=int(input("Enter Second Number:"))
3)  n3=int(input("Enter Third Number:"))
4)  if n1>n2 and n1>n3:
5)      print("Biggest Number is:",n1)
6)  elif n2>n3:
7)      print("Biggest Number is:",n2)
8)  else :
9)      print("Biggest Number is:",n3)
10)
11) D:\Python_classes>py test.py
12) Enter First Number:10
13) Enter Second Number:20
14) Enter Third Number:30
15) Biggest Number is: 30
16)
17) D:\Python_classes>py test.py
18) Enter First Number:10
```

```
Enter Second Number:30
Enter Third Number:20
Biggest Number is: 30
```

Q. Write a program to find smallest of given 2 numbers?
Q. Write a program to find smallest of given 3 numbers?
Q. Write a program to check whether the given number is even or odd?
Q. Write a program to check whether the given number is in between 1 and 100?

```
n=int(input("Enter Number:"))
if n>=1 and n<=10 :
    print("The number",n,"is in between 1 to 10")
else:
    print("The number",n,"is not in between 1 to 10")
```

Q. Write a program to take a single digit number from the key board and print is value in English word?

```
0==>ZERO
1 ==>ONE

n=int(input("Enter a digit from o to 9:"))
if n==0 :
    print("ZERO")
elif n==1:
    print("ONE")
elif n==2:
    print("TWO")
elif n==3:
    print("THREE")
elif n==4:
    print("FOUR")
elif n==5:
    print("FIVE")
elif n==6:
    print("SIX")
elif n==7:
    print("SEVEN")
elif n==8:
    print("EIGHT")
elif n==9:
    print("NINE")
else:
    print("PLEASE ENTER A DIGIT FROM 0 TO 9")
```

II. Iterative Statements

If we want to execute a group of statements multiple times then we should go for Iterative statements.

Python supports 2 types of iterative statements.

1. for loop
2. while loop

1) for loop:

If we want to execute some action for every element present in some sequence(it may be string or collection)then we should go for for loop.

Syntax:

```
for x in sequence :
   body
```

where sequence can be string or any collection.
Body will be executed for every element present in the sequence.

Eg 1: To print characters present in the given string

```
1)  s="Sunny Leone"
2)  for x in s :
3)      print(x)
4)
5)  Output
6)  S
7)  u
8)  n
9)  n
10) y
11)
12) L
13) e
14) o
15) n
16) e
```

Eg 2: To print characters present in string index wise:

```
1)  s=input("Enter some String: ")
2)  i=0
3)  for x in s :
4)      print("The character present at ",i,"index is :",x)
5)      i=i+1
6)
7)
8)   D:\Python_classes>py test.py
9)  Enter some String: Sunny Leone
10) The character present at  0 index is : S
11) The character present at  1 index is : u
12) The character present at  2 index is : n
13) The character present at  3 index is : n
14) The character present at  4 index is : y
15) The character present at  5 index is :
16) The character present at  6 index is : L
17) The character present at  7 index is : e
18) The character present at  8 index is : o
19) The character present at  9 index is : n
20) The character present at  10 index is : e
```

Eg 3: To print Hello 10 times

```
1)  for x in range(10) :
2)      print("Hello")
```

Eg 4: To display numbers from 0 to 10

```
1)      for x in range(11) :
2)      print(x)
```

Eg 5: To display odd numbers from 0 to 20

```
1)      for x in range(21) :
2)      if (x%2!=0):
3)          print(x)
```

Eg 6: To display numbers from 10 to 1 in descending order

```
1)          for x in range(10,0,-1) :
2)            print(x)
```

Eg 7: To print sum of numbers presenst inside list

```
1)          list=eval(input("Enter List:"))
2)      sum=0;
3)      for x in list:
4)          sum=sum+x;
5)      print("The Sum=",sum)
6)
7)  D:\Python_classes>py test.py
8)  Enter List:[10,20,30,40]
9)  The Sum= 100
10)
11) D:\Python_classes>py test.py
12) Enter List:[45,67]
13) The Sum= 112
```

2) while loop:

If we want to execute a group of statements iteratively until some condition false,then we should go for while loop.

Syntax:

```
while condition :
   body
```

Eg: To print numbers from 1 to 10 by using while loop

```
1)      x=1
2)      while x <=10:
3)      print(x)
4)      x=x+1
```

Eg: To display the sum of first n numbers

```
1)  n=int(input("Enter number:"))
2)  sum=0
3)  i=1
4)  while i<=n:
5)      sum=sum+i
6)      i=i+1
7)  print("The sum of first",n,"numbers is :",sum)
```

Eg: write a program to prompt user to enter some name until entering Durga

```
1) name=""
2) while name!="durga":
3)     name=input("Enter Name:")
4) print("Thanks for confirmation")
```

Infinite Loops:

```
1) i=0;
2) while True :
3)     i=i+1;
4)     print("Hello",i)
```

Nested Loops:

Sometimes we can take a loop inside another loop,which are also known as nested loops.

Eg:

```
1) for i in range(4):
2)     for j in range(4):
3)         print("i=",i,"   j=",j)
4)
5) Output
6) D:\Python_classes>py test.py
7) i= 0   j= 0
8) i= 0   j= 1
9) i= 0   j= 2
10) i= 0   j= 3
11) i= 1   j= 0
12) i= 1   j= 1
13) i= 1   j= 2
14) i= 1   j= 3
15) i= 2   j= 0
16) i= 2   j= 1
17) i= 2   j= 2
18) i= 2   j= 3
19) i= 3   j= 0
20) i= 3   j= 1
21) i= 3   j= 2
22) i= 3   j= 3
```

Q. Write a program to dispaly *'s in Right angled triangled form

