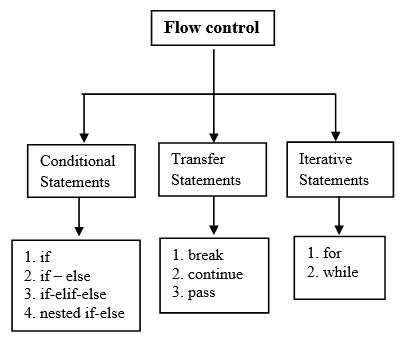
## Control Flow Statements

## The flow control statements are divided into **three categories**

1. Conditional statements
2. Iterative statements.
3. Transfer statements

Python control flow statements

**Conditional statements**

In Python, condition statements act depending on whether a given condition is true or false. You can execute different blocks of codes depending on the outcome of a condition. Condition statements always evaluate to either True or False.

## If statement in Python

In control statements, The if statement is the simplest form. It takes a condition and evaluates to either True or False.

If the condition is True, then the True block of code will be executed, and if the condition is False, then the block of code is skipped, and The controller moves to the next line

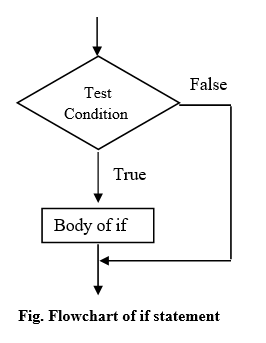
**Syntax of the if statement**

**if** condition:

statement 1

statement 2

statement n



Let’s see the example of the if statement. In this example, we will calculate the square of a number if it greater than 5

**Example**

number = 6

**if** number > 5:

# Calculate square

**print**(number \* number)

**print**('Next lines of code')

### If – else statement

The if-else statement checks the condition and executes the if block of code when the condition is True, and if the condition is False, it will execute the else block of code.

**Syntax of the if-else statement**

**if** condition:

statement 1

**else**:

statement 2

### if-elif-else

In Python, the if-elif-else condition statement has an elif blocks to chain multiple conditions one after another. This is useful when you need to check multiple conditions.

With the help of if-elif-else we can make a tricky decision. The elif statement checks multiple conditions one by one and if the condition fulfills, then executes that code.

Syntax of the if-elif-else statement:

**if** condition-1:

statement 1

**elif** condition-2:

stetement 2

**elif** condition-3:

stetement 3

...

**else**:

statement

**Example**

**def** user\_check(choice):

**if** choice == 1:

**print**("Admin")

**elif** choice == 2:

**print**("Editor")

**elif** choice == 3:

**print**("Guest")

**else**:

**print**("Wrong entry")

user\_check(1)

user\_check(2)

user\_check(3)

user\_check(4)

### Nested if-else statement

In Python, the nested if-else statement is an if statement inside another if-else statement. It is allowed in Python to put any number of if statements in another if statement.

Indentation is the only way to differentiate the level of nesting. The nested if-else is useful when we want to make a series of decisions.

**Syntax of the nested-if-else:**

**if** conditon\_outer:

**if** condition\_inner:

statement of inner **if**

**else**:

statement of inner **else**:

statement ot outer **if**

**else**:

Outer **else**

statement outside **if** block

**Example**: Find a greater number between two numbers

num1 = **int**(**input**('Enter first number '))

num2 = **int**(**input**('Enter second number '))

**if** num1 >= num2:

**if** num1 == num2:

**print**(num1, 'and', num2, 'are equal')

**else**:

**print**(num1, 'is greater than', num2)

**else**:

**print**(num1, 'is smaller than', num2)

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

19) Enter Second Number:30

20) Enter Third Number:20

21) 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?

1) n=int(input("Enter Number:"))

2) if n>=1 and n<=10 :

3) print("The number",n,"is in between 1 to 10")

4) else:

5) print("The number",n,"is not in between 1 to 10")

Instead of writing a block after the colon, we can write a statement immediately after the colon.

**Example**

number = 56

**if** number > 0: **print**("itvedant")

**else**: **print**("itvedant")

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:

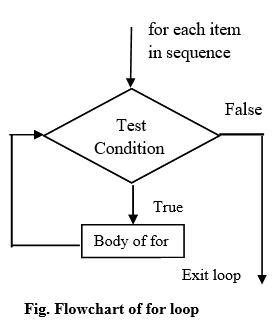
If we want to execute some action for every element present in some sequence (it may be string or collection) then we should go for for loop.

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

Syntax:

for x in sequence:

body of for loop



Eg 2: To print characters present in the given string

1) s="itvedant"

2) for x in s :

3) print(x)

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)

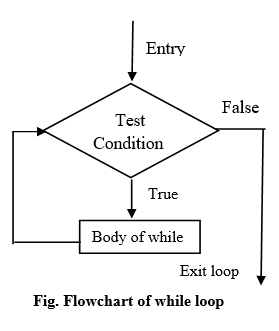
Eg 6: To display numbers from 10 to 1 in descending order

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

2) print(x)

While Loop:

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

In a while-loop, every time the condition is checked at the beginning of the loop, and if it is true, then the loop’s body gets executed. When the condition became False, the controller comes out of the block. Python while loop

Syntax:

while condition :

body of while loop

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 Itvedant

1) name=""

2) while name!="Itvedant":

3) name=input("Enter Name:")

4) print("Thanks for confirmation")

Infinite Loops:

) 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)

Output:

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) \* \* \* \* \* \* \*

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()

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)

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) #out[ut

7) 1

8) 3

9) 5

10) 7

11) 9

2) pass:

pass is a keyword in Python.

When nothing is happened inside the block, we use pass statement

Eg:

if True:

Syntax Error: unexpected EOF while parsing

if True:

pass ==>valid

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

Important question

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