PYTHON NOTE :

part 1 : python Basics

comments in python:

```
''' source : youtube and w3school
written : Rajendra R'''
# we can write title and more information like this
```

```
' source : youtube and w3school\nwritten : Rajendra R'
```

basic module:

module are 2 type external(added to another source) and build-in or internal(already exist)

```
pip install playsound
```

```
Requirement already satisfied: playsound in c:\users\rajen\anaconda3\
lib\site-packages (1.3.0)
Note: you may need to restart the kernel to use updated packages.
```

```
from playsound import playsound
playsound('E:\\PROJECT_MEDIA\\Aathma Rama Ananda Ramana - Brodha V -
No English Only Hindi Version - Spirituality - ॐ_2.mp3')

# always exchange '\' and '/' to '\\' in path to execute the command
coz single slash denoted to black_space character

# internal/build-in module  >>>pip install os
```

```
pip install os
```

```
Note: you may need to restart the kernel to use updated packages.
```

```
ERROR: Could not find a version that satisfies the requirement os
(from versions: none)
ERROR: No matching distribution found for os
```

```
import os
print(os.listdir)
```

```
<built-in function listdir>
```

PRINT() FUNCTION :

```
print("hello world")
```

```
hello world
```

```
print(2+3)
```

```
5
```

```
a = 5
b = 3
c = a + b

print("sum = ",c)
```

```
sum =  8
```

```
a = 5
b = 3
c = a + b

print("sum of ",a," and ",b," is ",c)   # default way
```

```
sum of  5  and  3  is  8
```

```
a = 5
b = 3
c = a + b

print(f"sum of {a} and {b} is {c}")    # by using f-string
```

```
sum of 5 and 3 is 8
```

```
age = '34'
msg = 'my name is john, and I am ' + age

print(msg)
```

```
my name is john, and I am 34
```

```
n1 = 'ram'
n2 = 'syam'
txt = "{} and {} are good friends."

print(txt.format(n1,n2))
```

```
ram and syam are good friends.
```

```
name = 'ram'
age = 25
place = 'taj-mahal'

print("my favorite place {2}, and i'm {0} with {1} years
old.".format(name,age,place))
```

```
my favorite place taj-mahal, and i'm ram with 25 years old.
```

```
a = 5
b = 3
c = a + b
sum = 'sum of %f and %f is'%(a,b)
print(sum,c)
```

```
sum of 5.000000 and 3.000000 is 8
```

```
x = 'I'
y = 'am'
z = 'ram'
print(x,y,z)
```

```
I am ram
```

```
x = 'I'
y = 'am'
z = 'ram'
print(x + y + z)
```

```
Iamram
```

Global Variable:

we'll study about it in Oop concept.

DATA TYPES:

```
a = 10       #integer
b = -5

print("a = ",a)
print("data type of a = ",type(a))

print("b = ",b)
print("data type of b = ",type(b))
```

```
a =  10
data type of a =  <class 'int'>
b =  -5
data type of b =  <class 'int'>
```

```
a = 10.6       #float
b = -5.13

print("a = ",a)
print("data type of a = ",type(a))

print("b = ",b)
print("data type of b = ",type(b))
```

```
a =  10.6
data type of a =  <class 'float'>
b =  -5.13
data type of b =  <class 'float'>
```

```
a = 3 + 4j  #  complex number
print(a)
print(type(a))
b = 5j
print(b)
print(type(b))
```

```
(3+4j)
<class 'complex'>
5j
<class 'complex'>
```

```
#  complex number operations
a = 3 + 4j
b = 5 + 9j


c1 = a + b
c2 = a - b
c3 = a * b
c4 = a / b
c5 = a ** b

print("a + b",c1)
print(type(c1))
print("a - b",c2)
print(type(c2))
print("a * b",c3)
print(type(c3))
print("a / b",c4)
print(type(c4))
print("a ** b",c5)
print(type(c5))
```

```
a + b (8+13j)
<class 'complex'>
a - b (-2-5j)
<class 'complex'>
a * b (-21+47j)
<class 'complex'>
a / b (0.4811320754716981-0.06603773584905659j)
<class 'complex'>
a ** b (0.714704513387209+0.19923326288380847j)
<class 'complex'>
```

```python
a = 'i am string'       #string with single qoute
print("a = ",a)
print("data type of a = ",type(a))
```

```
a =  i am string
data type of a =  <class 'str'>
```

```python
a = "i am string"       #string with double qoute
print("a = ",a)
print("data type of a = ",type(a))
```

```
a =  i am string
data type of a =  <class 'str'>
```

```python
a = '''i am
a string
with
multiple line '''       #string with multiple line with single quote
print("a = ",a)
print("data type of a = ",type(a))
```

```
a =  i am
a string
with
multiple line
data type of a =  <class 'str'>
```

```python
a = """i am
a string
with
multiple line"""      #string with multiple line with double quote
print("a = ",a)
print("data type of a = ",type(a))
```

```
a =  i am
a string
with
multiple line
data type of a =  <class 'str'>
```

```python
# string types acc to quotes
a = '"ram" and "shyam" is my friend.'
b = "mohan is meena's friend"
c = '''"ram" is mohan's brother and a good student'''
d = """"ram" is mohan's brother and a good student"""

print('strings are :')
print("1. ",a)
print(type(a))
print("2. ",b)
print(type(b))
print("3. ",c)
```

```
print(type(c))
print("4. ",d)
print(type(d))
```

```
strings are :
1.  "ram" and "shyam" is my friend.
<class 'str'>
2.  mohan is meena's friend
<class 'str'>
3.  "ram" is mohan's brother and a good student
<class 'str'>
4.  "ram" is mohan's brother and a good student
<class 'str'>
```

```
a = True  #boolean
b = False
print(a)
print(type(a))
print(b)
print(type(b))
```

```
True
<class 'bool'>
False
<class 'bool'>
```

```
print(10 > 9)
print(10 == 9)
print(10 < 9)
```

```
True
False
False
```

```
print(bool("Hello"))
print(bool(15))
```

```
True
True
```

```
a = None
print(a)
print(type(a))
```

```
None
<class 'NoneType'>
```

Variable:

Variables are containers for storing data values, like numbers and characters.

Do :

best way to naming variable is to use camel case ex. 'MyName' see 'M' and 'N' and only we can use '_' between characters

Don't :

no space, no special character

```
a = 10
ram = 5
MyName = 'raj'
my_name = 'raj'
_my_name = 'raj'
my_name2 = 'raj'
MYNAME = 'raj'

# a b = 10
# a%b = 1
# a-b = 'ram'
# 2name = 'ram'
```

assign multi value in multiple variable :

```
a,b,c = 10,11,12
print(a)
print(b)
print(c)
```

```
10
11
12
```

single value in multiple variables :

```
x = y = z = 'ram'
print(x)
print(y)
print(z)
```

```
ram
ram
ram
```

if you have collection of value like any list, tuple, set. you can extract them in single valued variable.

```
cars = ['oudi', 'suzuki', 'benz', 'toyota']
a, b, c, d = cars

print(a)
print(b)
```

```
print(c)
print(d)
```

```
oudi
suzuki
benz
toyota
```

OPERATORS :

1. arithmatic operator:

use for basic math operatons.

```
# variable with values
a = 5
b = 3
c = 2.4
d = 3.1
e = True
f = False

# sum
a = 5
b = 3
c = 2.4
d = 3.1

sum1 = a + b
sum2 = c + d
print("sum :",sum1)
print("sum :",sum2)
```

```
sum : 8
sum : 5.5
```

```
# sub
a = 5
b = 3
c = 2.4
d = 3.1

sub1 = a - b
sub2 = c - d
print("sub :",sub1)
print("sub :",sub2)
```

```
sub : 2
sub : -0.7000000000000002
```

```python
# mul
a = 5
b = 3
c = 2.4
d = 3.1

mul1 = a * b
mul2 = c * d
print("mul1 :",mul1)
print("mul2 :",mul2)
```

```
mul1 : 15
mul2 : 7.4399999999999995
```

```python
# div
a = 5
b = 3
c = 2.4
d = 3.1

div1 = b / a
div2 = d / c
print("div1 :",div1)
print("div2 :",div2)
```

```
div1 : 0.6
div2 : 1.2916666666666667
```

```python
# expo
a = 5
b = 3
c = 2.4
d = 3.1

ex1 = a ** b
ex2 = c ** d
print("ex1 :",ex1) # a to the power b
print("ex2 :",ex2) # c to the power d
```

```
ex1 : 125
ex2 : 15.088805115741808
```

```python
# rem
a = 5
b = 3
c = 2.4
d = 3.1

rem1 = a % b
rem2 = c % d
```

```
print("rem1 :",rem1)
print("rem2 :",rem2)
```

```
rem1 : 2
rem2 : 2.4
```