```
1)  *
2)  * *
3)  * * *
4)  * * * *
5)  * * * * *
6)  * * * * * *
7)  * * * * * * *
8)
9)  n = int(input("Enter number of rows:"))
10) for i in range(1,n+1):
11)     for j in range(1,i+1):
12)         print("*",end=" ")
13)     print()
```

Alternative way:

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

Q. Write a program to display *'s in pyramid style(also known as equivalent triangle)

```
1)       *
2)      * *
3)     * * *
4)    * * * *
5)   * * * * *
6)  * * * * * *
7) * * * * * * *
8)
9)  n = int(input("Enter number of rows:"))
10) for i in range(1,n+1):
11)     print(" " * (n-i),end="")
12)     print("* "*i)
```

III. Transfer Statements

1) break:

We can use break statement inside loops to break loop execution based on some condition.

Eg:

```
1)  for i in range(10):
2)      if i==7:
3)          print("processing is enough..plz break")
4)          break
5)      print(i)
6)
7)  D:\Python_classes>py test.py
8)  0
9)  1
10) 2
11) 3
12) 4
13) 5
14) 6
15) processing is enough..plz break
```

Eg:

```
1)  cart=[10,20,600,60,70]
2)  for item in cart:
3)      if item>500:
4)          print("To place this order insurence must be required")
5)          break
6)      print(item)
7)
8)  D:\Python_classes>py test.py
9)  10
10) 20
11) To place this order insurence must be required
```

2) continue:

We can use continue statement to skip current iteration and continue next iteration.

Eg 1: To print odd numbers in the range 0 to 9

```
1)  for i in range(10):
2)      if i%2==0:
3)          continue
4)      print(i)
5)
6)  D:\Python_classes>py test.py
7)  1
8)  3
9)  5
10) 7
11) 9
```

Eg 2:

```
1)  cart=[10,20,500,700,50,60]
2)  for item in cart:
3)      if item>=500:
4)          print("We cannot process this item :",item)
5)          continue
6)      print(item)
7)
8)  Output
9)  D:\Python_classes>py test.py
10) 10
11) 20
12) We cannot process this item : 500
13) We cannot process this item : 700
14) 50
15) 60
```

Eg 3:

```
1)  numbers=[10,20,0,5,0,30]
2)  for n in numbers:
3)      if n==0:
4)          print("Hey how we can divide with zero..just skipping")
5)          continue
6)      print("100/{} = {}".format(n,100/n))
7)
```

```
Output

100/10 = 10.0
100/20 = 5.0
Hey how we can divide with zero..just skipping
100/5 = 20.0
Hey how we can divide with zero..just skipping
100/30 = 3.3333333333333335
```

loops with else block:

Inside loop execution,if break statement not executed ,then only else part will be executed.

else means loop without break

Eg:

```
cart=[10,20,30,40,50]
for item in cart:
    if item>=500:
        print("We cannot process this order")
        break
    print(item)
else:
    print("Congrats ...all items processed successfully")

Output
10
20
30
40
50
Congrats ...all items processed successfully
```

Eg:

```
cart=[10,20,600,30,40,50]
for item in cart:
    if item>=500:
        print("We cannot process this order")
        break
    print(item)
else:
    print("Congrats ...all items processed successfully")
```

```
9)
10) Output
11) D:\Python_classes>py test.py
12) 10
13) 20
14) We cannot process this order
```

Q. What is the difference between for loop and while loop in Python?

We can use loops to repeat code execution
Repeat code for every item in sequence ==>for loop
Repeat code as long as condition is true ==>while loop

Q. How to exit from the loop?
by using break statement

Q. How to skip some iterations inside loop?
by using continue statement.

Q. When else part will be executed wrt loops?
If loop executed without break

3) pass statement:

pass is a keyword in Python.

In our programming syntactically if block is required which won't do anything then we can define that empty block with pass keyword.

pass
|- It is an empty statement
|- It is null statement
|- It won't do anything

Eg:

```
if True:
SyntaxError: unexpected EOF while parsing

if True: pass
==>valid

def m1():
SyntaxError: unexpected EOF while parsing
```

def m1(): pass

use case of pass:

Sometimes in the parent class we have to declare a function with empty body and child class responsible to provide proper implementation. Such type of empty body we can define by using pass keyword. (It is something like abstract method in java)

Eg:

```
def m1(): pass
```

Eg:

```
1)  for i in range(100):
2)      if i%9==0:
3)          print(i)
4)      else:pass
5)
6)  D:\Python_classes>py test.py
7)  0
8)  9
9)  18
10) 27
11) 36
12) 45
13) 54
14) 63
15) 72
16) 81
17) 90
18) 99
```

del statement:

del is a keyword in Python.

After using a variable, it is highly recommended to delete that variable if it is no longer required,so that the corresponding object is eligible for Garbage Collection.
We can delete variable by using del keyword.

Eg:

```
1)  x=10
2)  print(x)
3)  del x
```

After deleting a variable we cannot access that variable otherwise we will get NameError.

Eg:

```
1) x=10
2) del x
3) print(x)
```

NameError: name 'x' is not defined.

Note:

We can delete variables which are pointing to immutable objects.But we cannot delete the elements present inside immutable object.

Eg:

```
1) s="durga"
2) print(s)
3) del  s==>valid
4) del s[0] ==>TypeError: 'str' object doesn't support item deletion
```

Difference between del and None:

In the case del, the variable will be removed and we cannot access that variable(unbind operation)

```
1) s="durga"
2)  del s
3)  print(s)      ==>NameError: name 's' is not defined.
```

But in the case of None assignment the variable won't be removed but the corresponding object is eligible for Garbage Collection(re bind operation). Hence after assigning with None value,we can access that variable.

```
1) s="durga"
2) s=None
3) print(s)    # None
```