Core Python By Sandeep Kumar Sharma

‘Give a man a fish and you feed him for a day.Teach a man to fish and to feed him for a lifetime.’

Introduction (I)

Computer

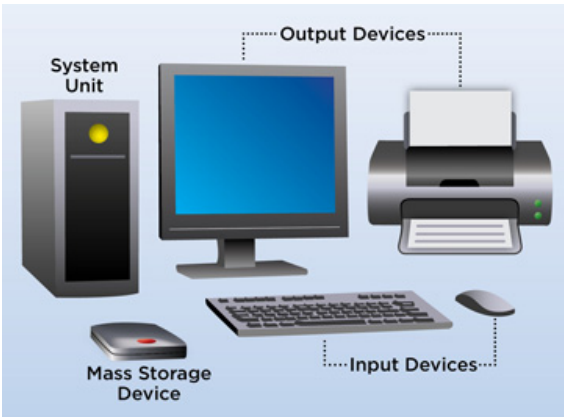
Any Calculating device is called Computer.

Definition - What does Computer mean?

A computer is a machine or device that performs processes, calculations and operations based on instructions provided by a software or hardware program. It is designed to execute applications and provides a variety of solutions by combining integrated hardware and software components.

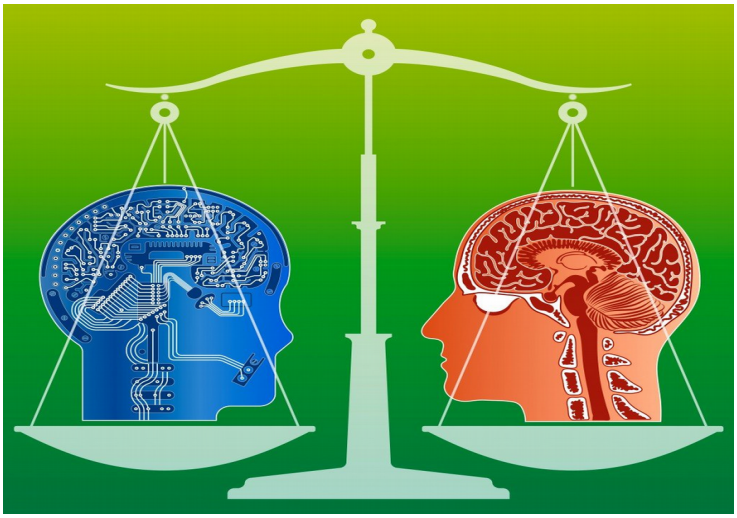
Full Form

C= Common
O= Oriented
M= Machine
P= Particularly
U= United and used under
T= Technical and
E= Educational
R= Research



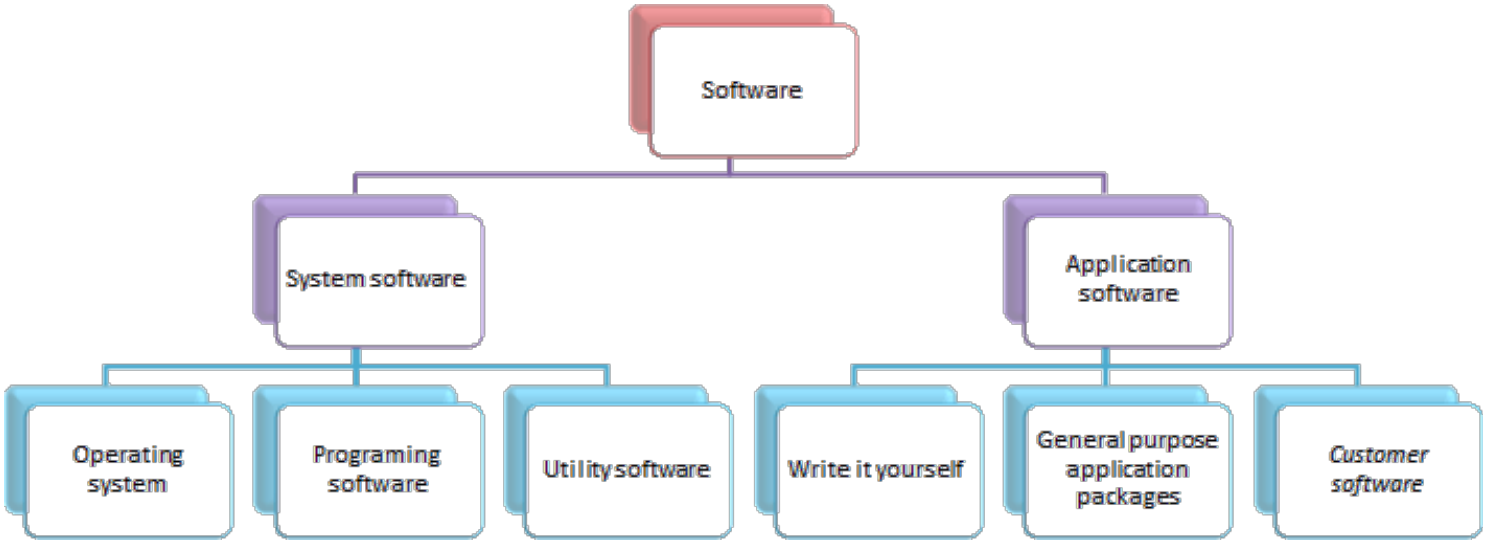
Brain Vs. Computer

Speed
Accuracy
Processing
Size and Complexity
Storage
Control Mechanism
Automation
Versatile
Diligency
Reliability



Software

Collection of programs is called Software.



Program

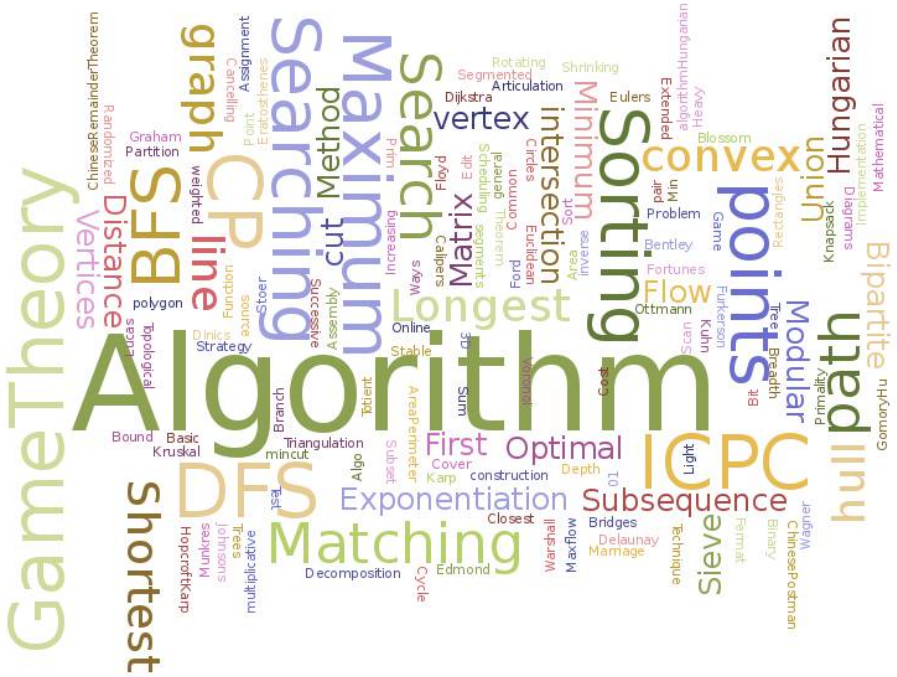
Collection of instructions is called pragram.

Programing

Solving the problem step by step is called is called Programming.

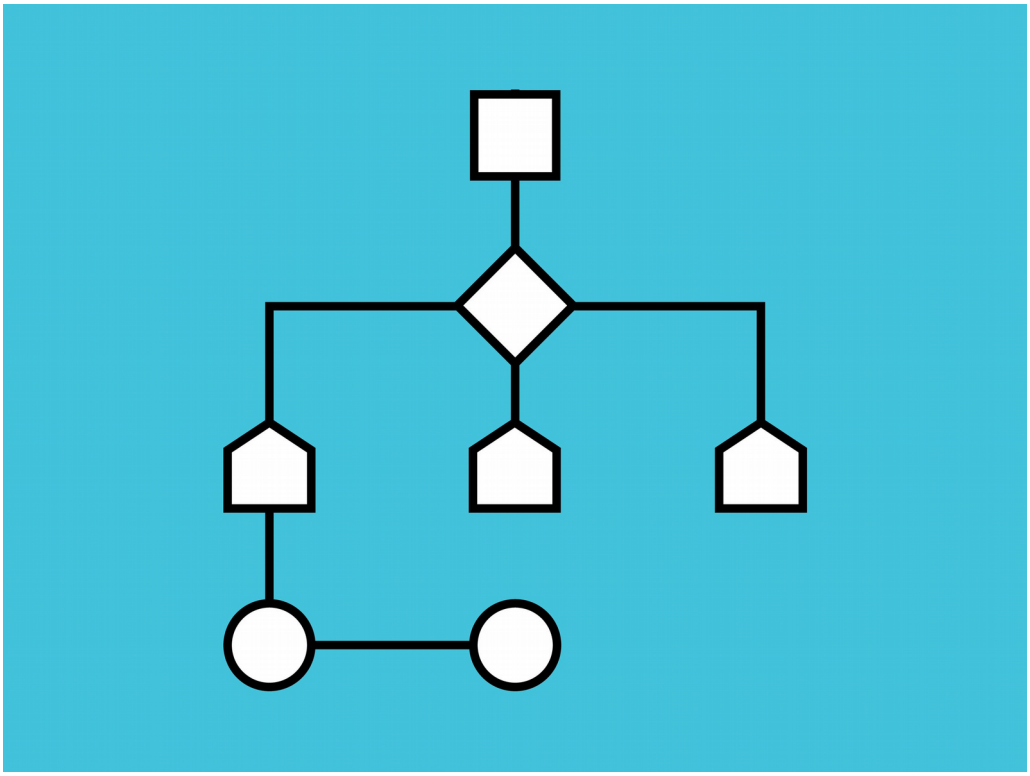
Algorithm

An algorithm (pronounced AL-go-rith-um) is a procedure or formula for solving a problem, based on conductiong a sequence of specified actions. A computer program can be viewed as an elaborate algorithm. In mathematics and computer science, an algorithm usually means a small procedure that solves a recurrent problem.



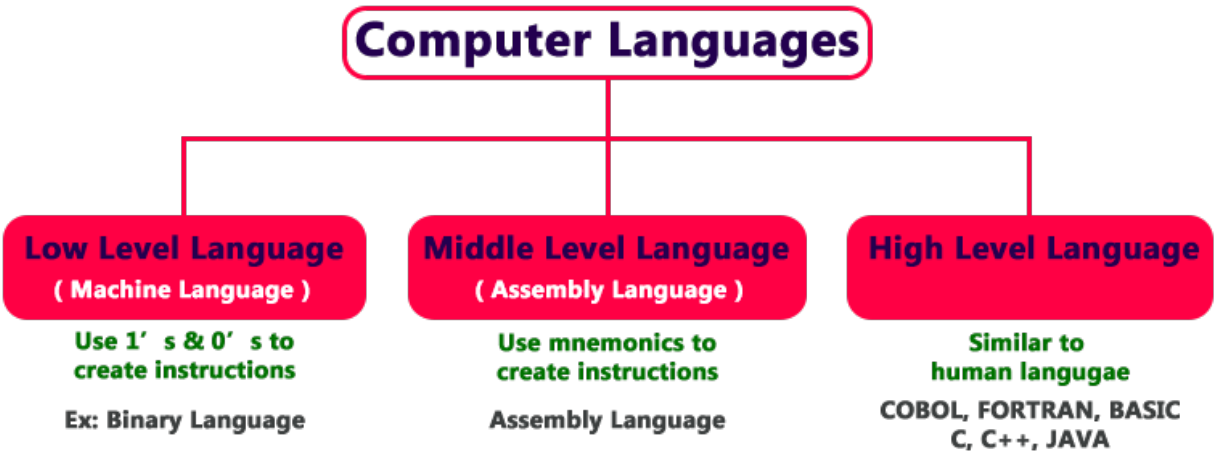
Flow Chart

A flowchart is a formalized graphic representation of a logic sequence, work or manufacturing process, organization chart, or similar formalized structure. The purpose of a flow chart is to provide people with a common language or reference point when dealing with a project or process.



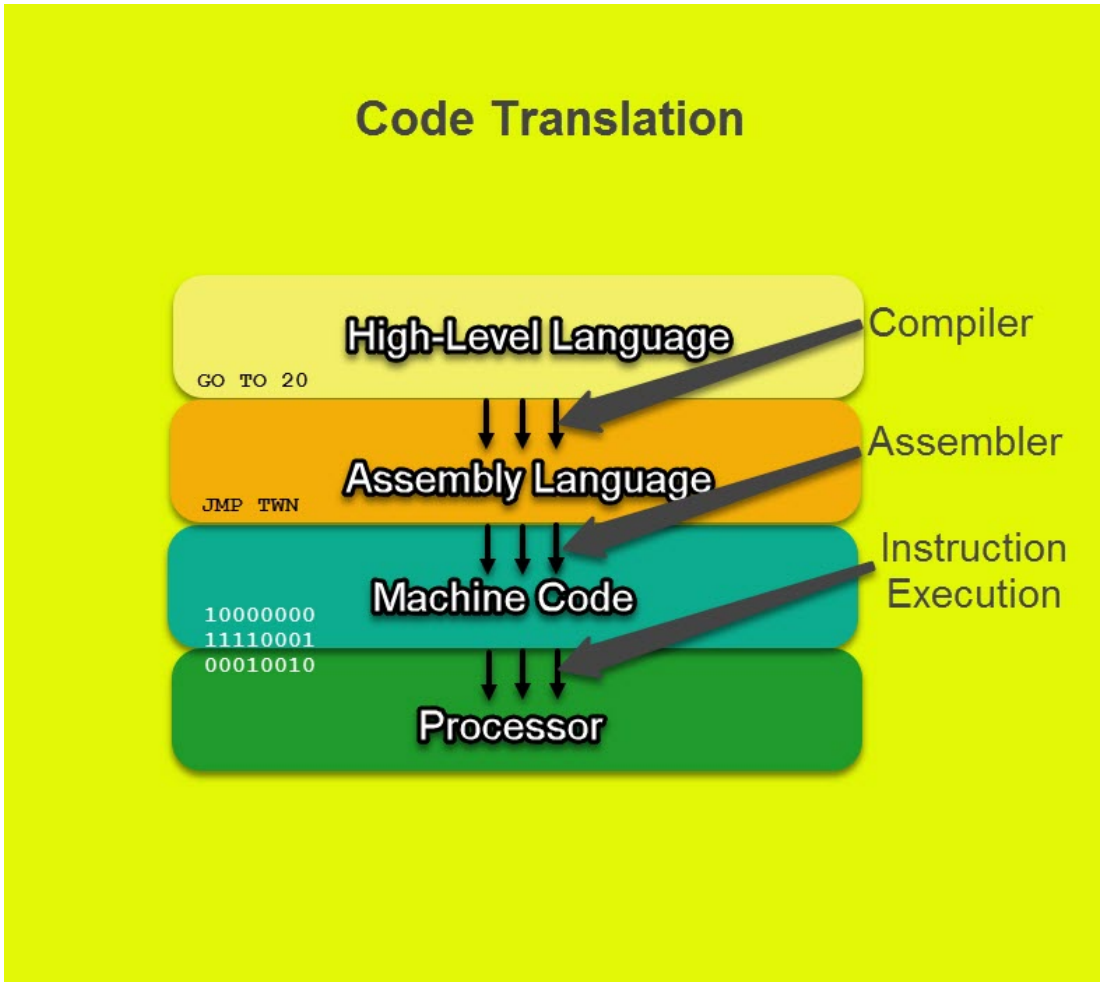
Programming Languages

There are two types of programming language.
1. Low level language (L.L.L)
2. High level language (H.L.L)



Translators

There are three types of translators
1. Assembler
2. Interpreter
3. Compiler



Assembler

It Converts the program writen in Assembly language to Low Level Language.