Floor division is a normal division operation except that it returns the largest possible integer. This integer is either less than or equal to the normal division result. Floor function is mathematically denoted by this [] symbole.

```
# div
a = 5
b = 3
c = 2.4
d = 3.1

div1 = b // a
div2 = d // c
print("div1 :",div1)
print("div2 :",div2)
```

```
div1 : 0
div2 : 1.0
```

2. assignment operators:

who hold '=' and used to assign value to exponant(variable)

```
x = 4
print("(=assignment) x=", x)

x += 3  #x = x + 3
print("(x += 3)=",x)

x -= 3  #x = x - 3
print("(x -= 3)=",x)

x *= 3  #x = x * 3
print("(x *= 3)=",x)

x /= 3  #x = x / 3
print("(x /= 3)=",x)

x %= 3  #x = x % 3
print("(x %= 3)=",x)

x /= 3  #x = x // 3
print("(x /= 3)=",x)
```

```
x **= 3  #x = x ** 3
print("(x **= 3)=",x)
```

```
(=assignment) x= 4
(x += 3)= 7
(x -= 3)= 4
(x *= 3)= 12
(x /= 3)= 4.0
(x %= 3)= 1.0
(x /= 3)= 0.3333333333333333
(x **= 3)= 0.03703703703703703
```

Comparison Operators:

Comparison operators are used to compare two values:

```
x = 5
y = 8

print("(x == y) = ",x == y)
print("(x != y) = ",x != y)
print("(x < y) = ",x < y)
print("(x <= y) = ",x <= y)
print("(x > y) = ",x > y)
print("(x >= y) = ",x >= y)
```

```
(x == y) =  False
(x != y) =  True
(x < y) =  True
(x <= y) =  True
(x > y) =  False
(x >= y) =  False
```

Logical operator:

Returns True if both statements are true rather than false. and, or, not

```
bool1 = True
bool2 = False

print("value of 'bool1 and bool2' is : ",bool1 and bool2)
print("value of 'bool1 or bool2' is ",bool1 or bool2)
print("value of 'not bool2' is",not bool2)
```

```
value of 'bool1 and bool2' is :  False
value of 'bool1 or bool2' is  True
value of 'not bool2' is True
```

identity operator:

identity operators are the operators who used to compare the object that they are equal to or not

```
x = 10
y = 15
z = 10

print("x is y = ", x is y)
print("x is y = ", x is not y)
print("x is y = ", x is z)
```

```
x is y =  False
x is y =  True
x is y =  True
```

membership operators:

```
x = [1, 2, 3, 4, 5]

print("1 in x = ",1 in x)
print("1 not in x = ",1 not in x)
```

```
1 in x =  True
1 not in x =  False
```

Bitwise operator:

Bitwise operators are used to compare (binary) numbers

```
print(6 & 3)
```

```
2
```

```
print(6 | 3)
```

```
7
```

```
print(~3)
```

```
-4
```

```
print(8 >> 2)
```

```
2
```

```
print(3 << 2)
```

```
12
```

Operator Precedence:

Operator precedence describes the order in which operations are performed. (read online)

type casting:

There may be times when you want to specify a type on to a variable. This can be done with casting. Python is an object-orientated language, and as such it uses classes to define data types, including its primitive types. it try to change datatype in all possible ways. Casting in python is therefore done using constructor functions: ***int()*** - constructs an integer number from an integer literal, a float literal (by removing all decimals), or a string literal (providing the string represents a whole number) ***float()*** - constructs a float number from an integer literal, a float literal or a string literal (providing the string represents a float or an integer) ***str()*** - constructs a string from a wide variety of data types, including strings, integer literals and float literals

```
x = int(1)
y = int(1.2)
#z = int('1.5')     # unable to do
z = int('15')

print(x,"type of x : ",type(x))
print(y,"type of x : ",type(y))
#print(z,"type of x : ",type(z))
print(z,"type of x : ",type(z))
```

```
1 type of x :  <class 'int'>
1 type of x :  <class 'int'>
15 type of x :  <class 'int'>
```

```
x = float(1)
y = float(2.8)
z = float("3")
w = float("4.2")

print(x,type(x))
print(y,type(y))
print(z,type(z))
print(w,type(w))
```

```
1.0 <class 'float'>
2.8 <class 'float'>
3.0 <class 'float'>
4.2 <class 'float'>
```

```
x = str("s1")
y = str(2)
z = str(3.0)
```

```
print(x,type(x))
print(y,type(y))
print(z,type(z))
```

```
s1 <class 'str'>
2 <class 'str'>
3.0 <class 'str'>
```

type conversion:

type conversion is a technique to convert data type from one to another in possible ways. to see the data type we use type() function and to covernt data type in typer converstion we use int(), float() and complex() functions

```
x = 10
y = 7.5
z = 5j

# before conversion
type(x)
type(y)
type(z)

#convert from int to float:
a = float(x)
print(a)
print(type(a))

#convert from float to int:
b = int(y)
print(b)
print(type(b))

#convert from int to complex:
c = complex(x)
print(c)
print(type(c))
```

```
10.0
<class 'float'>
7
<class 'int'>
(10+0j)
<class 'complex'>
```

Input() function:

it's used to take user input. By default, it returns the user input in form of a string.

```
name = input('enter your name : ')
add = input("enter your address : ")

print("my name is ",name," and ","i live in ",add)   #default type
print(f"my name is {name} and i live in {add}")      # using f-string
```

```
enter your name : Rajendra Randa
enter your address : indore madhya pradesh india
my name is  Rajendra Randa  and  i live in  indore madhya pradesh
india
my name is Rajendra Randa and i live in indore madhya pradesh india
```

input function take input as 'string' by default but with type conversion we can take input as any data type i.e. int and float

```
a = input("enter any number: ")
print(a)
print(type(a))

a = float(input("enter any float number: "))
print(a)
print(type(a))

a = int(input("enter any int number: "))
print(a)
print(type(a))
```

```
enter any number: 12
12
<class 'str'>
enter any float number: 154
154.0
<class 'float'>
enter any int number: 124
124
<class 'int'>
```

```
a = float(input("enter no. 1 :"))
b = float(input("enter no. 2 :"))
c = a + b
sum = 'sum of %f and %f is'%(a,b)
print(sum,c)
```

```
enter no. 1 :98
enter no. 2 :79
sum of 98.000000 and 79.000000 is 177.0
```

string operation:

there are various operation and functions, we can applied on strings.

```
greet = 'good morning !!!, '
name = input("enter name : ")

print(greet+name)

msg = greet+name
print(msg)
```

```
enter name : rajendra kumar randa
good morning !!!, rajendra kumar randa
good morning !!!, rajendra kumar randa
```

```
name = "Rajendra Kumar Randa"

print("index of 0 :", name[0])
print("index of 2 :", name[2])
print("index of 18 :", name[18])
# print("index of 23 :", name[23])   #IndexError: string index out of
range
```

```
index of 0 : R
index of 2 : j
index of 18 : d
```

nagative index:

it goes to string = [-n, ... , -4, -3, -2, -1, 0]

```
print("index of 2 :", name[-2])
print("index of 2 :", name[-14])
```

```
index of 2 : d
index of 2 : r
```

