**Input function:**

input() : this is user input method

name=input(“enter your name”)

print(name)

age=int(input(“enter your age”))

print(age)

## Decision Making Statements:(if …else)

Syntax:

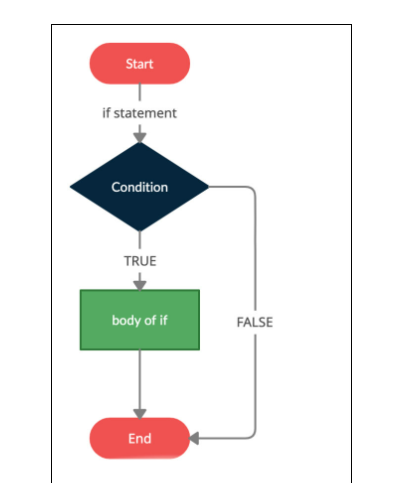
if (condition):

statement 1

statement 2

…

Flow chart:



Types:

1. if
2. if else
3. if elif
4. Nested if
5. **if :**

x=5

if (x>0):

print(“positive number”)

**2.if else:**

x=-5

if (x>0):

print(“positive number”)

else:

print(“negative number”)

**3.if elif :**

x=0

if (x>0):

print(“positive number”)

elif(x==0):

print(“value is Zero”)

else:

print(“negative number”)

**4.**nested if:

**num = 15**

**if num >= 0:**

**if num == 0:**

**print("Zero")**

**else:**

**print("Positive number")**

**else:**

**print("Negative number")**

**-----------------------------------------------------------------------------**

**Looping statement:**

Python has two primitive loop commands:

* while loops
* for loops

**while loops:**

With the while loop we can execute a set of statements as long as a condition is true.

**Ex:**

i = 1  
while i < 6:  
  print(i)  
  i += 1

**For Loops**

A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).keyword is **for**

**Ex:**

**1.**

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

**2.**

**for x in "banana":  
  print(x)**

**3.**

**for x in range(2, 6):  
  print(x)**

**4.**

**for x in range(2, 30, 3):  
  print(x)**

## ---------------------------------------------------------------------------

## Jump Statements:

## 1.break

## 2.continue

## Examble

## Break statement:

## i = 1 while i < 6:   print(i)   if i == 3:     break   i += 1

## continue statement:

## i = 1 while i < 6:   print(i)   if i == 3:     continue   i += 1

## function:

## 1.Built in function

## 2. User defined function

## User defined function:

## In Python a function is defined using the def keyword:

### Example

### 

def my\_function():  
  print("Hello from a function")

## my\_function( )

## function creating method:

## 1.no argument no return

## 2.no argument return

## 3.argument no return

## 4.argument return

## Arbitrary argument method:

## Ex:

## def hello(\*names):

## for i in names:

## print(“hiii…”,i)

## hello(“karan”,”vignesh”,”raja”,”prasanth”)

Recursion

Python also accepts function recursion, which means a defined function can call itself.

Recursion is a common mathematical and programming concept

## def tri\_recursion(k):

## if(k==1):

## return 1

## else:

## return k\*tri\_recursion(k-1)

## print(tri\_recursion(5))

## lambda function:

A lambda function can take any number of arguments, but can only have one expression.

Syntax

lambda *arguments*: *expression*

***examble 1 :***

x = lambda a : a + 10  
 print(x(5))

## examble 2:

## x = lambda a : a + 10 print(x(5))

Why Use Lambda Functions?

The power of lambda is better shown when you use them as an **anonymous function** inside another function.

## def myfunc(n):    return lambda a : a \* n mydoubler = myfunc(2) print(mydoubler(11))

## lambda with map()examble 2:

animals = ['dog', 'cat', 'parrot', 'rabbit']

uppered\_animals = list(map(lambda animal: animal.upper(), animals))

print(uppered\_animals)

lambda with filter () examble 3:

ages = [13, 90, 17, 59, 21, 60, 5]

adults = list(filter(lambda age: age > 18, ages))

print(adults)

---------------------------------------------------------