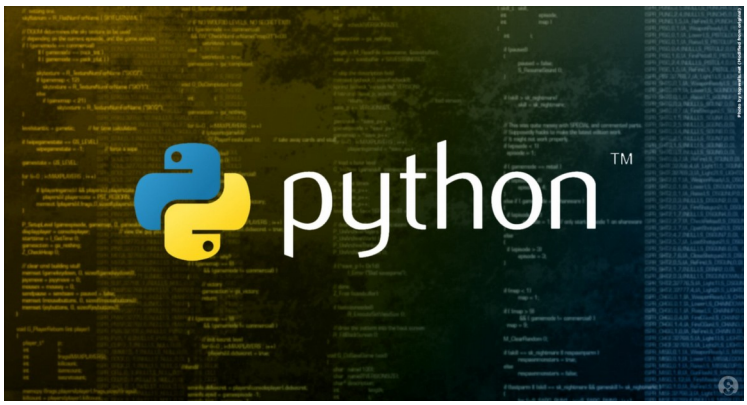
Interpreter

It converts the program written in High Level Language into Low Level Language line by line if any error in any line then Execution process will terminate.

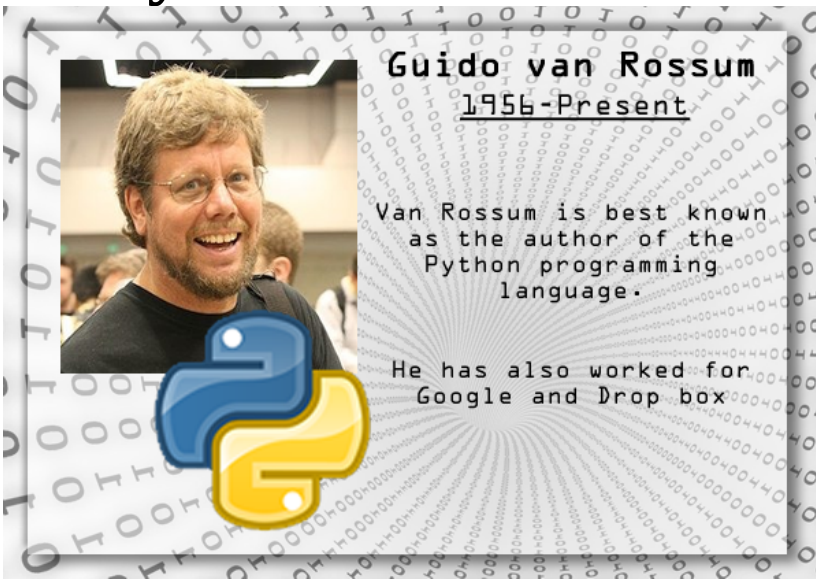
Compiler

It converts the program written in High Level Language into Low Level Language at one go. If there is any syntax error in program ,program won’t be execute.

Python



Father Of Python



Python is a general purpose, high level,dynamically typed programming language. It is developed by Guido Van Rossum in 1989 at NRI(National Research Institute) in Netherlands but Python was available in 1991.

Comparision

C

```
/* print hello world */
#include<stdio.h>
void main()
{
printf("Hello world ")
}
/* Output=Hello world */
```

Java

```
public class HelloWorld{

    public static void main(String []args){
       System.out.println("Hello World");
    }
}
```

Python

```
print('Hello world')
# Output Hello world
```

Addition of two numbers

C

```
 #include<stdio.h>
int main()
{
   int a, b, c;
   printf("Enter two numbers to add\n");
   scanf("%d%d", &a, &b);
   c = a + b;
   printf("Sum of the numbers = %d\n", c);
   return 0;
}
```

Python

```
a,b=[int (x) for x in input("Enter number:").split()]
print("The sum:",a+b)
```

Java

```
import java.util.Scanner;

class AddNumbers
{
   public static void main(String args[])
   {
      int x, y, z;

      System.out.println("Enter two integers to calculate their sum");
      Scanner in = new Scanner(System.in);

      x = in.nextInt();
```

```
    y = in.nextInt();
    z = x + y;

    System.out.println("Sum of the integers = " + z);
  }
}
```

Why Guido Sir named own Language "PYTHON"

At the time when he began implementing Python, Guido van Rossum was also reading the published scripts from "Monty Python's Flying Circus" (a BBC comedy series from the seventies, in the unlikely case you didn't know). It occurred to him that he needed a name that was short, unique, and slightly mysterious, so he decided to call the language Python.

Python borrow various features from different languages

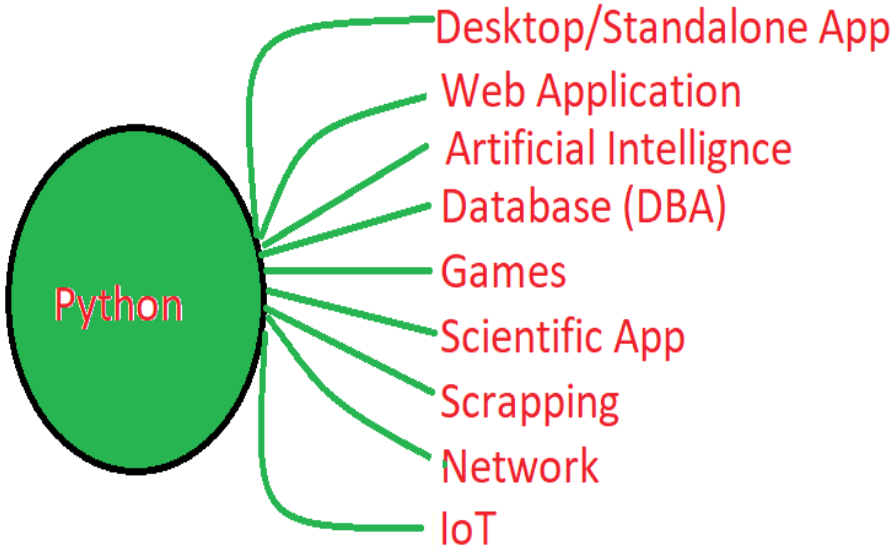
Functional programming (C language)
Object Oriented programming (C++)
Modular (Modula 3)
Scripting (Perl & Shell)

“There are many languages but there are approx 256 programming languages known in the world.”
https://www.tiobe.com/tiobe-index/programming-languages-definition/#instances

Applications Area’s of python

we can use python in various fields. Some Application areas of python are given below
1. Desktop appication/Standalone application
2. Web appication
3. Scrapping
4. Games
5. Scientific application
6. Artificial Intelligence (A.I.)
7. Machine Learning (M.L)
8. Data Science
9. IoT (Internet of Things)
10. Database

11. Socket Programming



Note:

Google,Youtube, Quora,Yahoo, Nasa ,IBM, Mozila foundation ,Insta organizations used python

Features of Python

1. Simple
2. Easy to learn
3. Functional programming
4. Object Oriented programming (C++)
5. Open source
6. Free ware
7. Dynamically typed language
8. Interpreted language
9. Extensiblle
10. Plateform independent

Limitations of python:

Performance is low incomparison with C and C++

Python Version

1.X (1994)
2.X (2000)
3.X (2008)

Current Version of Python is 3.7.0 (20-Jul-2018)

Python 2.X Vs. Python 3.X

Basics Of Programming

Identifiers (II)

A name in program which, can be used for identification purpose is called an Identifier.
It can be function name, variable name, method name, class name, module name.

```
i.e.  addition=20
      def shoaib():
        a,b=10,20
          print("Learning Python is Very Easy")
     # here 'addition, shoaib, a, b and print() 'are identifiers
```

Rules to define identifier in Python

1. Identifiers can be combination of uppercase and lowercase letters, digits or an underscore (_)sign so We can’t use special any symbols like !,#,@,%,$ etc in our Identifier.
2.An Identifier can not start with digit
3.We can't use reserve words as identifier.
4.Identifier can be of any length
5.Python is case sensitive so
  var1 & Var both are two different identifier (variable)

```
i.e. var    ------ valid
     var123   ------ valid
     Var121   ------ valid
     var_123  ------ valid
```

_Var245 ------ valid
2121Var245 ------ invalid
if=20   invalid
_Var&45 ------ invalid
_Var$45 ------ invalid

(III) Reserve words

Reserve words

Reserve words are also called keywords. It is predifne word which is used for do specific task in program.
Python has 33 Reserve words
All keywords in python are in lower case

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

Except following 3
True, False, None

WAP to view keyword in Python 3.X

```
from keyword import kwlist
print(kwlist)
#  Output
```

(IV) Data Types

Dynmically typed

A lot of people define static typing and dynamic typing with respect to the point at which the variable types are checked. Using this analogy, static typed languages are those in which type checking is done at compile-time, whereas dynamic typed languages are those in which type checking is done at run-time.

1. int
2. float
3. complex
4. bool
5. str
6. bytes
7. bytearray
8. list
9. tuple
10 set
11. dict

range() function

int data type

It is used to represent integer value.
we can assign integral literal in four ways

1. integral value
ie. a=500
type(a)

O/p--- <class 'int'>

2. in binary (allowed value 0 and 1)
we can specify binary literal with 0B/0b
i.e. a =0b11101 (valid)
c=0b11121 (invalid)

3. in octal (allowed values 0 to 7)
we can specify octal literal with 0o/0O
i.e. a =0o7754 (valid)
b=0O777 (valid)
c=0o74847 (invalid)

4. in hexa decimal (allowed values 0 to 9 and A to F / a to f)
we can specify hexa decimal literal with 0X/0x
i.e. a =0X77AB (valid)
b=0x777FEed (valid)
c=0x74847Rare (invalid)

float data type

It is used to represent float value.
we can assign float literal in two ways

ie. a=1.752
b=1.5326

a=1.732E2 (Scientific notation where e is exponetial)
b=2.7e3

complex data type

It is used to represent complex value

syntax

a+bj where a is real part b is imaginary part
i.e

3+5j
5+4j
2.5+6.25j

a=3+5j
a.real---------->3
a.imag---------->5

bool

it is used represent boolean value
Allowed value True and False
ie.
a=True (Valid)
type(a)
<class 'bool'>
a=False (Valid)
type(a)
<class 'bool'>
a=true (InValid)
a=false (InValid)

str

str is is used to represent String
String is collection of characters enclosed with either single quotes (') or in double quotes ("")
ie.
abc='Techavera Solutions Pvt Ltd'
type(abc)
<class 'str'>
abc="Techavera Solutions Pvt Ltd"
type(abc)
<class 'str'>

Multiline Strings

we can represent multiline string either triple single quotes (''' Techavera Python Trainer ''') or in triple double quotes (""" Techavera Python Trainer""")

i.e
abc='''Welcome To

```
        techavera
                Solutions
                        Pvt. Ltd.'''
print(type(abc))
#<class 'str'>
print(abc)
# Welcome To
        techavera
                Solutions
                        Pvt. Ltd.

abc="""Welcome To
        techavera
                Solutions
                        Pvt. Ltd."""
type(abc)
#<class 'str'>
print(abc)
#Output
Welcome To
        techavera
                Solutions
                        Pvt. Ltd.
```

(* imp) Note-

1. int,float,str, bool,complex are fundamental data type.
2. All fundamental data types are immutabel
3. In Python everything is an object.

Base Conversion

Base conversion is used to convert one base to another base(ie. binary to decimal)

1. bin() function

It is used to convert any base (decimal, octal, hexa decimal)to binary
ie. bin(7) ------- 0b111
bin(0o7)------- 0b111 { please fill output in all cases }
bin(0XA)-------0b1010

2. oct() function

It is used to convert any base (decimal, binary, hexa decimal)to octal
ie. oct(7)--------
oct(0b111)----0o7 { please fill output in all cases }
oct(0xA)------
3. hex() function
It is used to convert any base (decimal, octal, hexa decimal)to hexa decimal
ie. hex(7)--------
hex(0b111)---- { please fill output in all cases }
hex(0o777)------



(V) Type Casting

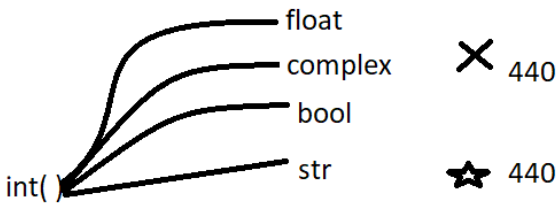
Type Casting

Python defines type conversion functions to directly convert one data type to another which is useful in day to day and competitive programming. This article is aimed at providing the information about certain conversion functions.

int()

It is used to convert another type to int.

ie. int(12.5) ---------->12
(imp) int(3+4j) ---------->Type Error
int(True) ---------->1
int(False) ---------->0
int('12') ---------->12 but in string only int value is required otherwise we will get value error.