slicing : dividing string into parts

```
str = "A quick brown fox jumps over the lazy dog ."

print("index 0 to 5 : ", str[0:6])  # [starting_index, ending+1]
print("index 0 to 5 : ",str[:6])

print("index -6 to -2 : ", str[-6:-2])

print("index -9 to 0 : ", str[-9:])
print("index -9 to 0 : ", str[-9:])
```

```
index 0 to 5 :  A quic
index 0 to 5 :  A quic
index -6 to -2 :   dog
```

```
index -9 to 0 :  azy dog .
index -9 to 0 :  azy dog .
```

some more examples:

```
name = "Rajendra Kumar Randa"

print(name)
print(type(name))
print(len(name))

print(name[0])
print(name[3])
print(name[4]) #not work for out of index
print(name[:5])
print(name[:4])
print(name[0:])
print(name[1:3]) #[1:2]
c = name[-4:-2] #nagative index
print(c)
print(name[-4:-2])
d = name[1::5] #ski p indexing value
d = name[0:6:1] #string[start_index:end_index:skip_index]
print(d)
name[::2]
num = '0123456789'
num[::2] #skil each 2nd values
```

```
Rajendra Kumar Randa
<class 'str'>
20
R
e
n
Rajen
Raje
Rajendra Kumar Randa
aj
an
an
Rajend
```

```
'02468'
```

```
# silicing wwork with collection of datatype (list,tuple...)
l = [0, 1, 2, 3, 4, 5, 6, 7, 8]
m = l[0:9:2]
print(m)
```

```
[0, 2, 4, 6, 8]
```

string functions:

```
story = '''In “The Necklace” by Guy de Maupassant, the main character,

Mathilde, has always dreamed of being an aristocrat but lives in
poverty. Embarrassed about her lack of fine possessions, she borrows a

necklace from a wealthy friend but loses it. The story is known for
its
subversive and influential twist ending.'''

# string functions:
print(story)
print("length : ",len(story))
```

```
In “The Necklace” by Guy de Maupassant, the main character,
Mathilde, has always dreamed of being an aristocrat but lives in
poverty. Embarrassed about her lack of fine possessions, she borrows a

necklace from a wealthy friend but loses it. The story is known for
its
subversive and influential twist ending.
length :  312
```

```
print(story.endswith('.'))
print(story.endswith('ending'))
print(story.count('v'))
print(story.count(' '))
print(story.capitalize())
print(story.find(' '))
print(story.replace("main", "lead")) #replace all accurances
```

```
True
False
4
50
In “the necklace” by guy de maupassant, the main character,
mathilde, has always dreamed of being an aristocrat but lives in
poverty. embarrassed about her lack of fine possessions, she borrows a

necklace from a wealthy friend but loses it. the story is known for
its
subversive and influential twist ending.
2
In “The Necklace” by Guy de Maupassant, the lead character,
Mathilde, has always dreamed of being an aristocrat but lives in
poverty. Embarrassed about her lack of fine possessions, she borrows a

necklace from a wealthy friend but loses it. The story is known for
its
subversive and influential twist ending.
```

Let's see more with new example:

```
story = "A quick brown fox jumps  over the lazy dog . dog are smart
pet."

print(story)
print(len(story))
print(story.endswith("dog"))
print(story.endswith("."))
print(story.count("v"))
print(story.count("brown"))
print(story.count(" "))
print(story.find("fox"))
print(story.find(" "))
print(story.replace("dog", "cat"))
```

```
A quick brown fox jumps  over the lazy dog . dog are smart pet.
63
False
True
1
1
14
14
1
A quick brown fox jumps  over the lazy cat . cat are smart pet.
```

space sequence character:

they are special characters who used to make output better. some of them are : \n : new line \t : a tab \ : useing : use same quote as start or end inside the strings and etc.

```
new_story = "this is a ball.\nthis color is blue.\nin maths,\n\tif we
divide 2 into 2 (2/2) is 1\n and this is harry\'s\tball.\\'"

print(new_story)
```

```
this is a ball.
this color is blue.
in maths,
     if we divide 2 into 2 (2/2) is 1
 and this is harry's  ball.\'
```

excersize:

```
# ex 1
letter = '''Dear <|NAME|> ,
greeting from company 'ABC'. I'm happy to tell about your selection.
you're selected.\n
```

```
have a great day!
\t\t\t\t\t\tdate: <|DATE|>'''

name = input("enter your name: ")
date = input("enter date: ")

letter = letter.replace("<|NAME|>",name)
letter = letter.replace("<|DATE|>",date)

print("********** greeting ************")
print(letter)
```

```
enter your name: Rajendra Kumar Randa
enter date: 22/12/2022
********** greeting ************
Dear Rajendra Kumar Randa ,
greeting from company 'ABC'. I'm happy to tell about your selection.
you're selected.

have a great day!
                                    date: 22/12/2022
```

```
str1 = "this is my  ball."
str2 = "this is my pen."
str3 = "this  is my  book."
doublespace1 = str1.find("  ")
doublespace2 = str2.find("  ")
print(doublespace1)
print(doublespace2)

str3 = str3.replace("  "," ")
print(str3)
```

```
10
-1
this is my book.
```

```
ltr = "dear harry, this python course is nice!thanks!"
formatted_letter = "dear harry, \n\tthis python course is nice!\n\t\t\
t\t\t\t\tthanks!"
print(formatted_letter)
```

```
dear harry,
     this python course is nice!
                                          thanks!
```

List (data type)

Lists are used to store multiple items in a single variable. it's changable/ mutable and indexed/ordered, it allows duplicates.

```
l = []
print(l)
print(len(l))
print(type(l))

# after add an element
l.insert(0,2)
print(l)
print(len(l))
print(type(l))
```

```
[]
0
<class 'list'>
[2]
1
<class 'list'>
```

```
a = [1, 2.3, 'ram', True, 0, 0, 0]

print("list : ",a)
print("type : ",type(a))
print("length : ",len(a))
print("index of [0] : ",a[0])
print("index of [2] : ",a[2])
# print("index of [6] : ",a[6])       # out of index
print("type of element of list on index[1] : ",type(a[1]))
print("type of element of list on index[2] : ",type(a[2]))
```

```
list :  [1, 2.3, 'ram', True, 0, 0, 0]
type :  <class 'list'>
length :  7
index of [0] :  1
index of [2] :  ram
type of element of list on index[1] :  <class 'float'>
type of element of list on index[2] :  <class 'str'>
```

```
a[6] = 9
print("after update item on index[6] list will be : ",a)
```

```
after update item on index[6] list will be :  [1, 2.3, 'ram', True, 0,
0, 9]
```

```
friends = ["ram","syam","mohan","sohan","meena","sita", 4, 500]

print(friends)
```

```
print(friends[:4]) #slice with index
print(friends[-4:]) #slice with index

friends[-1] = "kajal"
print(friends)
```

```
['ram', 'syam', 'mohan', 'sohan', 'meena', 'sita', 4, 500]
['ram', 'syam', 'mohan', 'sohan']
['meena', 'sita', 4, 500]
['ram', 'syam', 'mohan', 'sohan', 'meena', 'sita', 4, 'kajal']
```

functions on list:

```
num = [1, 16, 56, 10, 0, 14, 0]

print("num[:3] = ",num[:3])    # num[0:3]
print("num[3:] = ",num[3:])    # num[3:last element]
print("num[::2] = ",num[::2])    #num[0:n-1:2nd element skipped]
```

```
num[:3] =  [1, 16, 56]
num[3:] =  [10, 0, 14, 0]
num[::2] =  [1, 56, 0, 0]
```

sort the list of char./no.

```
num = [1, 16, 56, 10, 0, 14, 0]
num.sort()
print("sorted list : ",num)
```

```
sorted list :  [0, 0, 1, 10, 14, 16, 56]
```

reverse the list:

```
num = [1, 16, 56, 10, 0, 14, 0]
num.reverse()
print("reversed list : ",num)
```

```
reversed list :  [0, 14, 0, 10, 56, 16, 1]
```