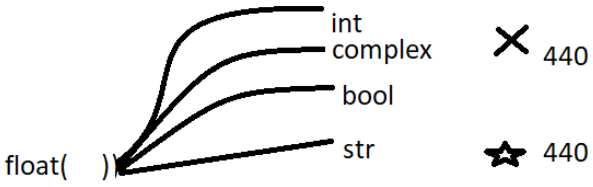


str to int is possible but in str is in only int form

float()

It is used to convert another type to float.

ie. float(12) ---------->12.0
(imp) float(3+4j) ---------->Type Error
float(True) ---------->1.0
float(False) ---------->0.0
float('12') ---------->12.0 but in string only int/float value is required otherwise we will get value error.



str to int is possible but in str is in only int form

bool()

It is used to
convert another type to bool.

for int argument

non-zero ---------->True
zero ---------->False

for float argument

non-zero ---------->True
zero ---------->False

for complex

if any one is non-zero ---------->True
otherwise ---------->False

for non-empty string

for empty str ---------->False
non-empty ---------->False

complex()

It is used to convert another type to complex.

syntax

complex(x) complex(2) ------->2+0j
complex(x,y)
complex(3,4)--------->3+4j

str()

It is used to convert another type to str.

ie. str(12.5) ---------> '12.5'
str(3+4j) ---------> '3+4j'
str(True) ---------> 'True'
str(False) --------->'False'
str('12') ---------> '12'

range()

It is used to generate some sequence of value.
It is applicable only for integer.

range(end)
|
end=end-1

range(9)--------> output------>0,1,2.....,9

range(begin,end)
| |
0 end-1

ie range(1,10)
1,2,3.....9
ie range(10,20)
10,20,30,.....19

Complete Form Of range()

range(begin,end,step)

ie. range(1,10,2)

(VI) Operators

Operators are symbols used to perform certain operations
There are variouos operators in Python

1. Arithmatic Operators
2. Relational Operators/Comparison Operators
3. Equality Operator
4. Logical Operators
5. Bitwise Operators
6. Assignment Operators
7. Special operators

1. Arithmatic Operators

\+ ----------> for addition
\- ----------> for subtraction
* ----------> for multiplication
/ ----------> for division
% ----------> for remainder (modulo)
// -------> floor division
** ----------> power operator

floor Division (//)

If both operands are int then result is int

ie. 85//2 -------> 42

89//3 -------> 29

If both operands are float then result is float but in nearest to the int

ie. 82.5//5.3 ---------->please fill output
89//3 -------> 29
824.5//5.43 ---------->please fill output

If both operands are different then result is float

ie. 82//32.5
30.5//2

power operator(**)

It is used to evaluate power of any number

ie. 10^2 ----------> 10**2 ---------->100
10^3 ----------> 10**3 ---------->1000

string concatenation (+)

"+" take all operand in string otherwise we will get error

ie. 'Python' + 'Programming' ----------> Python Programming
'Python' + 123 ----------Error
'python' + '3'

String repetition (*)

If 1 operand is string & another is int then it work as String repetition operator order is not important

2. Relation operator (< ,<=,>,>=)

10<20 ----->True
10<=20 ----->True
10>20 ----->False
10>=20 ----->False

nesting is also applicable (if all conditions true then result is true)
10<20<30<40 ----------> True
10<20>30<40 ----------> False

3. Eqaulity operator (==,!=)

It is used for content comparision

ie. a=20 c=500
b=50 d=14664
a==b c==d ---------->False
a!=b c!=d ---------->True

4. Special operators

identity operator (is , is not)
membership operator (in, not in)

Identity operator

It is used for address comparision

ie. a=50
b='Koirala'

id(a) -------->154415454
id(b) -------->603564235

a is b ------False
a is not b ---------->True

```
a=20
b=50
print(a!=b)
print(a is b)
print(id(a))
print(id(b))

a=257
b=257
print(a==b)
```

```
print(a is b)
print(id(a))
print(id(b))
```

(* imp)
Q- Difference between (is) operator and (==) operator ?

Ans- 'is' operator is used for address comparision while '==' is used for content comparasion

Note-

Reusability of is operator

1. In the case of int range is 0 to 256
2. In the case of bool always
3. In the case of str always
4. In the case of float (always create new object)
5. In the case of complex (always create new object)

```
a=10
b=10
print(a==b)
print(a is b)
print(id(a))
print(id(b))
```

 # Output

```
True
True
93962259520320
93962259520320
```

```
a=1.57
b=1.57
print(a==b)
print(a is b)
print(id(a))
```

```
print(id(b))

# Output
True
False
139977416780032
139977416780176

a=True
b=True
print(a==b)
print(a is b)
print(id(a))
print(id(b))

# Output
True
True
93962259520320
93962259520320

a='Python'
b='Python'
print(a==b)
print(a is b)
print(id(a))
print(id(b))

# Output
True
True
93962259520320
93962259520320
```

membership operator (in, not in)

s='Manisha Yadav Diksha Sharma'

ie. 'Yadav' in s ---------->True
    'Yadav' not in s ---------->False

```
Pytagline='We can develop any type of applicatio   by usig Python'
print('Python' in Pytagline)
print( 'Python' not in Pytagline)
```

 # Output

```
True
False
```

5. Logical (and, or, not)

and

if all conditions are True then result is True

10>20 and 10<30

ie. True and False----------> False
     True and True-----------> True

or

if atleast one conditions are True then result is True

ie. True and False----------> True
     True and True-----------> True

not

if conditions is True then result is False

ie. not True ----------> False
     not False----------> True

6. Bitwise Operators (& ,| , ~ , <<, >>)

Only applicable for int and binary

& ----------> if both bits are True then result is True
| ----------> if atleast one bit is True then result is True

^ ----------> if both bits are differ then result is True
~ ----------> if bit is True then result is False

ie. 1 & 0 ---------->0
1 & 1 ---------->1

1 | 0 ---------->1
1 | 1 ---------->1
0 | 0 ---------->0

1 ^ 0 ----------> 1
1 & 1 ----------> 0

4.5 & 5 ---------->(valid)

imp----------
~4 ----------> -5

(imp) Note-

+ve number directly stored in memory but -ve number is stored in 2's compliment form

shift operator(<< >>)
10<<2 assume 32 bit-- 00000000000000000000---001010 right hand vacant bit filled by zero

10>>2 assume 32 bit-- 00000000000000000000---001010 left hand vacant bit filled by sign bit

7. Assignment Operator (=)

ie. a=50
b=True

a,b,c,d=10,20,30,True

compound Assignment

ie. a+=10 ---------- a=a+10
a*=5 ----------a=a*5
a/=6 ---------- a=a/6

Operator precedence

()
**
~,
*, /, %, //
+, -
<<, >>,
&
^
|
<, >= <, <=, == ,!=
=,+=,
is , is not,
in , not ins
not
and
or



(VII) Input/Output

Reading input from the keyboard
In python 2 the following 2 functions are available
1. raw_input
2 input
but in python 3 we have only input ()

Python2

raw-input()

This function always reads the data from the keyboard in the form of String Format. We have to convert that string type to our required type by using the corresponding type casting methods.

input()

input() function can be used to read data directly in our required format.We are not required to perform type casting.

Python3

input()

It will take input in str form so we have to require type casting.

int()

 # wap to input two no print their sum

```
number1=input("Enter first number:")
number2=input("Enter 2nd  number:")
a=int(number1)
b=int(number2)
print('The sum:',a+b)
```

 # Output

```
Enter first number:10
Enter 2nd  number:20
The sum: 30
```

float()

```
number1=float(input("Enter first number:"))
number2=float(input("Enter 2nd  number:"))
print('The sum:',number1+number2)
```

 # Output

```
Enter first number:10
Enter 2nd  number:58.65
The sum: 68.65
```

eval()

eval() can evaluate the Input to list, tuple, set, etc based the provided Input.

```
number1=eval(input("Enter first number:"))
number2=eval(input("Enter 2nd  number:"))
print('The sum:',number1+number2)
```

 # Output

```
Enter first number:10
Enter 2nd  number:3+45j
The sum: (13+45j)
```

Q. Write a program to read Employee data from the keyboard and print that data.

(imp) How to read multiple values from the keyboard in a single line

```
a,b=[int (x) for x in input("Enter number:").split()]
print(a+b)
```

(VIII) Command Line Arguments

The argument which are passes at the time of execution are called command line arguments
* argv hold command line arguments in python
* argv is list type
* argv present in sys module

ie c:\Er.sandy\Python_class python cmd.py 100,200, 300
| | | |
command line arguments

Check type of argv

```
from sys import argv
print(type(argv))
```

Wap to display cmd line arguments

Output Statements

print()

We can use print() function to display output.

print(str)

```
print('Maneesha Yadav')


print('Hello Python World')
```

print() with var-arg

```
print('Maneesha Yadav',10,True,10.25)

number1=eval(input("Enter first number:"))
number2=eval(input("Enter 2nd  number:"))
sum=number1+number2
print('The sum:of given numbers', number1, number2, 'is',sum, 3+3j)
```

print() without argument line seprator

print() with end attribute

```
print('Maneesha Yadav',10,True,10.25, end=' ')
print("Diksha")
```

print() with any object

```
a,b=50,60
l=[1,2,5,True,'Abha']
s='Python'
print(a)
print(b)
print(l)
print(s)
```

print() with repacement

```
name1='Diksha'
name2='Manisha'
name3='Abha'
```

print('{0} {1} {2} are studying at Techavera'.format(name1,name2,name3))

print() formatted string

%i ---------- int
%d ---------- int
%f ---------- float
%s ----------str

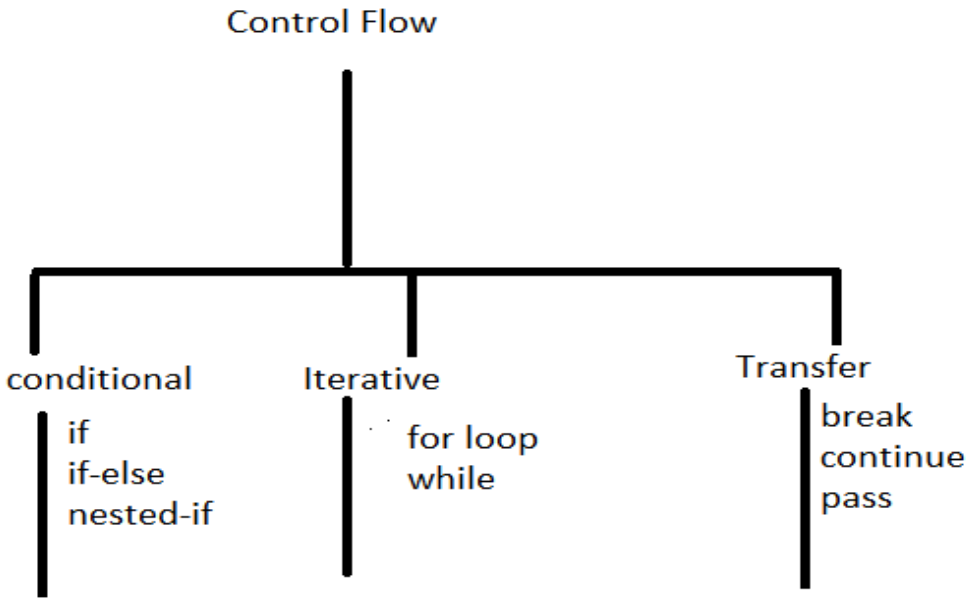
ie
a,b,c=1,2,3
print('a value is %i'%a)
print('b vales is %d'%b)
print('c vales is %d'%c)

 # Output

```
a value is 1
b vales is 2
c vales is 3
```

(IX) Control Flow

It describe the order of program.
It describe in which order statements are executed at runtime



Conditional Statement

if
syntax

```
 if condition:
        statement
        statement
        statement
        statement
        .
        .
        statement
```

if condition is true then if block statements will execute.

ie.

 # wap to input a number & check number is 'negative or not

```
a=int(input('Enter a number: '))
if a<0:
    print(a,'Number is negative')
```

 # Output
Enter a number: -10
-10 Number is negative

if-else

syntax

```
if condtion:
    statement
        statement
        statement
        statement
        .
        .
        statement
else:
    statement
        statement
        statement
        statement
        .
        .
        statement
```

 # wap to input a number & check number is even or odd

```
a=int(input('Enter a number: '))
if a%2==0:
    print(a,'is Even')
else:
    print(a,'is Odd')
```

 # Output
Enter a number: 10
10 is Even
Rerun
Enter a number: 15
15 is Odd

 # wap to input two no & print greater

```
number1 = int(input("Enter first number: "))
number2 = int(input("Enter second number: "))
if (number1 > number2):
   print(number1, " is greater than ", number2)
else:
   print(number2, " is greater than ", number1)
```

 # Output
Enter first number: 10
Enter second number: 20
20 is greater than 10

if-elif-else (nested if)

syntax

if condition:
    statement
    statement
    statement
    statement
    .
    .
    statement
elif condtion:
    statement
    statement
    statement
    statement
    .
    .
    statement
else:
    statement
    statement
    statement
    statement
    .
    .
    statement

 # wap to input two no & print lesser

```
number1 = int(input("Enter first number: "))
number2 = int(input("Enter second number: "))
if number1==number2:
   print(number1, " and ", number2,'are equals')
elif (number1 < number2):
```

```
    print(number1, " is lesser than ", number2)
else:
    print(number2, " is lessar than ", number1)
```

 # Output
Enter first number: 10
Enter second number: 10
10 and 10 are equals
Rerun
Enter first number: 10
Enter second number: 20
10 is lesser than 20

wap to input two no & print greater

```
number1 = int(input("Enter first number: "))
number2 = int(input("Enter second number: "))
if (number1 == number2):
    print(number1, " is equal to ", number2)
elif (number1 > number2):
    print(number1, " is larger than ", number2)
else:
    print(number2, " is larger than ", number1)
```

 # Output
Enter first number: 10
Enter second number: 10
10 is equal to 10
Rerun
Enter first number: 50
Enter second number: 60
60 is larger than 50

wap to input three no & print greater

```
a=int(input('Enter a number: '))
b=int(input('Enter a number: '))
c=int(input('Enter a number: '))
if (a>b and a>c):
    print(a, 'is greater number')
elif b>c:
    print(b,'is greater number')
```

```
else:
    print(c,'is greater number')
```

 # Output
Enter a number: 10
Enter a number: 20
Enter a number: 30
30 is greater number

wap to input three no & print greater

```
a=int(input('Enter a number: '))
b=int(input('Enter a number: '))
c=int(input('Enter a number: '))
if a==b==c:
    print('All are equals')
elif (a>b and a>c):
    print(a, 'is greater number')
elif b>c:
    print(b,'is greater number')
else:
    print(c,'is greater number')
```

 # Output
```
Enter a number: 10
Enter a number: 10
Enter a number: 10
All are equals
```

wap to input three no & print greater (complete logic)

```
a=int(input('Enter a number: '))
b=int(input('Enter a number: '))
c=int(input('Enter a number: '))
if a==b==c:
    print('All are equals')
elif (a>b and a>c):
    print(a, 'is greater number')
elif b==c:
    print(b,'and',c, 'are equal')
```

```
elif b>c:
    print(b,'is greater number')
else:
    print(c,'is greater number')
```

 # Output

```
Enter a number: 10
Enter a number: 10
Enter a number: 10
All are equals
```

Rerun

```
Enter a number: 10
Enter a number: 20
Enter a number: 20
20 and 20 are equal
```

Rerun

```
Enter a number: 10
Enter a number: 20
Enter a number: 5
20 is greater number
```

Ternary Operator in Python

Variable1= var2 if condition else var3

```
# wap to input two no & print greater by using ternary operator
a=int(input('Enter a number: '))
b=int(input('Enter a number: '))
c=a if a>b else b
print('Greater number is',c)
```

 # Output

```
Enter a number: 10
Enter a number: 20
Greater number is 20
```

```
# wap to input two no & print greater by using ternary operator

a=int(input('Enter a number: '))
b=int(input('Enter a number: '))
```

```
c=int(input('Enter a number: '))
if a==b==c:
   print('All are equals')
d=a if (a>b and a>c) else b if b<c else c
if a==d:
   pass
else:
   print('Greater number is',c)
```

 # wap to input three no & print smaller no by using ternary operator

```
a=int(input('Enter a number: '))
b=int(input('Enter a number: '))
c=int(input('Enter a number: '))
minno = a if  (a<b and a<c) else b if b<c else c
print('smaller no is ', minno)
```

 # Output

```
Enter a number: 10
Enter a number: 20
Enter a number: 5
smaller no is  5
```

 # wap to input Year & check year is leap or not

```
year=int(input('Enter any Year:'))
if (year%4==0 and year%100!=0) or (year%400==0):
        print(year,'Year is leap')
else:
            print(year,'Year is not leap')
```

 # Output

```
Enter any Year:2016
2016 Year is leap
```

Rerun

```
Enter any Year:100
100 Year is not leap
```

Rerun

```
Enter any Year:2000
2000 Year is leap
```

if-elif

syntax

```
if condition:
        statement
         statement
         statement
         statement
         .
         .
         statement
elif condition:
        statement
         statement
         statement
         statement
         .
         .
         statement
```

 # wap to input two no & print greater

```
number1 = int(input("Enter first number: "))
number2 = int(input("Enter second number: "))
if (number1> number2):
   print(number1, " is larger than ", number2)
elif (number1 <number2):
   print(number2, " is larger than ", number1)
```

 # Output

```
Enter first number: 50
Enter second number: 60
60  is larger than  50
```

 # W.A.P. to input 5 subject marks and print grade

```
sub1=int(input("Enter marks of the first subject: "))
sub2=int(input("Enter marks of the second subject: "))
sub3=int(input("Enter marks of the third subject: "))
sub4=int(input("Enter marks of the fourth subject: "))
sub5=int(input("Enter marks of the fifth subject: "))
avg=(sub1+sub2+sub3+sub4+sub4)/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: F")
```

 # Output
Enter marks of the first subject: 80
Enter marks of the second subject: 85
Enter marks of the third subject: 90
Enter marks of the fourth subject: 91
Enter marks of the fifth subject: 92
Grade: B

Note-

Both elif and else are optional
Multiple elif blocks are allowed.

Itrative/looping

If we want to execute a group of statements multiple times then we should go for iterative statements.
Python have two type of iterative statements

for loop
while loop

for (most enhanced loop)

If we want to execute some action for every elements present in some sequence then we should use for loop.

syntax

for var in sequence:
statement
statement
statement
statement
.
.
statement

i
e.

```
for a in range(20):
    print(a,end=' ')
```

 # Output

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
```

```
# wap to print 20 to 1
for a in range(20,0,-1):
    print(a,end=' ')
```

 # Output

```
20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
```

 # W.A.P to input a number and print their table

```
print("Please enter a number to print the table of : ")
number=input()
n = int(number)
for i in range(1,11):
    print(number," X ",i," = ",i*n)
```

 # Output

```
Please enter a number to print the table of :
5
5  X  1  =  5
5  X  2  =  10
5  X  3  =  15
5  X  4  =  20
5  X  5  =  25
5  X  6  =  30
5  X  7  =  35
5  X  8  =  40
5  X  9  =  45
5  X  10  =  50
```

print characters present in the sequence.

```
s='Shreya Khirwal'
for x in s:
     print(x)
```

 # output

S
h
r
e
y
a

K
h

i
r
w
a
l

```
s='Shreya Khirwal'
for x in s:
    print(x,end='')
```

 # Output
Shreya Khirwal

while loop

if we want to execute a group of statements iteratively until some condition false,then we should use while loop

syntax

```
while condition:
        statement
         statement
         statement
         statement
         .
         .
         statement
```

ie.

W.A.P. to print 100 to 1 by using while loop.

```
i=1
while i<=100:
      print(i,end=' ')
      i+=1
```

 # Output

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57
58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75
76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93
94 95 96 97 98 99 100
```

 # Wap to input a number & print their digits sum.

```
n=int(input("Enter a number:"))
tot=0
while(n>0):   #187>0 18>0 1>0  0>0
    dig=n%10  #7
    tot=tot+dig # 0+7+8+1
    n=n//10   #187//10  18//10  1//10
print("The total sum of digits are:",tot)
```

 # Output

```
Enter a number:187
The total sum of digits are: 16
```

 # Wap to input a number & print their digits product.

```
n=int(input("Enter a number:"))
tot=1
while(n>0):   #187>0 18>0 1>0  0>0
    dig=n%10  #7
    tot=tot*dig # 0+7+8+1
    n=n//10   #187//10  18//10  1//10
print("The total Product of digits are:",tot)
```

 # Output

```
Enter a number:187
The total Product of digits are: 56
```

 # Wap to input a number & print their digits sum at each stage.

```
n=int(input('Enter number '))
t=0
```

```
while n>0:
    rem=n%10
    t+=rem
    n=n//10
    print('The sum of given number at each stage',t)
print('The sum of given number is',t)
```

 # Output

```
Enter number 187
The sum of given number at each stage 7
The sum of given number at each stage 15
The sum of given number at each stage 16
The sum of given number is 16
```

 # WAP to input a number & print number in Reverse order

```
n=int(input("Enter number: "))
rev=0
while(n>0):
    dig=n%10
    rev=rev*10+dig
    n=n//10
print('Reverse Number of given Number is',rev)
```

 # Output

```
Enter number: 154
Reverse Number of given Number is 451
```

 # WAP to input a number & print number in Reverse order

```
n=int(input("Enter a number:"))  #125
rev=0
while(n>0):  #125>0 12>0  1>0   0>0 (False)
    dig=n%10  #125%10-->5  12%10  1%10
    rev=rev*10+dig # 0+5  5*10+2   52*10+1
    n=n//10   # 125//10   12//10   1//10
print("Reverse number is:",rev)
```

#Output

```
Enter a number:143
Reverse number is: 341
```

 # wap to input a number and check a number is palindrome or not.

```
n=int(input("Enter number: "))
n1=n
rev=0
while(n>0):
    dig=n%10
    rev=rev*10+dig
    n=n//10
if n1==rev:
    print(n1,'is Palindrome' )
else:
    print(n1,'is not Palindrome')
```

 # Output

```
Enter number: 121
121 is Palindrome
```

Rerun

```
Enter number: 143
143 is not Palindrome
```

 # Wap to input a number & check number is Armstrong or not

```
n=int(input("Enter number: "))
t=0
n1=n
while(n>0):
    rem=n%10
    t=rem**3+t
    n=n//10
if rem==n1:
    print(n1,'is Armstrong Number')
else:
```

```
    print(n1,'is Armstrong Number')
```

 # Output

```
Enter number: 153
153 is Armstrong Number
```

Rerun

```
Enter number: 143
143 is Armstrong Number
```

 # wap to input a number & number is prime or not

```
a=int(input("Enter number: "))
k=0
for i in range(2,a//2+1):
    if(a%i==0):
        k=k+1
if(k<=0):
    print(a,"Number is prime")
else:
    print(a,"Number isn't prime")
```

 # Output

```
Enter number: 23
23 Number is prime
```

Rerun

```
Enter number: 20
20 Number isn't prime
```

infinite loop

```
i=0
while True:
     i+=1
     print("Hello I am Your Python Trainer at Techavera Solutions Pvt. Ltd")
```

 # Output

```
Hello I am Your Python Trainer at Techavera Solutions Pvt. Ltd
```

Hello I am Your Python Trainer at Techavera Solutions Pvt. Ltd
Hello I am Your Python Trainer at Techavera Solutions Pvt. Ltd
Hello I am Your Python Trainer at Techavera Solutions Pvt. Ltd
Hello I am Your Python Trainer at Techavera Solutions Pvt. Ltd
Hello I am Your Python Trainer at Techavera Solutions Pvt. Ltd
Hello I am Your Python Trainer at Techavera Solutions Pvt. Ltd
Hello I am Your Python Trainer at Techavera Solutions Pvt. Ltd
Hello I am Your Python Trainer at Techavera Solutions Pvt. Ltd
Hello I am Your Python Trainer at Techavera Solutions Pvt. Ltd
Hello I am Your Python Trainer at Techavera Solutions Pvt. Ltd
Hello I am Your Python Trainer at Techavera Solutions Pvt. Ltd
Hello I am Your Python Trainer at Techavera Solutions Pvt. Ltd

.
.
.
.

Nested Loop

Loop inside loop is called nested loop.

Wap to print following pattern

```
1             2             3          4

$             1           1            1
$ $           1 2         22           2 3
$ $ $         1 2 3       333          4 5 6
$ $ $ $       1 2 3 4     4444         7 8 9 10
$ $ $ $ $     1 2 3 4 5   55555 11 12 13
```

#1

```
row=int(input("enter number of rows:"))
for i in range(1,row+1):
    for j in range(1,i+1):
        print('$',end=' ')
    print()
```

#Output

```
enter number of rows:7
$
$ $
$ $ $
$ $ $ $
$ $ $ $ $
$ $ $ $ $ $
$ $ $ $ $ $ $
```

2

```
row=int(input("enter number of rows:"))
for i in range(1,row+1):
    for j in range(1,i+1):
        print(j,end=' ')
    print()
```

#Output

```
enter number of rows:7
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
1 2 3 4 5 6 7
```

3

```
row=int(input("enter number of rows:"))
for i in range(1,row+1):
    for j in range(1,i+1):
        print(i,end=' ')
    print()
```

#Output

```
enter number of rows:7
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
6 6 6 6 6 6
7 7 7 7 7 7 7
```

4

```
row=int(input("enter number of rows:"))
a=1
for i in range(1,row+1):
    for j in range(1,i+1):
        print(a,end=' ')
        a+=1
    print()
```

#Output

```
enter number of rows:7
```

```
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
16 17 18 19 20 21
22 23 24 25 26 27 28
```

 # 5 print following pattern

```
$ $ $ $ $
$ $ $ $
$ $ $
$ $
$
```

```
row=int(input("enter number of rows:"))
for x in range(row,0,-1):
    for y in range(0,x):
        print('$',end=' ')
    print()
```

#Output

```
enter number of rows:7
$ $ $ $ $ $ $
$ $ $ $ $ $
$ $ $ $ $
$ $ $ $
$ $ $
$ $
$
```

 # 6 print following pattern

```
          $
        $ $
      $ $ $
    $ $ $ $
```

$ $ $ $ $

 # Function to demonstrate printing pattern

```
def pypart2(n):

    # number of spaces
    k = 2*n - 2
    # outer loop to handle number of rows
    for i in range(0, n):
        # inner loop to handle number spaces
        # values changing acc. to requirement
        for j in range(0, k):
            print(end=" ")

        # decrementing k after each loop
        k = k - 2

        # inner loop to handle number of columns
        # values changing acc. to outer loop
        for j in range(0, i+1):

            # printing stars
            print('$ ', end="")

        # ending line after each row
        print("\r")

# Driver Code
n = 5
pypart2(n)
```

#Output

```
        $
      $ $
    $ $ $
  $ $ $ $
$ $ $ $ $
```

 # print following pattern

```
   $
  $ $
 $ $ $
$ $ $ $
$ $ $ $ $
```

```
n=int(input('enter rows:'))
for i in range(0,n):
    for j in range(0,n-i-1):
        print(end=' ')
    for j in range(0,i+1):
        print('$',end=' ')
    print()
```

#Output

```
enter rows:7
      $
     $ $
    $ $ $
   $ $ $ $
  $ $ $ $ $
 $ $ $ $ $ $
$ $ $ $ $ $ $
```

```
   $
  $$
 $$$
$$$$
$$$$$
```

```
n=int(input('enter rows:'))
for i in range(0,n):
    for j in range(0,n-i-1):
        print(end=' ')
    for j in range(0,i+1):
        print('$',end='')
    print()
```

#Output

```
enter rows:7
      $
     $$
    $$$
   $$$$
  $$$$$
 $$$$$$
$$$$$$$
```

Transfer statements

break

It used to break statement inside loops to break loop execution based on some condition.

i.e.

```
for r in range(10):
    if r==7:
        break
    print(r)
print('outside of loop')
```

#Output

```
0
1
2
3
4
5
6
```

```
outside of loop
```

continue

It is used to skip current iteration and continue next iteration.