some baasic functions on list:

```
num = [1, 16, 56, 10, 14, 1, 0]

print(num)

num.sort()
print("sort : ",num)

num.reverse()
```

```
print("reverse : ",num)

num.append(50)
print("append 50 : ",num)

num.insert(0, 2.3)      # insert(index no, value)
print("insert '2.3' on index[0] : ",num)

num.pop(1)      # pop(index)
print("pop '1' : ",num)

num.pop()
print("pop : ",num)  #remove last item

num.remove(0)   # remove(value)
print("remove '0' : ",num)

del num[1]      #delete item permanantly from source list
print("del index[1] : ",num)
```

extend : murge two datatype

```
# list with list
x = ['a','b','c']
y = ['p','q','z']
x.extend(y)
print(x)
# list with tuple
p = ['a','b','c']
q = (1,2,3)
p.extend(q)
print(p)
```

```
['a', 'b', 'c', 'p', 'q', 'z']
['a', 'b', 'c', 1, 2, 3]
```

list with loop:

```
l = ['ram', 'sohan', 'mohan']
x = input("enter name :")
if x in l:
    print("yes,it is...")
else:
    print("no, it is not...")
```

```
enter name :ram
yes,it is...
```

```
# some basic use list in loop
x = [0, 10, 20, 30, 40, 50, 60, 70, 8, 90, 100]

print("for:")
```

```
# print item one by one
for item in x:
    print(item)


print("for:")
# print via index number using range() and len()
for item in range(len(x)):
    print(x[4])

print("while :")
# using while loop
i = 0
while item<len(x):
    print(x[item])
    item = item + 1
```

```
for:
0
10
20
30
40
50
60
70
8
90
100
for:
40
40
40
40
40
40
40
40
40
40
40
while :
100
```

```
print("for:")
# using for loop
x = [0, 10, 20, 30, 40, 50, 60, 70, 8, 90, 100]

[print(item) for item in x]
```

```
for:
0
10
20
30
40
50
60
70
8
90
100

[None, None, None, None, None, None, None, None, None, None, None]
```

tuple (data type):

it is collection of all type of data type. it is imutable(unchangable), allowed to hold duplicate values and also indexed/ordered.

```
empty_tuple = ()
print(empty_tuple)
print(type(empty_tuple))
print(len(empty_tuple))

()
<class 'tuple'>
0

t = (3)   # wrong way
print(t)
print(type(t))

t = (2,)  # right way
print(t)
print(type(t))

3
<class 'int'>
(2,)
<class 'tuple'>
```

operations on tuple:

```
t = (0, 12, 2.5, 24, 50, 2, 2, 2)
print(t)
print(type(t))
print(type(t[2]))
print("")
```

```
print("count '2' : ", t.count(2))
print("index(50) : ", t.index(50))   # return index no by value
index(value)
```

```
(0, 12, 2.5, 24, 50, 2, 2, 2)
<class 'tuple'>
<class 'float'>

count '2' :  3
index(50) :  4
```

```
# convert tuple into list to update it?
x = ('apple','banana','cherry')
print(x)
print(type(x))

y = list(x)
y[1] = 'kiwi'

print(y)
print(type(y))

x = tuple(y)
print(x)
print(type(x))

# by this way we can perform all operation on tuples
```

```
('apple', 'banana', 'cherry')
<class 'tuple'>
['apple', 'kiwi', 'cherry']
<class 'list'>
('apple', 'kiwi', 'cherry')
<class 'tuple'>
```

```
# loop with tuple
a = (1,3,5,6,7,8)
for i in a:
    print(i)
```

```
1
3
5
6
7
8
```

```
# loop with tuple and range
a = (1,3,5,6,7,8)
for i in range(len(a)):
    print(a[i])
```

```
1
3
5
6
7
8
```

```
# tuple with while loop
ab = (1,2,3,4,5,6)
i = 0
while i < len(ab):
    print(ab[i])
    i = i + 1
```

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

math with tuple:

```
# Join two tuples:
tuple1 = ("a", "b" , "c")
tuple2 = (1, 2, 3)
tuple3 = tuple1 + tuple2
print(tuple3)

# Multiply the fruits tuple by 2:
fruits = ("apple", "banana", "cherry")
mytuple = fruits * 2
print(mytuple)
```

```
('a', 'b', 'c', 1, 2, 3)
('apple', 'banana', 'cherry', 'apple', 'banana', 'cherry')
```

Using Asterisk*

If the number of variables is less than the number of values, you can add an * to the variable name and the values will be assigned to the variable as a list:

```
fruits = ("apple", "banana", "cherry", "strawberry", "raspberry")

(green, yellow, *red) = fruits

print(green)
print(yellow)
print(red)
```

```
apple
banana
['cherry', 'strawberry', 'raspberry']
```

revision (list & tuple):

```
f1 = input("enter fruit no 1 : ")
f2 = input("enter fruit no 2 : ")
f3 = input("enter fruit no 3 : ")
f4 = input("enter fruit no 4 : ")
f5 = input("enter fruit no 5 : ")

l =[f1, f2, f3, f4, f5]

print("list of give fruits are : ", l)
print(type(l))
print(len(l))
```

```
enter fruit no 1 : apple
enter fruit no 2 : banana
enter fruit no 3 : papaya
enter fruit no 4 : graps
enter fruit no 5 : cherry
list of give fruits are :  ['apple', 'banana', 'papaya', 'graps',
'cherry']
<class 'list'>
5
```

```
m1  = int(input("enter marks of student no. 1 : "))
m2  = int(input("enter marks of student no. 2 : "))
m3  = int(input("enter marks of student no. 3 : "))
m4  = int(input("enter marks of student no. 4 : "))
m5  = int(input("enter marks of student no. 5 : "))

l = [m1, m2, m3, m4, m5]
l.sort()
print("marks of students are in sorted form : ", l)
```

```
enter marks of student no. 1 : 98
enter marks of student no. 2 : 79
enter marks of student no. 3 : 86
enter marks of student no. 4 : 98
enter marks of student no. 5 : 59
marks of students are in sorted form :  [59, 79, 86, 98, 98]
```

```
mylist = [12, 23, 34, 45]
# way 1 : without using loop
sum = mylist[0] + mylist[1] + mylist[2] + mylist[3]
print("sum : ",sum)
# way 2
```

```
total = 0
for x in range(0, len(mylist)):
    total = total + mylist[x]
print("sum : ",total)
```

```
sum :  114
sum :  114
```

```
t  = ( 0, 21, 1, 0, 4, 6, 23, 0, 0 )
print("total no of zeroes : ",t.count(0))
```

```
total no of zeroes :  4
```

```
 # note the double round-brackets
thistuple = tuple(("apple", "banana", "cherry"))
print(thistuple)
```

```
('apple', 'banana', 'cherry')
```

```
t = (1,2,3,4,56)
if 1 in t:
    print("true")
else:
    print("false")
```

```
true
```

Dictionary:

its collection of 'key-value' pairs. it is indexed, unordered, mutable(changable), and hold non-duplicate keys.

NOTE: if two keys wpuld be same than key will hold new updated value and if values are same it doesn't matter.

```
empty_dict = {}
print(empty_dict)
print(type(empty_dict))

d = {'a': 1,'b': 2,'c':3,}
print(d)
print(type(d))

d2 = dict(x = 1, y = 3, z = 'raj')
print(d2)
print(type(d2))
```

```
{}
<class 'dict'>
{'a': 1, 'b': 2, 'c': 3}
<class 'dict'>
```

```
{'x': 1, 'y': 3, 'z': 'raj'}
<class 'dict'>
```

```
myDict = {
    "red"  : 'one type of color',
    "fast" : 'in a quick manner',
    "harry": 'a programmer',
    "l"    : [1, 2, 3, 4, 5],
    "t"    : (10, 20, 30, 40, 50),
    "D"    : {
        "nu": [11, 12, 13, 14],
        "tu": (110, 120, 130, 140),
        "di": {
            "x" : [7, 8, 9],
            "y" : (77, 88, 99),
            "d2":{"m": [9, 99, 999],
                  "n": [8, 88, 888],
                  "o":(5,55,555),
                 }
              }
    },
  }

# print dictionary
print("Dictionary is : ",myDict)

# type of element N dictionary
print("type of Dictionary is : ",type(myDict))
print("type of 'o' is : ",type(myDict['D']['di']['d2']['o']))


# print elements
print("value of 'red' is : ",myDict['red'])
print("value of 'fast' is : ",myDict['fast'])
print("value of 'harry' is : ",myDict['harry'])
print("value of 'l' is : ",myDict['l'])
print("value of 't' is : ",myDict['t'])
print("value of 'D' is : ",myDict['D'])

# print inner elements
print("value of 'nu' is : ",myDict['D']['nu'])
print("value of 'di' is : ",myDict['D']['di'])

# print inner value
print("value of 'x' is : ",myDict['D']['di']['x'])
print("value of 'd2' is : ",myDict['D']['di']['d2'])
print("value of 'o' is : ",myDict['D']['di']['d2']['o'])

print("keys : ",myDict.keys())
print("type of values : ",type(myDict.values()))
```

```
print("values : ",myDict.values())
print("type of keys : ",type(myDict.keys()))

# item() if you want to see key-value pair
print("item : ",myDict.items())

# convert key to list of keys:
print("keys : ",list(myDict.keys()))

# add value
myDict['marks'] = ['a','b','c']
myDict['D']['di']['hero'] = 'save the life'
print(myDict.items())
```

```
Dictionary is :  {'red': 'one type of color', 'fast': 'in a quick
manner', 'harry': 'a programmer', 'l': [1, 2, 3, 4, 5], 't': (10, 20,
30, 40, 50), 'D': {'nu': [11, 12, 13, 14], 'tu': (110, 120, 130, 140),
'di': {'x': [7, 8, 9], 'y': (77, 88, 99), 'd2': {'m': [9, 99, 999],
'n': [8, 88, 888], 'o': (5, 55, 555)}}}}
type of Dictionary is :  <class 'dict'>
type of 'o' is :  <class 'tuple'>
value of 'red' is :  one type of color
value of 'fast' is :  in a quick manner
value of 'harry' is :  a programmer
value of 'l' is :  [1, 2, 3, 4, 5]
value of 't' is :  (10, 20, 30, 40, 50)
value of 'D' is :  {'nu': [11, 12, 13, 14], 'tu': (110, 120, 130,
140), 'di': {'x': [7, 8, 9], 'y': (77, 88, 99), 'd2': {'m': [9, 99,
999], 'n': [8, 88, 888], 'o': (5, 55, 555)}}}
value of 'nu' is :  [11, 12, 13, 14]
value of 'di' is :  {'x': [7, 8, 9], 'y': (77, 88, 99), 'd2': {'m':
[9, 99, 999], 'n': [8, 88, 888], 'o': (5, 55, 555)}}
value of 'x' is :  [7, 8, 9]
value of 'd2' is :  {'m': [9, 99, 999], 'n': [8, 88, 888], 'o': (5,
55, 555)}
value of 'o' is :  (5, 55, 555)
keys :  dict_keys(['red', 'fast', 'harry', 'l', 't', 'D'])
type of values :  <class 'dict_values'>
values :  dict_values(['one type of color', 'in a quick manner', 'a
programmer', [1, 2, 3, 4, 5], (10, 20, 30, 40, 50), {'nu': [11, 12,
13, 14], 'tu': (110, 120, 130, 140), 'di': {'x': [7, 8, 9], 'y': (77,
88, 99), 'd2': {'m': [9, 99, 999], 'n': [8, 88, 888], 'o': (5, 55,
555)}}}])
type of keys :  <class 'dict_keys'>
item :  dict_items([('red', 'one type of color'), ('fast', 'in a quick
manner'), ('harry', 'a programmer'), ('l', [1, 2, 3, 4, 5]), ('t',
(10, 20, 30, 40, 50)), ('D', {'nu': [11, 12, 13, 14], 'tu': (110, 120,
130, 140), 'di': {'x': [7, 8, 9], 'y': (77, 88, 99), 'd2': {'m': [9,
99, 999], 'n': [8, 88, 888], 'o': (5, 55, 555)}}})])
keys :  ['red', 'fast', 'harry', 'l', 't', 'D']
```

```
dict_items([('red', 'one type of color'), ('fast', 'in a quick
manner'), ('harry', 'a programmer'), ('l', [1, 2, 3, 4, 5]), ('t',
(10, 20, 30, 40, 50)), ('D', {'nu': [11, 12, 13, 14], 'tu': (110, 120,
130, 140), 'di': {'x': [7, 8, 9], 'y': (77, 88, 99), 'd2': {'m': [9,
99, 999], 'n': [8, 88, 888], 'o': (5, 55, 555)}, 'hero': 'save the
life'}}), ('marks', ['a', 'b', 'c'])])
```

```
newDict = {
    'a': 'this is a',
    'b': 'this is b',
    'c': 'this is c',
}
print(newDict)
print(newDict.items())
print(newDict.keys())
print(newDict.values())

newDict['d'] = 'this is d'
print('after update : ',newDict)

print(newDict['d'])
print(newDict.get('d'))
```

```
{'a': 'this is a', 'b': 'this is b', 'c': 'this is c'}
dict_items([('a', 'this is a'), ('b', 'this is b'), ('c', 'this is
c')])
dict_keys(['a', 'b', 'c'])
dict_values(['this is a', 'this is b', 'this is c'])
after update :  {'a': 'this is a', 'b': 'this is b', 'c': 'this is c',
'd': 'this is d'}
this is d
this is d
```

```
# if with dictionary:

thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
if "model" in thisdict:
    print("Yes, 'model' is one of the keys in the thisdict
dictionary")
else:
    print("sorry, we keyword doesn't match...")
```

```
Yes, 'model' is one of the keys in the thisdict dictionary
```

```
# Change the value:
thisdict = {
  "brand": "Ford",
```

```
  "model": "Mustang",
  "year": 1964
}
thisdict["year"] = 2018
print(thisdict)

# update dictionary
thisdict.update({'year':2020})
print(thisdict)

# add pair
thisdict.update({'color': 'blue'})
print(thisdict)

# add pair
thisdict['engin'] = '900CC'
print(thisdict)

# delete pair
del thisdict['engin']
print(thisdict)

# loop in dictionary
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
 }

# print all key names
for x in thisdict:
    print(x)

# print all keys
for x in thisdict.keys():
    print(x)

# print all values
for x in thisdict:
    print(thisdict[x])

# print all values
for x in thisdict.values():
    print(x)

# print pairs of dict
for x,y in thisdict.items():
    print(x,y)

# print pairs of dict in better way
```

```
for x,y in thisdict.items():
    print(x," : ",y)
```

```
brand
model
year
brand
model
year
Ford
Mustang
1964
Ford
Mustang
1964
brand Ford
model Mustang
year 1964
brand  :  Ford
model  :  Mustang
year  :  1964
```

some functions with dictionary :

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
 }
# make a copy of dict using co[y]
mydict = thisdict.copy()
print(mydict)

# make a copy of dict using dict()
mydict1 = dict(thisdict)
print(mydict1)
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

```
# nexted dictionary
child1 = {
  "name" : "Emil",
  "year" : 2004
}
child2 = {
  "name" : "Tobias",
  "year" : 2007
}
child3 = {
```

```
  "name" : "Linus",
  "year" : 2011
}

myfamily = {
  "child1" : child1,
  "child2" : child2,
  "child3" : child3
}

print(myfamily["child2"]["name"])
```

```
Tobias
```

Dictionary Methods:

Python has a set of built-in methods that you can use on dictionaries.