#W.A.P to print even numbers from 1 to 10 by using continue statement

```
for r in range(10):
    if r%2==0:
        continue
    print(r)
print('outside of loop')
```

#Output

```
1
3
5
7
9
outside of loop
```

pass

it is empty statement in python

i.e.

```
def m1():
    pass
print('Sign out')
```

#Output

```
Sign out
```

Q. Difference between None and del Keyword in python

```
x=[1,2,3]
print(x)
x=None
```

```
print(x)
del x
print(x)

#Output
[1, 2, 3]
None
---------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-16-6d068b616fba> in <module>()
      4 print(x)
      5 del x
----> 6 print(x)
      7
      8 #Output

NameError: name 'x' is not defined
```

 # Program to find how many times a digit occured in number

```
#**************having values**************************************************
number=int(input("Please enter number of minimum 2 digits : "))
while number<10:
    number=int(input("Invalid number ! Please enter number again : "))
print("Please enter digit : ",end='')
digit=int(input())
#*****************************************************************************
**************


copy_number=number
count=0
```

```
#************************Caluculations*****************************
************
while number>0:
        if number%10==digit:
                count+=1
        number//=10
#*************************************************************
**************
if count!=0:
        print(digit," is present in ",copy_number," | ",count," times.")
else:
        print(digit," is not present in ",copy_number)
```

 # Output

```
Please enter number of minimum 2 digits : 456
Please enter digit : 5
5  is present in  456  |  1  times.
```

 # Odd even number in given range

```
print("Please enter starting point of series : ")
num1=input()
print("Please enter ending point of series : ")
num2=input()
start=int(num1)
end=int(num2)
if start > end :
   temp = start
   start= end
   end = temp
```

```
print("**********ALL EVEN NUMBERS ARE**********")
for i in range(start,end) :
    if i%2 == 0 :
        print(i,end=' ')
print()
print("**********ALL ODD NUMBERS ARE**********")
for i in range(start,end) :
    if i%2 != 0 :
        print(i,end=' ')
```

 # Output

```
Please enter starting point of series :
5
Please enter ending point of series :
50
**********ALL EVEN NUMBERS
ARE**********
6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40
42 44 46 48
**********ALL ODD NUMBERS
ARE**********
5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41
43 45 47 49
```

 # FINDER OF LARGEST AND SMALLEST NUMBER GIVEN BY THE USER

```
print("**********FINDER OF LARGEST AND SMALLEST NUMBER GIVEN BY THE USER **********")
print("Please enter 1st number : ",end='')
num1=input()
print("Please enter 2nd number : ",end='')
num2=input()
print("Please enter 3rd number : ",end='')
num3=input()
print("Please enter 4th number : ",end='')
num4=input()
```

```
# finding the largest number.

if num1>num2 and num1>num3 and num1>num4 :
        largest=num1
if num2>num1 and num2>num3 and num2>num4 :
        largest=num2
if num3>num1 and num3>num2 and num3>num4 :
        largest=num3
if num4>num1 and num4>num2 and num4>num3 :
        largest=num4

# finding the smallest number.

if num1<num2 and num1<num3 and num1<num4 :
        smallest=num1
if num2<num1 and num2<num3 and num2<num4 :
        smallest=num2
if num3<num1 and num3<num2 and num3<num4 :
        smallest=num3
if num4<num1 and num4<num2 and num4<num3 :
        smallest=num4



print()
print("Largest number from ",num1,",",num2,",",num3,",",num4," is : ",largest)

print()
print("Smallest number from ",num1,",",num2,",",num3,",",num4," is :
",smallest)
```

 # Output

```
**************FINDER OF LARGEST AND SMALLEST NUMBER
GIVEN BY THE USER **********************
Please enter 1st number : 50
Please enter 2nd number : 60
Please enter 3rd number : 41
Please enter 4th number : 50

Largest number from  50 , 60 , 41 , 50  is :  60
```

Smallest number from 50 , 60 , 41 , 50 is : 41

#

Small Program on Student management System

```
print('**********************************************************************')
print("----------STUDENT ACADEMIC MANAGEMENT SYSTEM----------")
print('**********************************************************************')
print("Please enter enrollment number of student : ")
student_enrollment=input()
print("Please enter name of the student : ")
student_name=input()
print("Please enter the marks of HINDI : ")
hindi=input()
print("Please enter the marks of ENGLISH : ")
english=input()
print("Please enter the markts of MATH : ")
math=input()
print("Please enter the marks of SCIENCE : ")
science=input()
print("Please enter the marks of HISTORY : ")
history=input()
print('**********************************************************************')
print("----------RESULT OF ",student_name.upper(),"----------")
print('**********************************************************************')

total = int(hindi) + int(english) + int(math) + int(science) + int(history)
#int(total,3)
```

```
percentages=(total*100)/500
#float(percentages)
print("Total marks are : ",total)
print("Percentages : ",percentages)
if percentages > 33 :
    print(student_name.upper()," is Passed ")
else :
    print(student_name.upper()," is Failed ")


#Output
**************************************************************
*********
--------------STUDENT ACADEMIC MANAGEMENT
SYSTEM------------------
**************************************************************
*********
Please enter enrollment number of student :
101
Please enter name of the student :
Tushar
Please enter the marks of HINDI :
95
Please enter the marks of ENGLISH :
90
Please enter the markts of MATH :
98
Please enter the marks of SCIENCE :
98
Please enter the marks of HISTORY :
80
**************************************************************
*********
------------------RESULT OF  TUSHAR --------------
**************************************************************
*********
Total marks are :  461
Percentages :  92.2
TUSHAR  is Passed
```

Small Program on SIMPLE INTEREST CALCULATION SYSTEM

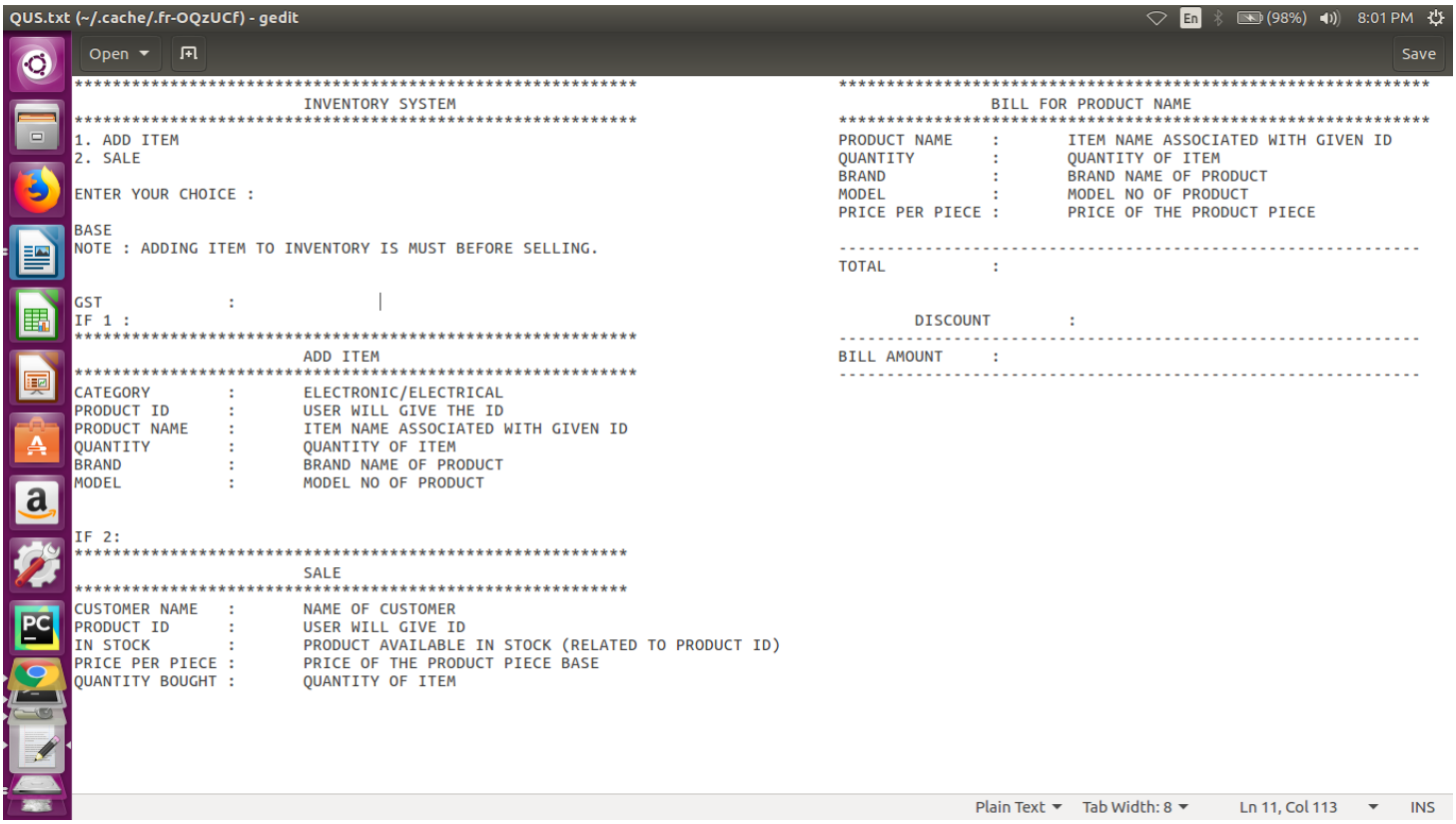
```
print("**********SIMPLE INTEREST CALCULATION SYSTEM********")
si=0
found=False
print("PLEASE ENTER THE LOAN AMOUNT : ",end='')
amount=int(input())
print("PLEASE ENTRE THE ANNUAL RATE : ",end='')
rate=float(input())
print("PLEASE ENTER THE TIME PERIOD IN YEARS : ",end='')
time=int(input())
print()
print("***************CALCULATION METHOD*****************")
print("HOW DO YOU WANT TO CALCULATE THE SI ?")
print("""\n\t 1.YEARLY\n
        2.HALF YEARLY\n
        3.QUATERLY\n
        4.MONTHLY\n""")
print("YOUR CHOICE : ",end='')
choice=int(input())
if choice != 1 and choice != 2 and choice !=3 and choice != 4 :
   print("WRONG INPUT !")
else:
   found=True
   if choice == 1:
      si=(amount*rate*time)/100
   if choice == 2:
      si=((amount)*(rate/2)*(time/2))/100
   if choice == 3:
      si=((amount)*(rate/4)*(time/4))/100
   if choice == 4:
      si=((amount)*(rate/12)*(time/12))/100
```

```
if found == True:
    print("**************************************************")
    print()
    print("SIMPLE INTEREST IS : ",si)
```

Output

```
**********SIMPLE INTEREST CALCULATION SYSTEM********
PLEASE ENTER THE LOAN AMOUNT : 65000
PLEASE ENTRE THE ANNUAL RATE : 13
PLEASE ENTER THE TIME PERIOD IN YEARS : 1

**************CALCULATION METHOD*****************
HOW DO YOU WANT TO CALCULATE THE SI ?

         1.YEARLY

         2.HALF YEARLY

         3.QUATERLY

         4.MONTHLY

YOUR CHOICE : 1
**************************************************

SIMPLE INTEREST IS :  8450.0
```

Inventory System




```
def choice(menu_choice):
        while menu_choice==0 or menu_choice>2:
                menu_choice=int(input("INVALID CHOICE! PLEASE CHOOSE
VALID ONE : "))
        return menu_choice

#**********************************MAIN FUNCTION DEFINITION
AND DECLARATION*****************************
def main():
#-------------------------------ADD
PRODUCT-----------------------------------------
        if menu_choice==1:
                added+=1
                sale_product()

#-------------------------------SALE
PRODUCT-----------------------------------------
        elif menu_choice==2 and added!=0:
                customer_name=input("CUSTOMER NAME \t\t:\t ")
```

```
			product=int(input("ENTER PRODUCT ID \t\t:\t "))
			sale_product(customer_name,product)
		else:
			print("PLEASE ADD ITEM FIRST")

#********************************ADD PRODUCT FUNCTION
DEFINITION AND DECLARATION***************************
def add_product(add):

print("**********************************************************")
		print("\t\tADD PRODUCT TO INVENTORY")

print("**********************************************************\n
")
		to_be_added=int(input("HOW MANY PRODUCTS YOU WANT TO
ADD : "))
		items=dict()
		for x in range(1,to_be_added+1,+1):
			product_id=int(input("PRODUCT ID\t\t:\t"))
			product_name=input("PRODUCT NAME\t\t:\t")
			product_brand=input("PRODUCT BRAND\t\t:\t")
			product_model=product_brand+"-"+input("PRODUCT
MODEL\t\t:\t")
			in_stock=int(input("ENTER QUANTITY\t\t:\t"))
			print("Product have been added successfully...!")
			item={'PRODUCT_ID ':product_id,'PRODUCT_NAME
':product_name,'PRODUCT_BRAND ':product_brand,'PRODUCT_MODEL ':
product_model,'AVAILABLE ':in_stock}
			items.update({x:item})
			add+=1
		print(items)
		return add
#********************************SALE PRODUCT FUNCTION
DEFINITION AND DECLARATION***************************
def sale_product():
		print("Executed")
#^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ -- MAIN --
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
def main_menu():
		added=0;

print("**********************************************************")
```

```
        print("\t\tINVENTORY SYSTEM")

print("**************************************************************")
        print("1. ADD PRODUCT")
        print("2. SALE PRODUCT\n")
        menu_choice=int(input("PLEASE ENTER YOUR CHOICE : "))
        menu_choice=choice(menu_choice)
        if menu_choice==1:
                added=add_product(added)
                main_menu()
        elif menu_choice==2 and added!=0:
                print("EXECUTED")
        else:
                print("NOT PRODUCT AVAILABLE PLEASE ADD FIRST")
                main_menu()
main_menu()

# Output
```



(X) String Handling

String

str is is used to represent String
String is collection of characters enclosed with either single quotes (') or in double quotes ("")
ie.

```
abc='Techavera Solutions Pvt Ltd'
type(abc)
<class 'str'>
abc="Techavera Solutions Pvt Ltd"
type(abc)
<class 'str'>
```

Multiline Strings

we can represent multiline string either triple single quotes (''' Techavera Python Trainer ''') or in triple double quotes (""" Techavera Python Trainer""")

i.e

```
abc='''Welcome To
        techavera
                Solutions
                        Pvt. Ltd.'''
type(abc)
#<class 'str'>
print(abc)
# Welcome To
        techavera
                Solutions
                        Pvt. Ltd.
```

```
abc="""Welcome To
        techavera
                Solutions
                        Pvt. Ltd."""
type(abc)
#<class 'str'>
print(abc)
# Welcome To
        techavera
                Solutions
                        Pvt. Ltd.
```

Indexing

In python both +ve and -ve indexes are available

```
s='PYTHON'
print(s[0])
print(s[-1])
print(s[-3])
print(s[2])
#print(s[110]) error
```

ie <<<< .....--------3-2-1

```
s= 'Sandeep kumar sharma Techavera solutions Pvt Ltd Python Trainer'
   0123456789.....................................................>>>>
print(s[4])   #---->e
print(s[-3])  # ---->n
```

slicing

syntax

end=end-1

|

var[begin : end : step]

| | |

All are optional
In slicing we won't get any error

```
s='Learning Python is very easy'
```

```
print(s[9:15:1])
print(s[::])
print(s[0:28:1])
print(s[100:20:2])
```

s= 'Sandeep kumar sharma Techavera solutions Pvt Ltd Python Trainer'
0123456789.....>>>>

```
print(s[::])   ------------->Please fill output
print(s[5::])  ------------->Please fill output
print(s[5:15:])  ------------->Please fill output
print(s[5:15:2])  ------------->Please fill output
```

Note

* if step is +ve then we should move in forword direction (Left to Right)
* if step is -ve then we should move in backward direction (Right to Left)
* if step is +ve then we should move in forword direction (begin to end)
* if step is -ve then we should move in backward direction (begin to end)

In step is +ve
    default value for begin -------->0
    default value for end -------->length of string