set:

it is non-pair, immutable/non-changable, non.indexed, unordered and not hold duplicate values. you can't change value but u can add or delete values.

```
# empty set
g = set()
print(g)
print(type(g))

g.add(10)
g.add(20)
g.add(30)
print(g)
```

```
set()
<class 'set'>
{10, 20, 30}
```

```
s = {10, 20, 30}
print("s : ",s)
print(type(s))
s.add(40)
print("s : ",s)
print(len(s))
```

```
s :  {10, 20, 30}
<class 'set'>
s :  {40, 10, 20, 30}
4
```

```python
a = {11, 21, 31, 42, 51, True, 12.5, 3+9j}
# a.add(['a','b'])
print(a)
a.remove(21)
print("remove 21 :",a)
a.pop()  # remove random item
print(a)
```

```
{True, 42, 11, 12.5, 51, 21, (3+9j), 31}
remove 21 : {True, 42, 11, 12.5, 51, (3+9j), 31}
{42, 11, 12.5, 51, (3+9j), 31}
```

```python
b = set()
b.add(10)
b.add(20)
b.add(30)
b.add(35)
print(b)
print(type(b))
```

```
{10, 35, 20, 30}
<class 'set'>
```

```python
# union and intersection
s1 = {1, 2, 3, 4, 5, 7, 9, 11}
s2 = {1, 2, 4, 6, 8, 10, 3, 5}
x = s1.union(s2)
y = s1.intersection(s2)
z = s1.symmetric_difference_update(s2)
print('s1 = ',s1)
print('s2 = ',s2)
print("union of 's1' and 's2' is :",x)
print("intersection of 's1' and 's2' is :",y)
print("all element who are not in both set s1 and s2 : ",z)
```

```
s1 =  {6, 7, 8, 9, 10, 11}
s2 =  {1, 2, 3, 4, 5, 6, 8, 10}
union of 's1' and 's2' is : {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11}
intersection of 's1' and 's2' is : {1, 2, 3, 4, 5}
all element who are not in both set s1 and s2 :  None
```

```python
s1 = set()
s2 = {'a', True, 2.5, 12}
s3 = set(('char', False, 12.65, 13))
print(s1,s2,s3)
```

```
set() {True, 2.5, 'a', 12} {False, 12.65, 13, 'char'}
```

```python
# set with loop:
myset = {'raj', 'mohan', 'sita', 'gita'}
for x in myset:
    print(x)
```

```
# check value present or not
print('raj' in myset)
```

```
raj
gita
mohan
sita
True
```

```
a1 = {'a', 'b', 'c'}
a2 = {1, 2, 3}
a1.update(a2)
print(a1)
a2.update(a1)
print(a2)
```

```
{'c', 1, 'a', 2, 3, 'b'}
{1, 2, 3, 'c', 'b', 'a'}
```

```
# delete items in set
o = {'a', 12, 'd', 45, 's'}
print('o is : ',o)
o.remove(12)
print('remove : ',o)
o.discard('a')
print('discard : ',o)
```

```
# delete any set
x = {1, 3, 5, 7, 9}
print(x)
del x
# print(x)      # 'x' is not define
```

```
o is :  {'s', 'a', 12, 45, 'd'}
remove :  {'s', 'a', 45, 'd'}
discard :  {'s', 45, 'd'}
{1, 3, 5, 7, 9}
```

Set Methods :

excersize:

dictionary() and set()

```
f1 = input('enter favorite language of geeta : ')
f2 = input('enter favorite language of meeta : ')
f3 = input('enter favorite language of seeta : ')
f4 = input('enter favorite language of neeta : ')
```

```
dict = {}
dict['geeta'] = f1
dict['meeta'] = f2
dict['seeta'] = f3
dict['neeta'] = f4
print(dict.items())
```

```
enter favorite language of geeta : hin
enter favorite language of meeta : sans
enter favorite language of seeta : sp
enter favorite language of neeta : ph
dict_items([('geeta', 'hin'), ('meeta', 'sans'), ('seeta', 'sp'),
('neeta', 'ph')])
```

Indentation:

the spaces at the beginning of a code line. Where in other

programming languages the indentation in code is for readability only, the indentation in Python is very important. Python uses indentation to indicate a block of code.

pass statement:

if statements cannot be empty, but if you for some reason have an if statement with no content, put in the pass statement to avoid getting an error.

```
a = 33
b = 200

if b > a:
    pass
```

```
i = 4
if i > 3 :
    pass
while i > 6 :
    pass
def run(player):
    pass
print("that's it...")
```

```
that's it...
```

we can use pass statement to skip writing control statement or code inside of function to complete it later

```
def dummy(drug):
    pass
```

```
dummy('a')
print("usefull to create empty function using 'pass' statement ")
```

```
usefull to create empty function using 'pass' statement
```

Control statements:

```
        it works like a ladder when 1st condition will go true it'll
go for execute the code under it rather than go for next condition or
code.it's work ladder if 1st will true than it'll never go for next
```

if, elif and else also optional and work anywhere if-elif-else ladder

1. if control statement: it works only and only when if condition will be true than next statement will execute.

```
a = 33
b = 200
if b > a :
    print("b is greater than a")
# print() will execute only and only if condition will true
```

```
b is greater than a
```

```
x = 12
y = 10
if(x<y):
    print("y is greater than x")
# no output
```

elif: (nested if):

"if the previous conditions were not true, then try this condition"

```
a = 33
b = 33

if a > b :
    print("a is greater than b...")
elif a == b :
    print("a and b other are equals")
```

```
a and b other are equals
```

```
a = 12
if(a==7):
    print("yes")
elif(a>56):
    print("yes or no") #no output
```

```
#  default nested if : we rarely use nested if
# let's see
x = 40
if x > 10 :
    print("x is above 10 ")
    if x > 20 :
        print("and also above 20")
    else:
        print("and not above 20")
else:
    print("less then 10")
```

```
x is above 10
and also above 20
```

else:

it works whenever another all conditions will be false. it can work with all statements.

EXAMPLE:

```
a = 22
if a > 9 :
    print("greater")
else:
    print("lesser")
```

```
greater
```

```
age = int(input("enter age: "))
if age >= 18 :
    print("yes")
else:
    print("no")
```

```
enter age: 13
no
```

```
# u can work if your age is : age greater than 33 and also age less
than 55
print("u can work if your age is : age greater than 33 and also age
less than 55")
age = int(input("enter age: "))
if(age>33 and age<55):
    print("you're ",age,",you can work....")
else:
    print("you're ",age,",you can't work...")
```

```
u can work if your age is : age greater than 33 and also age less than
55
```

```
enter age: 50
you're  50 ,you can work....
```

if-elif-else: IN SHORT HANDS 🫶

```
a = 200
b = 33
if b > a :
    print("b is greater than a")
elif a == b:
    print("a and b both are equals")
else:
    print("a is greater than b")
```

```
a is greater than b
```

one life with if:

```
a, b = 15, 12
if a > b : print("a is greater than b")
```

```
a is greater than b
```

one line with if-else:

```
a = 2
b = 330
print("a is greater than b") if a > b else print("b is greater than
a")
```

```
b is greater than a
```

one line with ternary operator or conditional expressions

```
a = 330
b = 330
print("A") if a > b else print("=") if b == a else print("B")
```

```
=
```

'and' in if statements:

```
a = 200
b = 33
c = 500
if a > b and c > a :
    print("bothe conditions are true...")
```

```
bothe conditions are true...
```

'or' in if statements:

```
a = 200
b = 33
```

```
c = 500
if a > b or c < a :
    print("anyone of them, condition should be true...")
```

```
anyone of them, condition should be true...
```

'not' in if statement:

```
a = 10
b = 5
if not a < b :
    print("b is not greater than a")
```

```
b is not greater than a
```