In step is -ve
    default value for begin -------->-1
    default value for end -------->(length of string+1)

```
s='Learning Python is very easy'
print(s[::])
print(s[::1])
print(s[::-1])
print(s[-1:-29:-1])
```

W.A.P to input a string & check string is palindrome or not

```
s=input('Enter string: ')
a=s
b=s[::-1]
print(b)
if b==a:
    print(s,'is palindrome')
else:
    print(s,'is not palindrome')
```

 # Output

```
Enter string: NitiN
NitiN
NitiN is palindrome
```

Rerun

```
Enter string: Sandeep
peednaS
Sandeep is not palindrome
```

Access of string

1. By using indexing

```
s='PYTHON'
print(s[4])
print(s[-4])
```

 # Output

```
O
T
```

2. By using slicing

```
s='Manisha Yadav'
s[2:7:]
```

 # Output

```
'nisha'
```

```
l='Learning Python is very Easy'
print(l[9:15:1])
print(l[-1:-10:-1])
```

 # Output

```
Python
ysaE yrev
```

3. By using loops

```
pytag_line='We Can Develop Any Type Of Application By using Python '
a=len(pytag_line)
i=0
while i<a:
    print(pytag_line[i],end='')
    i=i+1
```

 # Output

```
We Can Develop Any Type Of Application By using Python
```

```
s='Learning Python is very Easy'
for x in s:
    print(x, end='')
```

 # Output

```
Learning Python is very Easy
```

```
s="Learning Python is very easy !!!"
n=len(s)
i=0
print("Forward direction")
while i<n:
    print(s[i],end=' ')
    i +=1
print()
print("Backward direction")
i=-1
while i>=-n:
```

```
    print(s[i],end=' ')
    i=i-1
```

 # Output

```
Forward direction
L e a r n i n g   P y t h o n   i s   v e r y   e a s y   ! ! !
Backward direction
! ! !   y s a e   y r e v   s i   n o h t y P   g n i n r a e L
```

4. By using reference variable

```
s="Welcome To techavera \n Solutions Pvt. Ltd. Noida"
print(s)
```

 # Output

```
Welcome To techavera
 Solutions Pvt. Ltd. Noida
```

```
s='We can develop Any type of application by using Python'
print(s)
```

 # Output

```
We can develop Any type of application by using Python
```

Method of String

1. lower() no-arg return string characters into lower case
2. upper() no-arg return string characters into uper case

```
s= 'Sandeep kumar Sharma Techavera Solution Pvt Ltd.'
print(s.upper())
print(s.lower())
print(s.capitalize())
```

 # Output

```
SANDEEP KUMAR SHARMA TECHAVERA SOLUTION PVT LTD.
sandeep kumar sharma techavera solution pvt ltd.
Sandeep kumar sharma techavera solution pvt ltd.
```

```
s='We can develop Any type of application by using Python'
print(s.upper())
print(s.lower())
print(s.capitalize())
```

 # Output

```
WE CAN DEVELOP ANY TYPE OF APPLICATION BY USING PYTHON
we can develop any type of application by using python
We can develop any type of application by using python
```

3. replace(str,str)

```
s= 'Sandeep kumar sharma'
print(s.replace('sharma','verma'))
```

 # Output

```
Sandeep kumar verma
```

```
s= 'Learning python is very easy'
print(s.replace('easy','hard'))
```

 # Output

```
Learning python is very hard
```

4. split( )

```
s= 'Sandeeep kumar Sharma Techaveera Solution Pvt Ltd.'
a=s.split()
print(a)
print(len(a))
print(type(a))
```

 # Output

```
['Sandeeep', 'kumar', 'Sharma', 'Techaveera',
'Solution', 'Pvt', 'Ltd.']
7
<class 'list'>
```

```
s= 'Sandeep kumar sharma'
print(s.split('a'))
```

 # Output

```
['S', 'ndeep kum', 'r sh', 'rm', '']
```

```
s='We can develop any type of application by using python'
print(s.split())
```

 # Output

```
['We', 'can', 'develop', 'any', 'type', 'of',
'application', 'by', 'using', 'python']
```

```
s='We can develop any type of application by using python'
print(s.split('e'))
```

 # Output

```
['W', ' can d', 'v', 'lop any typ', ' of application
by using python']
```

5. strip() no-arg method

without strip()

```
s=input("""Enter password 'Sandeep':""")
a=s
if a=='Sandeep':
    print("Hello Sandeep Welcome")
else:
    print("Please enter Your Correct Pasword")
```

 # Output

```
Enter password 'Sandeep':    Sandeep
Please enter Your Correct Pasword
```

With strip()

```
s=input("""Enter password 'Sandeep':""")
a=s.strip()
if a=='Sandeep':
   print("Hello Sandeep Welcome")
else:
   print("Please enter Your Correct Pasword")
```

 # Output

```
Enter password 'Sandeep':    Sandeep
Hello Sandeep Welcome
```

6. lstrip()

```
s=input("""Enter 'Sandeep':""")
a=s.lstrip()
if a=='Sandeep':
   print("Hello class GM")
else:
   print("Please enter Sandeep")
```

 # Output

```
Enter 'Sandeep':  Sandeep
Hello class GM
```

7. rstrip()

```
s=input("""Enter password 'Sandeep':""")
a=s.strip()
if a=='Sandeep':
   print("Hello Sandeep Welcome")
else:
   print("Please enter Your Correct Pasword")
```

 # Output

```
Enter password 'Sandeep':    Sandeep
Hello Sandeep Welcome
```

8. capitalize()

```
s='python programming Language'
print(s.capitalize())
```

 # Output
`Python programming language`

9. find(obj)

```
s='python Programming language'
print(s.find('P'))
```

 # Output
7

10. index(obj)

```
s='python Programming language'
print(s.index('P'))
```

 # Output
7

```
s='We can develop any type of application by using python'
print(s.index('e'))
```

 # Output
1

```
s='We can develop any type of application by using python'
print(s.rindex('e'))
```

Output
22

11.count(obj)

```
s='python programming language'
print(s.count('python'))
```

 # Output
1

```
s='We can develop any type of application by using python'
print(s.count('develop'))
```

 # Output
1

```
s='We can develop any type of application by using python'
print(s.count('e'))
```

 # Output
4

Python Program to Count the Number of Vowels in a String

```
var=input("Enter string:")
vowels=0
for x in var:
    if(x=='a' or x=='e' or x=='i' or x=='o' or x=='u' or x=='A' or x=='E' or x=='I' or x=='O' or x=='U'):
        vowels=vowels+1
print("Number of vowels are:")
print(vowels)
```

 # Output

```
Enter string:Sandeep Kumar Sharma
Number of vowels are:
7
```

 # W.A.P. to input a string remove vowel from string

```
str1 = input('Enter any string to remove vowels: ')
newstr = str1;
print('Removing vowels from the given string');
vowels = ('a', 'e', 'i', 'o', 'u');
for x in str1.lower():
    if x in vowels:
        newstr = newstr.replace(x,"");
print("New string after removed all the vowels:");
print(newstr);
```

 # Output

```
Enter any string to remove vowels: Shyam Babu Sharma
Removing vowels from the given string
New string after removed all the vowels:
Shym Bb Shrm
```

 # W.A. Program to reverse order of words.

```
s=input("Enter Some String:")
l=s.split()
print(l)
l1=[]
i=len(l)-1
while i>=0:
    l1.append(l[i])
    i=i-1
output=' '.join(l1)
print(output)
```

 # Output

```
Enter Some String:Shyam babu Sharma
['Shyam', 'babu', 'Sharma']
Sharma babu Shyam
```

Program to reverse internal content of each word.

```
s=input("Enter Some String:")
l=s.split()
l1=[]
i=0
while i<len(l):
    l1.append(l[i][::-1])
    i=i+1
output=' '.join(l1)
print(output)
```

 # Output

```
Enter Some String:Sandeep Kumar Sharma
peednaS ramuK amrahS
```

Write a program to print characters at odd position and even position for the given String?

1.

```
s=input("Enter Some String:")
print("Characters at Even Position:",s[0::2])
print("Characters at Odd Position:",s[1::2])
```

 # Output

```
Enter Some String:Sandeep
Characters at Even Position: Snep
Characters at Odd Position: ade
```

2.

```
s=input("Enter Some String:")
i=0
print("Characters at Even Position:")
while i< len(s):
    print(s[i],end=',')
    i=i+2
print()
print("Characters at Odd Position:")
i=1
```

```
while i< len(s):
    print(s[i],end=',')
    i=i+2
```

 # Output

```
Enter Some String:Sandeep
Characters at Even Position:
S,n,e,p,
Characters at Odd Position:
a,d,e,
```

Program to merge characters of 2 strings into a single string by taking characters alternatively.

```
s1=input("Enter First String:")
s2=input("Enter Second String:")
output=''
i,j=0,0
while i<len(s1) or j<len(s2):
    if i<len(s1):
        output=output+s1[i]
        i+=1
    if j<len(s2):
        output=output+s2[j]
        j+=1
print(output)
```

 # Output

```
Enter First String:Sandeep
Enter Second String:Techavera
STaencdheaevpera
```



(XI) List

List is the colletion of hetrogenious object where insertion order is preserved & duplicates are allowed.
It is mutable & represented by [ ]

We can perform CRUD operation on List Data type

C ---------- Create
R ---------- Read
U ---------- Update
D ---------- Delete

```
list1=[1,2,True,"Python",1.5,3+4j,1,2,1,2]
print(list1)
print(type(list1))
```

 # Output

```
[1, 2, True, 'Python', 1.5, (3+4j), 1, 2, 1, 2]
<class 'list'>
```

Creation of list

1. by using []

```
l=eval(input('Enter List Element:'))
print(l)
```

#Output
Enter List Element:[10,20,21,11,11,2,3]
[10, 20, 21, 11, 11, 2, 3]

```
list1=[1,2,True,"Python",1.5,3+4j,1,2,1,2]
print(list1)
```

#Output
[1,2,True,"Python",1.5,3+4j,1,2,1,2]

2 by using list()

```
i=(10,20,True,'python')
print(type(i))
l=list(i)
print(l)
print(type(l))
```

#Output

```
'''<class 'tuple'>
[10, 20, True, 'python']
<class 'list'>'''
```

```
tup=(1,2,True,"Python",1.5,3+4j,1,2,1,2,'Barish')
print(type(tup))
print(tup)
lis=list(tup)
print(type(lis))
print(lis)
```

 # Output

```
<class 'tuple'>
(1, 2, True, 'Python', 1.5, (3+4j), 1, 2, 1, 2,
'Barish')
<class 'list'>
[1, 2, True, 'Python', 1.5, (3+4j), 1, 2, 1, 2,
'Barish']
```

3. By using list comprehension

List comprehension

Write the logic to generate list Element is called List Comprehension.

```
l= [x*x for x in range(1,10)]
```

```
l= [x*x for x in range(1,10)]
l
```

```
l= [x**x for x in range(1,10)]
print(l)
```

Output

```
[1, 4, 27, 256, 3125, 46656, 823543, 16777216,
387420489]
```

4. By using split()

```
s='Learning Python is Very easy'
x=s.split()
print(x)
```

 # Output

```
['Learning', 'Python', 'is', 'Very', 'easy']
```

Accessing of list

1. By using reference vaiable

```
i=[10,20,True,'python']
print(i)
```

 # Output

```
[10, 20, True, 'python']
```

2. By using indexing

```
i=[10,20,True,'python']
print(i[2])
```

 # Output

```
True
```

```
list1=[1,2,True,"Python",1.5,3+4j,1,2,1,2,'Barish']
print(list1[5])
```

 # Output

3+4j

3. By using slicing

```
i=[10,20,True,'python']
print(i[1:3])
```

 # Output

```
[20, True]
```

```
list1=[1,2,True,"Python",1.5,3+4j,1,2,1,2,'Barish']
print(list1[2:50:2])
```

Output

```
[True, 1.5, 1, 1, 'Barish']
```

[ ]:

```
list1=[1,2,True,"Python",1.5,3+4j,1,2,1,2,'Barish']
print(list1[::-1])
```

 # Output

```
['Barish', 2, 1, 2, 1, (3+4j), 1.5, 'Python', True, 2, 1]
```

4. By using loop

```
i=[10,20,True,'python']
for x in i:
    print(x)
```

 # Output

```
'''10
20
True
python'''
```

```
i=[10,20,True,'python']
x=0
while(x<len(i)):
    print(i[x])
    x+=1
```

 # Output

```
'''10
20
True
python'''
```

#W.A.P. To print even numbers from list:

```
n=[0,1,2,3,4,5,6,7,8,9,10]
for n1 in n:
    if n1%2==0:
        print(n1)
```

```
# Output
0
2
4
6
8
10
```

#W.A.P. To print element of list index wise:

```
l=["A","B","C"]
x=len(l)
for i in range(x):
    print(l[i],"is available at index: ",i,"and at index: ",i-x)
```

 # Output

```
A is available at index:  0 and at index:  -3
B is available at index:  1 and at index:  -2
C is available at index:  2 and at index:  -1
```

Built-in-function of list

max(Object)
min(obj)
len(obj)
list(obj)

```
i=[10,20,1,50,500,-600]
print(max(i))
print(min(i))
print(len(i))
```

 # Output

```
500
-600
6
```

```
j=['A','P','z']
print(max(j))
```

 # Output
z

Method of list

1. append(obj) take exactly one argument

```
i=[10,20,1,50,500,-600]
i.append('Hankit')
print(i)
```

 # Output
[10, 20, 1, 50, 500, -600, 'Python']

2. insert(index,obj) take exactly two argument

```
i=[10,20,1,50,500,-600]
i.insert(2, 'Python')
print(i)
```

 # Output
[10, 20, 'Python', 1, 50, 500, -600]

(* Imp) What is the difference Between append() & insert()

```
l=[1,2,'Python','Ajgar wale gupta',True]
l.append('A')
l.append('Saurabh')
l.insert(4,'Tabish')
print(l)
```

 # Output

```
[1, 2, 'Python', 'Ajgar wale gupta', 'Tabish', True, 'A', 'Saurabh']
```

3. remove(obj) take exactly one argument

```
i=[10,20,1,50,500,-600,1]
i.insert(2, 'Python')
print(i)
i.remove(1)
print(i)
```

 # Output

```
[10, 20, 'Python', 1, 50, 500, -600, 1]
[10, 20, 'Python', 50, 500, -600, 1]
```

4. pop() without argument

```
i=[10,20,1,50,500,-600]
i.pop()
print(i)
```

 # Output

```
[10, 20, 1, 50, 500]
```

pop(index) with argument

```
i=[10,20,1,50,500,-600]
i.pop()
print(i)
i.pop(2)
print(i)
```