'in' and 'is' if statement:

```
a = None
if a is None:
    print(True)
elif a < 0 or a > 0 :
    print(False)
else:
    print("a is zero")
```

```
True
```

```
a = 12
if(a==7):
    print("yes")
elif(a>56):
    print("yes or no")
else:
    print("i'm optional")
```

```
i'm optional
```

prectice : with if

quiz: a spam comment is defined as a text containing keywords[make a lot of money, buy now,suscribe this,click this,]WAP to detect it.

```
print("smap word: make a lot of money, buy now, suscribe this, click
this")

comment = input("to check your comment spam or not\nenter your comment
:\n")
if("make a lot of money" in comment or "buy now" in comment or
"suscribe this" in comment or "click this" in comment):
    print("your comment is spam..")
```

```
else:
    print("your comment is not spam..")
```

```
smap word: make a lot of money, buy now, suscribe this, click this
to check your comment spam or not
enter your comment :
hurry up!! get a lot of money and gift cards,
your comment is not spam..
```

```
username = input("enter your username : ")
total_char = len(username)
if total_char >= 10 :
    print("valid username")
else:
    print("invalid username")
```

```
enter your username : emsrajen
invalid username
```

```
names = ["raj","mohan","sohan","sita","gita"]
name = input("enter name to check wether name is in list or not : ")
if name in names:
    print("name is exist in list...")
else:
    print("name isn't exist in list...")
```

```
enter name to check wether name is in list or not : raj
name is exist in list...
```

```
post = input("enter post : ")
name = input("enter name to search : ")
if name in post:
    print("name is exist in list...")
else:
    print("name isn't exist in list...")
```

```
enter post : dummies transforms the hard-to-understand into easy-to-
use to enable learners at every level to fuel their pursuit of
professional and personal advancement.
enter name to search : fuel
name is exist in list...
```

LOOP : repeatation of block of statement

1. while loop :

```
i = 0
while i < 10 :
    print("yes ",i)
```

```
    i = i + 1
print("Done...!")
```

```
yes  0
yes  1
yes  2
yes  3
yes  4
yes  5
yes  6
yes  7
yes  8
yes  9
Done...!
```

```
print("numbers from 1 to 15 : ")
i = 1
while i <= 15 :
    print(i)
    i += 1
print("that's it")
```

```
numbers from 1 to 15 :
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
that's it
```

```
print("numbers from 1 to 15 : ")
i = 1
while i < 16 :
    print(i)
    i += 1
print("that's it")
```

```
numbers from 1 to 15 :
1
2
3
```

```
4
5
6
7
8
9
10
11
12
13
14
15
that's it
```

```
# print string multiple times
i = 1
while i < 6 :
    print(i, ". I'm rajendra")
    i = i + 1
```

```
1 . I'm rajendra
2 . I'm rajendra
3 . I'm rajendra
4 . I'm rajendra
5 . I'm rajendra
```

```
# print element from collection / list
fruit = ['apple', 'banana', 'papaya', 'graps', 'mango', 'cherry']
i = 0
while i < len(fruit) :
    print(fruit[i])
    i = i + 1
```

```
apple
banana
papaya
graps
mango
cherry
```

break : when break condition is true in loop then after successful executing break statement loop stop printing values.

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

```
1
2
3
```

continue: when condition for continue is true than that value skipped and againg loop will start the printing values.

```
i = 0
while i < 6 :
    i += 1
    if i == 3 :
        continue
    print(i)
```

```
1
2
4
5
6
```

pass: it means null statement(stands for do nothing with previous statement)

else : when loop condition is false then else condition execute and it worksame as else work with if keywords.

```
i = int(input("enter any number : "))
while i < 6 :
    print(i)
    i += 1
else:
    print(" i is greater than 6")
```

```
enter any number : 8
 i is greater than 6
```

2. for loop :

used to iterate through a sequesnce like list, tuple, string[iterables] with keywords (is,in,range)

```
fruits = ['banana', 'mango', 'apple', 'watermalon', 'graps']
for items in fruits:
    print(items)
```

```
banana
mango
apple
watermalon
graps
```

```
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
    for y in fruits:
        print(x, y)
```

```
red apple
red banana
red cherry
big apple
big banana
big cherry
tasty apple
tasty banana
tasty cherry
```

for with range:

used to generate sequence of number and it's print until 'range'

```
for i in range(8):
    print(i)
```

```
0
1
2
3
4
5
6
7
```

```
for i in range(1,11):
    print(i)
```

```
1
2
3
4
5
6
7
8
9
10
```

```
for i in range(1,11,2):
    print(i)
```

```
1
3
5
7
9
```

```
for i in range(11):
    print(i)

else:   #when condition will be false
    print("this is inside of 'for'")
```

```
0
1
2
3
4
5
6
7
8
9
10
this is inside of 'for'
```

```
for i in range(11):
    print(i)
    if i == 5 :
        break
else:
    print("this is inside of 'for'")      # 0 1 2 3 4 5

# else will execute when for loop will exit with condition
# not exit with break, else will run with suceess termination of for
```

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

```
for i in range(8):
    print(i)
    if(i==5):
        break
else:
    print("this is inside of 'for'")  #0 1 2 3 this is inside of 'for'
```

```
0
1
```

```
2
3
4
5
```

```
for i in range(11):
    if i == 5 or i == 7 :
        continue
    print(i)
```

```
0
1
2
3
4
6
8
9
10
```

for more example of 'for' loop :

```
num = int(input("enter number to print table : "))
i = 1
print("way 1")
for i in range(1,11):
    print(i * num)
print("way 2")
for i in range(1,11):
    print(num, " X " ,i, " = ",i*num)
print("way 3")
for i in range(1,11):
    print(f"{num} X {i} = {i*num}")
```

```
enter number to print table : 19
way 1
19
38
57
76
95
114
133
152
171
190
way 2
19  X  1  =  19
19  X  2  =  38
19  X  3  =  57
```

```
19  X  4  =  76
19  X  5  =  95
19  X  6  =  114
19  X  7  =  133
19  X  8  =  152
19  X  9  =  171
19  X  10  =  190
way 3
19 X 1 = 19
19 X 2 = 38
19 X 3 = 57
19 X 4 = 76
19 X 5 = 95
19 X 6 = 114
19 X 7 = 133
19 X 8 = 152
19 X 9 = 171
19 X 10 = 190
```

try to study str() and fstrings too f"string_char {values,express,calculation}string_char {values,express,calculation}string_char {values,express,calculation}" via online sources or in string chapter

excersize :

```
names = ['ram', 'mohan', 'sohan', 'sita', 'suresh', 'lakhan',
'sachin', 'harry']
print("list : ",names,"\ngreeting for all whose names starts with
's' :")
for name in names:
    if name.startswith('s'):
        print("good morning !",name)
```

```
list :  ['ram', 'mohan', 'sohan', 'sita', 'suresh', 'lakhan',
'sachin', 'harry']
greeting for all whose names starts with 's' :
good morning ! sohan
good morning ! sita
good morning ! suresh
good morning ! sachin
```

```
num = int(input("enter any number to check wether number is prime or
not : "))
prime = True
for i in range(2,num):
    if num % i == 0 and num != 2 :
        prime = False
        break

if prime:
```

```
    print(num," is prime number...")
else:
    print(num," is not prime number...")
```

```
enter any number to check wether number is prime or not : 153
153  is not prime number...
```

```
num = int(input("enter any number to get factorial : "))
fact = 1
for i in range(1,num+1):
    fact = fact *i
print(f"factorial of {num} is : {fact}")
```

```
enter any number to get factorial : 9
factorial of 9 is : 362880
```

```
num = int(input("enter range to get sum of first natural number : "))
sum = 0
i = 1
print("natural no. are : ")
while i < num+1 :
    sum = sum + i
    n = i
    print(n)
    i += 1
print(f"sum of first {i-1} natural no. is : {sum}")
```

```
enter range to get sum of first natural number : 5
natural no. are :
1
2
3
4
5
sum of first 5 natural no. is : 15
```

```
num = int(input("enter number to get table : "))
limit = 10
i = 0
for i in range(limit,i,-1):
    print(f"{num} X {i} = {num*i}")
```

```
enter number to get table : 3
3 X 10 = 30
3 X 9 = 27
3 X 8 = 24
3 X 7 = 21
3 X 6 = 18
3 X 5 = 15
3 X 4 = 12
3 X 3 = 9
```

```
3 X 2 = 6
3 X 1 = 3
```

let's try pattern code with python:

```
i = 3
for i in range(3) :
    print(" " * (3-i-1),end="")
    print("*" * (2*i+1),end="")
    print(" " * (3-i-1))
```