 # Output

```
 [10, 20, 1, 50, 500]
[10, 20, 50, 500]
```

(* Imp) Difference b/w remove() & pop() method

```
l=[1,2,'Python','Ajgar wale gupta',True]
l.append('A')
l.append('Saurabh')
l.insert(4,'Tabish')
print(l)
l.remove('A')
print(l)
print(l.pop())
print(l.pop(3))
print(l)
```

 # Output

```
[1, 2, 'Python', 'Ajgar wale gupta', 'Tabish', True, 'A', 'Saurabh']
[1, 2, 'Python', 'Ajgar wale gupta', 'Tabish', True, 'Saurabh']
Saurabh
Ajgar wale gupta
[1, 2, 'Python', 'Tabish', True]
```

5. copy() no-arg method

```
i=[10,20,1,50,500,-600,'Uttkarsh','Prachi']
uttkarsh= i
print(id(uttkarsh))
print(id(i))
prachi= i.copy()   # no arg
print(id(prachi))
print(prachi)
print(i)
```

 # Output

```
140362773960392
140362773960392
140362773906824
[10, 20, 1, 50, 500, -600, 'Uttkarsh', 'Prachi']
[10, 20, 1, 50, 500, -600, 'Uttkarsh', 'Prachi']
```

```
list1=[1,2,'Python','Python wale gupta',True,1,2,34]
sl1=list1
print(id(sl1))
print(id(list1))
print(sl1)
dp1=list1.copy()
list1.append('Tanmay')
print(id(dp1))
print(dp1)
print(sl1)
```

 # Output

```
140362834039816
140362834039816
[1, 2, 'Python', 'Python wale gupta', True, 1, 2, 34]
140362773909064
[1, 2, 'Python', 'Python wale gupta', True, 1, 2, 34]
[1, 2, 'Python', 'Python wale gupta', True, 1, 2, 34, 'Tanmay']
```

6. clear () no arg method

```
i=[10,20,1,50,500,-600,'Uttkarsh','Prachi']
i.clear()
print(i)
```

 # Output

```
[]
```

7. reverse() no-arg

```
i=[10,20,1,50,500,-600,'Uttkarsh','Prachi']
i.reverse()
print(i)
```

 # Output

```
['Prachi', 'Uttkarsh', -600, 500, 50, 1, 20, 10]
```

8. sort() no-arg

```
i=[10,20,1,50,500,-600]
i.sort()
print(i)
```

 # Output

```
[-600, 1, 10, 20, 50, 500]
```

```
i=['A','python']
i.sort()
print(i)
```

 # Output

```
['A', 'python']
```

```
i2=['Ujjwal','Shivesh']
i2.sort()
print(i2)
```

 # Output

```
['Shivesh', 'Ujjwal']
```

9 count(obj)

```
i=[10,20,1,1,1,1,2,2,2,50,500,-600]
print(i.count(1))
```

 # Output

4

10. index(obj)

```
i=[10,20,1,1,1,1,2,2,2,50,500,-600]
print(i.index(500))
```

11. extend(obj)

```
x = [1, 2, 3]
x1=[4,5,6,7]
x.extend(x1)
print (x)
```

 # Output

```
[1, 2, 3, 4, 5, 6, 7]
```

Q. Difference between append() & extend()

Nested List

List Inside List is called nested list.

creation of nested list

```
l =[10,20,30,[True,'Phython']]
```

Accessing of list

By using reference variable

```
l =[10,20,30,[True,'Phython']]
print (l)
```

 # Output

```
[10, 20, 30, [True, 'Phython']]
```

by using indexing

```
l =[10,20,30,[True,'Phython']]
l[3]
```

 # Output

```
[True, 'Phython']
```

```
l =[10,20,30,[True ,[1,2,3]],'Phython']
print(l)
print(l[3][1][0])
```

 # Output

```
[10, 20, 30, [True, [1, 2, 3]], 'Phython']
1
```

by using slicing

```
l =[10,20,30,[True,'Phython']]
print(l[3::2])
```

 # Output

```
[[True, 'Phython']]
```

Tuple

Tuple

Tuple is the colletion of hetrogenious object where insertion order is preserved & duplicates are allowed.
It is immutable & represented by ()

```
tuple1=(1,2,True,"Python",1.5,3+4j,1,2,1,2)
print(tuple1)
print(type(tuple1))
```

 # Output

```
(1, 2, True, 'Python', 1.5, (3+4j), 1, 2, 1, 2)
<class 'tuple'>
```

prove immutable behaviour of tuple

```
tuple1=(1,2,True,"Python",1.5,3+4j,1,2,1,2)
print(tuple1)
tuple1[2]= 'program'
```

```
# Output
(1, 2, True, 'Python', 1.5, (3+4j), 1, 2, 1, 2)
---------------------------------------------------
-------------------------
TypeError
Traceback (most recent call last)
<ipython-input-34-d45483c92703> in <module>()
      1
tuple1=(1,2,True,"Python",1.5,3+4j,1,2,1,2)
      2 print(tuple1)
----> 3 tuple1[2]= 'program'
      4
      5 # Output

TypeError: 'tuple' object does not support item
assignment
```

Cretion of tuple

1. by using ( )

```
tuple1=eval(input('Enter tuple Elements:'))
print(tuple1)

# Output
Enter tuple Elements:(1, 2, True, 'Python', 1.5,
(3+4j), 1, 2, 1, 2)
(1, 2, True, 'Python', 1.5, (3+4j), 1, 2, 1, 2)


tuple1=(1,2,True,"Python",1.5,3+4j,1,2,1,2)
print(tuple1)
# tuple[1]=5  immuatable behaviour

# Output
(1, 2, True, 'Python', 1.5, (3+4j), 1, 2, 1, 2)
```

2 . by using tuple()

```
i=[10,20,True,'python']
print(type(i))
l=tuple(i)
print(l)
print(type(l))
```

 # Output

```
<class 'list'>
(10, 20, True, 'python')
<class 'tuple'>
```

Accessing of tuple

1. By using reference variable

```
i=(10,20,True,'python')
print(i)
```

```
# Output
(10, 20, True, 'python')
```

2. By using loop

```
i=(10,20,True,'python')
for x in i:
    print(x)
```

```
# Output
10
20
True
python
```

While loop

```
i=[10,20,True,'python']
x=0
while x<len(i):
    print(i[x])
    x+=1

# Output
10
20
True
python
```

3. By using indexing

```
i=(10,20,True,'python')
print(i[-1])

# Output
python
```

4. By using slicing

```
i=(10,20,True,'python')
print(i[1:3])

# Output
(20, True)
```

Built-in-function of tuple

max(Obj)
min(obj)
len(obj)
tuple(obj)

```
i=(10,20,1,50,500,-600)
print(max(i))
print(min(i))
print(len(i))
```

```
# Output
500
-600
6
```

method of tuple

count(obj)

```
i=(10,20,True,'python',20,20,30,40,50,True,10,10,10)
print(i.count(10))
```

```
# Output
4
```

index(obj)

```
i=(10,20,True,'python',20,20,30,40,50,True,10,10,10)
print(i.index(True))
```

```
# Output
2
```

(* imp) Difference between list & tuples

Tuple Comprehension

Not supported

Tuple Packing and Unpacking:

Packing

```
a=10
b=20
c=30
d=40
t=a,b,c,d
print(type(t))
print(t)
```

 # Output

```
<class 'tuple'>
(10, 20, 30, 40)
```

Unpacking

```
t=(10,20,30,40)
a,b,c,d=t
```

```
t=(10,20,30,40)
a,b,c,d=t
print(a)
print(b)
```

 # Output

```
10
20
```

Q. Write a program to take a tuple of numbers from the keyboard and print its sum and average?

```
t=eval(input("Enter Tuple of Numbers:"))
l=len(t)
sum=0
for x in t:
    sum=sum+x
print("The Sum=",sum)
print("The Average=",sum/l)
```

 # Output

```
Enter Tuple of Numbers:(10,20,30,40,50,60)
The Sum= 210
The Average= 35.0
```



Set

set

set is the collection of hetrogenois object where insertion order is not preserved and duplicate are not allowed.
It is mutable & represented by { }

Note-
indexing & slicing concept is not applicable for set

(Q) Why indexing & slicing concept is not applicable for sets

```
d={100,'Python','Abhi',101,100,101,104,105,True}
print(d)
# d[5]=10  Indexing not applicable for set
print(d)
```

 # Output

```
{True, 100, 101, 104, 105, 'Python', 'Abhi'}
{True, 100, 101, 104, 105, 'Python', 'Abhi'}
```

Creation of set

1. By using { }

```
d=eval(input('Enter set Elements:'))
print(d)
```

 # Output
d=eval(input('Enter set Elements:'))
print(d)

```
d={100,'Python','Abhi',101,100,101,104,105,True}
print(d)
```

Output
```
{True, 100, 101, 104, 105, 'Python', 'Abhi'}
<class 'set'>
```

Note-
{ & } represent dict not set.

```
b=set()
print(type(b))
print(b)
b.add(10)
print(b)
```

Output
```
<class 'set'>
set()
{10}
```

2. By using set() function

```
a=set()
a.add(10)
print(type(a))
print(a)
```

 # Output

```
<class 'set'>
{10}
```

```
l=[10,20,30,40]
s=set(l)
print(s)
```

 # Output

```
{40, 10, 20, 30}
```

Accessing of set

1. By using reference variable

```
d={100,'Python','Abhi',101,100,101,104,105,True}
print(d)
```

 # Output

```
{True, 100, 101, 104, 105, 'Python', 'Abhi'}
```

2. By using loops

```
d={100,'Python','Abhi',101,100,101,104,105,True}
for x in d:
    print(x)
```

```
# Output
True
100
101
104
105
Python
Abhi
```

Methods of set

1. add(obj) take exactly one argument

```
d={100,'Python','Abhi'}
d.add('Rajkumari')
print(d)
```

 # Output

```
{'Python', 100, 'Rajkumari', 'Abhi'}
```

2. remove(obj) take exactly one argument

```
d={100,'Python','Abhi',100,100}
print(d)
d.remove(100)
print(d)
```

```
# Output
{'Python', 'Abhi'}
```

3. discard(obj) take exactly one argument

```
d={100,'Python','Abhi'}
d.discard(100)
print(d)
```

 # Output

```
{'Python', 100, 'Abhi'}
```

(* imp) Difference between discard & remove method.
Ans-

4. pop() no-arg method

Delete Random Element

```
d={100,'Python','Abhi'}
d.pop()
print(d)
```

 # Output

```
{100, 'Python'}
```

(* Imp) difference between remove() & pop()

Ans-

5. difference(obj)

```
d={100,10,3,2,5,6}
e={100,101,104,105,1,3,4}
print(d.difference(e))  #10,5,2,6
```

 # Output

```
{2, 10, 5, 6}
```

6. intersection(obj1,obj2,obj3.....)

```
d={100,10,3,2,5,6,4}
e={100,101,104,105,1,3,4}
f={1,5,6,8,7,10,4}
g=d.intersection(e,f)
print(g)
```

 # Output

```
{4}
```

7. union(obj1,obj2,obj3.....)

```
d={100,10,3,2,5,6,4}
e={100,101,104,105,1,3,4}
f={1,5,6,8,7,10,4}
g=d.union(e,f)
print(g)
```

 # Output

```
{1, 2, 3, 100, 5, 6, 4, 101, 104, 10, 105, 7, 8}
```

8. clear() no-arg

```
d={100,'Python','Abhi'}
print(d)
d.clear()
print(d)
```

 # Output

```
{'Abhi', 100, 'Python'}
set()
```

9. copy() no-arg

```
d={100,'Python','Abhi'}
print(d)
e=d.copy()
print(e)
```

 # Output

```
{'Abhi', 100, 'Python'}
{'Abhi', 100, 'Python'}
```

10 update(obj)

```
d={100,'Python','Abhi'}
e={1,2,3}
d.update(e)
print(d)
```

 # Output

```
{'Abhi', 1, 2, 100, 3, 'Python'}
```

```
d={100,'Python','Abhi'}
e={1,2,3}
f={10,20,30}
print(d)
d.update(e,f)
print(d)
```

 # Output

```
{'Abhi', 100, 'Python'}
{1, 2, 3, 100, 20, 'Python', 'Abhi', 10, 30}
```

Q. Wap program to remove duplicate element of list

```
l=eval(input("Enter list elements: "))
s=set(l)
print(list(s))
```

```
# Output
Enter list elements: [10,10,10,2,2,3,3,4,5,6.5]
[2, 3, 4, 5, 6.5, 10]
```

```
l=eval(input("Enter list elements: ")
l1=[]
for x in l:
        if x not in l1:
        l1.append(x)
print(l1)
```

Q W.A.P. to print different vowels present in the given word ?

```
w = input('Enter A Word:')
s= set(w)
v= {'a','e','i','o','u'}
d=s.intersection(v)
print(d)
```

 # Output

```
Enter A Word:Sandeep
{'e', 'a'}
```


TECHAVERA
python
Python Training in Noida
Enquiry No: +91-8506-888-288
www.techaveranoida.in

Dictionary

dict

dict is the collection of key, value pairs where insertion order is not preserved and duplicate keys are not allowed while value can be duplicate.

It is represented by {} & it is mutable.
Hetrogenous is also allowed.

Syntax

ref var={key1:value,key2:value,.....keyn:value}

```
d={'Abhi':101,100:101,104:105,3+4j:'Sandeep'}
print(d)
print(type(d))
```

 # Output

```
{'Abhi': 101, 100: 101, 104: 105, (3+4j):
'Sandeep'}
<class 'dict'>
```

some syntaxes in dict

CRUD

reference_var[key]=value

reference_var[key]

del reference_var[key]

del reference_var

```
d={100:'Python','Abhi':101,104:105}
d[106]='soni'
d[100]='Manisha'
d['Anurag']='soni'
print(d)
```

i.e 2

```
d={}
d[106]='soni'
d[100]='soni'
d['Anurag']='vinod'
print(d)
d[100]=100
print(d)

d={100:'Python','Abhi':101,104:105}
d[106]='Vinod'
d[100]='soni'
print(d)
d[100]='Tabish'
del d[104]
print(d)
# Output
{100: 'soni', 'Abhi': 101, 104: 105, 106: 'Vinod'}
{100: 'Tabish', 'Abhi': 101, 106: 'Vinod'}
```

```
d={'Abhi':101,100:101,104:105,3+4j:'Sandeep'}
d['Satyam']='Satyam'
print(d)
print(d[100])
del d['Abhi']
print(d)
del d
print(d)
```

 # output

```
{'Abhi': 101, 100: 101, 104: 105, (3+4j):
'Sandeep', 'Satyam': 'Satyam'}
101
{100: 101, 104: 105, (3+4j): 'Sandeep', 'Satyam':
'Satyam'}
---------------------------------------------------
-------------------------
NameError
Traceback (most recent call last)
<ipython-input-54-b58282aad346> in <module>()
      6 print(d)
      7 del d
----> 8 print(d)
      9
     10 # Output

NameError: name 'd' is not defined
```

Creation of dict

By using { }

```
d=eval(input("Enter Dict Elements:"))
print(d)
print(type(d))
```

 # Output

```
Enter Dict Elements:
{10:20,'Satyam':101,101:102,102:102}
{10: 20, 'Satyam': 101, 101: 102, 102: 102}
<class 'dict'>
```

```
d={'Abhi':101,100:101,104:105,3+4j:'sandeep'}
print(d)
```

 # Output
{'Abhi':101,100:101,104:105,3+4j:'sandeep'}

Accessing of dict

1. By using reference variable

```
d={100:'Python','Abhi':101,100:101,104:105}
print(d)
```

 # Output
{'Abhi':101,100:101,104:105,3+4j:'sandeep'}

2. By using key

```
d={100:'Python','Abhi':101,104:105}
print(d[100])
print(d['Abhi'])
```

 # Output

```
Python
101
```

3. By using loops

```
d={100:'Python','Abhi':101,104:105}
for x in d:
    print(x)
```

```
print(d[100])
print(d['Abhi'])
```

 # Output

```
100
Abhi
104
Python
101
```

 # Write a program to enter name and percentage marks in a dictionary and
display information on the screen ?

```
rec={}
n=int(input("Enter number of students: "))
i=1
while i <=n:
    name=input("Enter Student Name: ")
    marks=input("Enter % of Marks of Student: ")
    rec[name]=marks
    i=i+1
print("Name of Student","\t","% of marks")
for x in rec:
    print("\t",x,"\t\t",rec[x])
```

 # Output

```
Enter number of students: 2
Enter Student Name: Sandy
Enter % of Marks of Student: 80
Enter Student Name: Mandy
Enter % of Marks of Student: 85
Name of Student          % of marks
         Sandy           80
         Mandy           85
```

Methods of dict

1. get(key,deafult)

i.e 1

```
d={100:'Python','Abhi':101,104:'Pratiksha'}
print(d.get(104,'Please enter valid key'))
print(d)
```

 # Output

```
Pratiksha
{100: 'Python', 'Abhi': 101, 104: 'Pratiksha'}
```

ie. 2

```
d={100:'Python','Abhi':101,104:105}
print(d.get('Anurag','Please enter valid key'))
print(d)
```

 # Output

```
Please enter valid key
{100: 'Python', 'Abhi': 101, 104: 105}
```

2. setdefault(key,value)

```
d={100:'Python','Abhi':101,104:'Pratiksha'}
print(d.setdefault(104,'B.tech'))
print(d)
```

 # Output

```
Pratiksha
{100: 'Python', 'Abhi': 101, 104: 'Pratiksha'}
```

```
d={100:'Python','Abhi':101,104:'Pratiksha'}
print(d.setdefault('Hina','B.tech'))
print(d)
```

 # Output

```
B.tech
```

{100: 'Python', 'Abhi': 101, 104: 'Pratiksha', 'Hina': 'B.tech'}

```
d={100:'Python','Abhi':101,104:105}
print(d.setdefault(104,'Please enter valid key'))
print(d)
```

 # Output
105
{100: 'Python', 'Abhi': 101, 104: 105}

Q. Difference Between get() and setdefault() method
Answer-

3. items() no-arg method

```
d={100:'Python','Abhi':101,104:105}
print(d.items())
```

 # Output
dict_items([(100, 'Python'), ('Abhi', 101), (104, 105)])

4. keys() no-arg method

```
d={100:'Python','Abhi':101,104:105}
print(d.keys())
```

 # Output
dict_keys([100, 'Abhi', 104])

5. values() no-arg method

```
d={100:'Python','Abhi':101,104:105}
print(d.values())
```

 # Output
dict_values(['Python', 101, 105])

6. pop(key)

```
d={100:'Python','Abhi':101,104:105}
print(d.pop(104))
print(d)
```

 # Output

```
105
{100: 'Python', 'Abhi': 101}
```

Q. Comparision of pop() method from List,Set,Dict.

Answer-

7. clear() no-arg

```
d={100:'Python','Abhi':101,104:105}
print(d.clear())
print(d)
```

 # Output

```
{}
```

8. copy method no-arg

```
d={100:'Python','Abhi':101,104:105}
a=d.copy()
print(d)
print(a)
```

 # Output

```
{100: 'Python', 'Abhi': 101, 104: 105}
{100: 'Python', 'Abhi': 101, 104: 105}
```

9. update(dictobj)

```
d={100:'Python','Abhi':101,104:105}
a={1:2,2:3,3:4}
d.update(a)
print(d)
```

Output

```
{100: 'Python', 'Abhi': 101, 104: 105, 1: 2, 2: 3,
3: 4}
```



Functions

functions

If a group of statements is repeatedly required then it is not recommended to write these statements everytime seperately.We have to define these statements as a single unit and we can call that unit any number of times based on our requirement without rewriting.
This unit is nothing but function.
The main advantage of functions is code Reusability.

function is a group of statements.
it's the smallest part of the program.
It is not automatically executed.

There are two types of function in python

1. predefine / Built-in-function
2 user define function

1. Built in Functions:

The functions which are coming along with Python software automatically,are called built
in functions or pre defined functions

Eg:
id()
type()
input()
eval()
etc..

2. User Defined Functions:

The functions which are developed by programmer explicitly according to business requirements ,are called user defined functions.

How to create Function in Python

Syntax

```
def func_Name(parameter):
    '''doc-string'''   # optional
    function Suite
    return expression


def demo():
    'doc-string'   # optional
    print("Hello Class GM")

demo()
Hello Class GA
```

Parameter

Parameters are input to the functions. It is also called arguments.
There are various types of argumnets in Python

1. Required arguments / positional argument
2. keyword argument
3. default argument
4. variable length argument

Explain formal/dummy & Actual Parameter .

Types of parameter

1. Required argument / positional argument

These are the arguments passed to function in correct positional order.

```
def add(a,b):
    print(a+b)
    print(a-b)
    print(a*b)
    print(a/b)
add(10,20)
# Output
30
-10
200
0.5
10
```

The number of arguments and position of arguments must be matched. If we change the order then result may be changed.
If we change the number of arguments then we will get error.

```
def arth(a,b):
    print(a+b)
    print(a-b)
    print(a*b)
    print(a/b)
a1=int(input("Enter Number:"))
a2=int(input("Enter Number:"))
arth(a1,a2)
```

2. keyword argument

We can pass argument values by keyword i.e by parameter name.s
Caller identifiy the positioning of required arguments.

```
def add(a,b):
    print(a+b)
    print(a-b)
    print(a*b)
    print(a/b)
add(b=10,a=20)
```

```
# Output
30
10
200
2.0
20
```

Note-

required argument does not follow keyword argument
positional argument follows keyword argument

3. default argument

Sometimes we can provide default values for our positional arguments.

ie. 1

```
def add(a=50,b=20):
    print(a+b)
    print(a-b)
    print(a*b)
    print(a/b)
add()
```

Note-

If we are not passing any name then only default value will be considered.

i.e 2

```
def add(a=100,b=200):
    print(a+b)
    print("Hello Class GM")

add(10,20)
```

i.e 3

```
def add(a=100,b=200):
```

```
    print(a+b)
    print("Hello Class GM")

add(10)
```

4. variable length argument

Sometimes we can pass variable number of arguments to our function,such type of arguments are called variable length arguments.
We can declare a variable length argument with * symbol as follows

```
    def f1(*n):
```

We can call this function by passing any number of arguments including zero number.
Internally all these values represented in the form of tuple.

i.e 2

```
def add(*argu):
    print("Hello i am in var-arg parameter")
    print("Hello Class GM")
    print("Hello i am in var-arg parameter")
    print("Hello Class GM")

add()
add(1,'Abhi')
add(1,'Abhi',100,1000)
```

i.e. 2

```
def prachi(*prachi):
    print("Hello i am in var-arg parameter")
    print("Hello Class GM")
    print("Hello i am in var-arg parameter")
    print("Hello Class GM")

prachi()
prachi(12)
```

ie 3

```
def sum(*n):
    total=0
    for n1 in n:
        total=total+n1
        print(total)
sum(10,20,30)
```

Return statement

In C/C++ and Java we can return only one value but in Python we can return Multiple values.
Whenever interpreter encounter the return statement then control come out from function

1. return single value

```
def add(a,b):
    c=a+b
    print(c)
    return c

add(10,20)
```

```
def add(a,b):
    c=a+b
    print(c)
    return c
    print('Ajgar wale Gupta g')
    print('Aman Wale Gupta g')
add(10,20)
```

```
def add(a,b):
    c=a+b
    return c
```

```
    print('Ajgar wale Gupta g')
    print('Aman Wale Gupta g')
d=add(10,20)
print(d)


def tiwari(a,b):
    c=a+b
    print('Tiwari Abhishek')
    return c
x=tiwari(10,50.5)
print(x)
```

2. return multiple value

```
def add_sub(a,b):
    return a+b,a-b

x,y=add_sub(10,20)
print("Sum:",x,"Sub:",y)

def add_sub(a,b):
    return a+b,a-b,a*b,a/b

x,y,z,p=add_sub(10,20)
print("Sum:",x,"Sub:",y,'mul:',z,'Div',p)

def add_sub(a,b):
    e=a+b
    f=a-b
    g=a*b
    h=a/b
    return e,f,g,h

x,y,z,p=add_sub(10,20)
print("Sum:",x,"Sub:",y,'mul:',z,'Div',p)
```

variables

Variables are named memory location used to store data.
There are two types of variables in functional Python.

Local
Global

```
c=10
d=50
def add(a,b):
    x=eval(input('Enter First number for math operations'))
    y=eval(input('Enter First number for math operations'))
    print(c*d)
    print(a*b)
    print(a+b)
    print(a/b)
    print(a-b)
    print(x+y)
    print(x-y)
    print(x/y)
    print(x-y)
    print("Hello Class GM")
    print("Hello")
    print("Python")

add(b=10,a=20)
print(c+d)
print(c,d)
print(x+y)
```

Anonymous function

A function without name is called anonymous function.
It is used for instant purpose.
We can implement anonymous function by using lambda keyword.

Syntax

lambda arguments:return one value/ expression

WAP to input two numbers & print their sum by using lambda function.

```
s=lambda a,b:(a+b)
print(s(10,20))
```

```
tiwari=lambda a,b:a-b
tiwari(10,20)
```

(* imp) Q- Difference between normal function & anonymous function.

	Normal function	Anonymous Function
1	define by def keyword	define by lmbda keyword
2	return multiple value	return only one value/expression
3	it has some name	it is nameless

```
# W.A.P to input a number a print its factorial
num=int(input("PLEASE ENTER A NUMBER TO FIND FACT  : "))
fact=1;
for i in range(1,num+1):
        fact=fact*i
```

print("THE FACTORIAL OF ",num," IS : ",fact)

Recursion

A function call itself is called recursion .

```
 # W.A.P to input a number a print its factorial by using recursion

n=int(input('Enter number for factorial:'))
def factorial(n):
    if n==0:
        result=1
    else:
        result=n*factorial(n-1)
    return result
print(factorial(n))
```

filter()
Syntax

filter() function

We can use filter() function to filter values from the given sequence based on some condition.

Syntax

filter(function,sequence)

where function argument is responsible to perform conditional check
sequence can be list or tuple or string.

without lambda

```
l=[1,2,5,8,6,4,7,9]
def even(*a):
    for x in range(0,len(l),+1):
        if l[x]%2==0:
            print(l[x])


def isEven(y)
     if x%2==0:
     return True
else:
return False
l=[]
l1=list(filter(isEven,l))
print(l)
```

with lambda

```
l=[5,4,1,2,10,15,9,80,70,80]
l1=list(filter(lambda x:x%2==0,l))
print(l1)
#Output
[4, 2, 10, 80, 70, 80]

l=eval(input('Enter List:'))
l1=list(filter(lambda x:x%2==0,l))
print(l1)
#Output
```

```
Enter List:[10,11,12,13,14,15,16]
[10, 12, 14, 16]


l=eval(input('Enter List'))
l1=list(filter(lambda x:x**2,l))
print(l1)
```

map()

For every element present in the given sequence,apply some functionality and generate new element with the required modification. For this requirement we should go for map() function.

syntax

map(function,sequence)

The function can be applied on each element of sequence and generates new sequence.

i.e.

```
l=[5,4,1,2,10,15,9,80,70,80]
l1=list(map(lambda x:3*x,l))
print(l)
print(l1)
# Output
[5, 4, 1, 2, 10, 15, 9, 80, 70, 80]
[15, 12, 3, 6, 30, 45, 27, 240, 210, 240]

l=eval(input('Enter List:'))
l1=list(map(lambda x:x**2,l))
print(l)
print(l1)

Enter List:[10,12,13,14,5,45,10,80]
[10, 12, 13, 14, 5, 45, 10, 80]
[100, 144, 169, 196, 25, 2025, 100, 6400]
```

reduce()

reduce() function reduces sequence of elements into a single element by applying the specified function.

syntax

reduce(function,sequnce)

reduce() function present in functools module and hence we should write import statement.

i.e

```
from functools import *
l=[5,4,1,2,10,15,9,80,70,80]
result=reduce(lambda x,y:x+y,l)
print(result)
#Output
276
```

```
from functools import *
l=eval(input('Enter List:'))
result=reduce(lambda x,y:x+y,l)
print(result)
#Output
Enter List:[10,12,13,14,5,45,10,80]
189
```

i.e.

```
from functools import *
l=[5,4,1,2,10,15,9,80,70,80]
result=reduce(lambda x,y:x*y,l)
result
```

Function Aliasing:

For the existing function we can give another name, which is nothing but function aliasing.

```
def wish(name):
    print("Good Morning:",name)
greeting=wish
print(id(wish))
print(id(greeting))
greeting('Sandy')
wish('Sandeep')
# Output
139894472885520
139894472885520
Good Morning: Sandy
Good Morning: Sandeep
```

```
name=input('Enter Any Name:')
def wish(name):
    print("Good Morning:",name)
greeting=wish
print(id(wish))
print(id(greeting))
greeting(name)
wish(name)
```

Note:

In the above example only one function is available but we can call that function by using either wish name or greeting name.
If we delete one name still we can access that function by using alias name

Thanks and Regards
Sandeep Kumar Sharma
Python Data Science Trainer
Techavera Solutions Pvt Ltd.
3rd Floor, Om Complex, Metro Station Road,
Naya Bans Village,
Sector 15, Noida, Uttar Pradesh 201301