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

```
n = int(input("enter range : "))
i = n
for i in range(n):
    print(" " * (n-i-1),end = "")
    print("*" * (2*i+1),end = "")
    print(" " * (n-i-1))
```

```
enter range : 5
    *
   ***
  *****
 *******
*********
```

```
for i in range(4):
    print("*" * (i+1))
```

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

```
n = int(input("enter range : "))
for i in range(n) :
    print("*" * (i + 1))
```

```
enter range : 5
*
**
***
****
*****
```

```
for i in range(5):
    print("*" * (5-i))
```

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

```
n = int(input("enter range : "))
for i in range(n):
    print("*" * (n-i))
```

```
enter range : 7
*******
******
*****
****
***
**
*
```

Function & recursion:

FUNCTION:

group of statement or specific task you can use it and we can stop repeatation to write code and use it many times. A function is a block of code which only runs when it is called. You can pass data, known as parameters, into a function. A function can return data as a result. when a code gets bigger in size and complexity of it grows,it gets difficult to track specific statement so, a function can be reused by the programmer in a given program any number of times.

let's see of functions with using example...

```
# in simple way
marks1 = [45, 78, 86, 77]
p1 = ((sum(marks1) / 400) * 100)
marks2 = [75, 89, 92, 70]
p2 = ((sum(marks2) / 400) * 100)
print(p1, p2)

# using function
def percent(marks):
    return (sum(marks) / 400) * 100
p1 = percent(marks1)
p2 = percent(marks2)
print(p1," and ",p2)
```

```
---------------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
```

```
~\AppData\Local\Temp\ipykernel_17100\2025504056.py in <module>
      1 # in simple way
      2 marks1 = [45, 78, 86, 77]
----> 3 p1 = ((sum(marks1) / 400) * 100)
      4 marks2 = [75, 89, 92, 70]
      5 p2 = ((sum(marks2) / 400) * 100)

TypeError: 'int' object is not callable
```

```
def demo():
    print('you are welcome')
def demo1(name):
    print('have a good day !!',name)

demo()
demo1('ram')
```

```
you are welcome
have a good day !! ram
```

Type of functions:

function can return results or print directly but most of them, there are two type of functions:

1. built-in : already define in python's library
2. user-defined : defined by users

```
# EXAMPLE:
# built-in function:
mark = [10, 20, 30, 40]
print(sum(mark))    # here sum()  is built-in function

# user defined function:
def greet(name):
    print('good morning !!!,', name)
greet('raj')
```

```
100
good morning !!!, raj
```

```
def add(x,y,z):
    return x + y + z
a = add(45,85,89)
print(a)
```

```
219
```

```
def gret(name):
    gr = "hello " + name
```

```
    return gr
gret('Mr. R')
```

```
'hello Mr. R'
```

```
def add(x,y):
    add = x + y
    return add

def sub(x,y):
    sub = x - y
    return sub

a = add(45,15)
b = sub(26,11)
print("addition : ",a,"\nsubtraction : ",b)
```

```
addition :  60
subtraction :  15
```

Arguments in functions :

1. number of arguments:

there are any number of args can have be in function if you put there specific number then that would be necessary to give them number of args as input of data.

By default, a function must be called with the correct number of arguments. Meaning that if your function expects 2 arguments, you have to call the function with 2 arguments, not more, and not less.

```
def myfun(x,y):
    print(f"my full name is {x} {y}.")
myfun("raj","kumar")
```

```
my full name is raj kumar.
```

default :

we can have a value as default argument in functions, if we have specify name = 'stranger' in line

```
def greeting(name = 'stranger'):
    gr = "Good Morning !!!, " + name
    return gr

a = greeting()
```

```
b = greeting("raj")
print(a)
print(b)
```

```
Good Morning !!!, stranger
Good Morning !!!, raj
```

*args and **kwargs in function :

*args :

If you do not know the number of arguments that will be passed into your function, add a * before the parameter name in the function definition.

This way the function will receive an arguments in the form of tuple, and can access the items according to requirements.

The special syntax *args in function definitions in python is used to pass a variable number of arguments to a function. It is used to pass a non-key worded, variable-length argument list.

The syntax is to use the symbol * to take in a variable number of arguments; by convention, it is often used with the word args.

What *args allows you to do is take in more arguments than the number of formal arguments that you previously defined. With* args, any number of extra arguments can be tacked on to your current formal parameters (including zero extra arguments).

For example, we want to make a multiply function that takes any number of arguments and is able to multiply them all together. It can be done using *args.

Using the , *the variable that we associate with the* becomes an iterable meaning you can do things like iterate over it, run some higher-order functions such as map and filter, etc.

```
# Example:
# If the number of arguments is unknown, add a * before the parameter
name:
def myfun(*name):
    result = f"my full name is {name[0]}, {name[1]} {name[2]}"
    return result

x = myfun('rajendra', 'kumar','randa')
print(x)
```

```
my full name is rajendra, kumar randa
```

```
def name(x,*args):
    print(x)
    print(*args)
```

```
    for item in args:
        print(item)
x = 'this is good boy'
l = ['a','b','c','d','e']
name(x,*l)
name(x)
```

```
this is good boy
a b c d e
a
b
c
d
e
this is good boy
```

The phrase Keyword Arguments are often shortened to kwargs in Python documentations.

**kwargs:

```
def my_function(**kid):
  print("His last name is " + kid["lname"])

my_function(fname = "Tobias", lname = "Refsnes")
```

```
His last name is Refsnes
```

```
def myFun(**kwargs):
    for key, value in kwargs.items():
        print(f"{key} = {value}")

# Driver code
myFun(first='Geeks', mid='for', last='Geeks')
```

```
first = Geeks
mid = for
last = Geeks
```

```
def fun1(**kwargs):
    for key,value in kwargs.items():
        print(key,":",value)

dict = {'a': 'this is a','b': 'this is b','c': 'this is c','d': 'this
is d',}
fun1(**dict)
```

```
a : this is a
b : this is b
c : this is c
d : this is d
```

difference between* args and** kwargs is that *args is works for collection of values and* ***kwargs for dictionaries

```
def fun1(s,**kwargs):
    print(s)
    for key,value in kwargs.items():
        print(key,":",value)

str = 'this is string'
dict = {'a': 'this is a','b': 'this is b','c': 'this is c','d': 'this
is d',}
fun1(str,**dict)
```

```
this is string
a : this is a
b : this is b
c : this is c
d : this is d
```

```
def show(x,y,*a,**c):
    print(x)
    print(y)
    print(a)
    print(c)

p = 'this is python'
q = 'it is so simple'
l1 = [1, 2, 3, 4, 5]
l2 = [9, 8 , 7, 6]
dict1 = {0: 'this is 0','x': 'this is x','y': 'this is y','z': 'this
is z',}
dict2 = {0: 'this is 0','x': 'this is x','y': 'this is y','z': 'this
is z',}
show(p,q,l1,l2,dict1,dict2)

# only one * and ** parameter allowed
```

```
this is python
it is so simple
([1, 2, 3, 4, 5], [9, 8, 7, 6], {0: 'this is 0', 'x': 'this is x',
'y': 'this is y', 'z': 'this is z'}, {0: 'this is 0', 'x': 'this is
x', 'y': 'this is y', 'z': 'this is z'})
{}
```

Recursion:

the function called himself is known as recursion.

let's see that via example:

```
# factorial : 4! = 4 * 3 * 2 * 1 = 42
# n! = n! * (n-1) * .......3 * 2 * 1 || fact(n) = n * fact(n-1)
def fact(n):
    if n == 1 or n == 0:       # coz fact of 1 or 0 is always 1
        return 1
    else:
        return (n*fact(n-1))

x = fact(5)
print(x)
```

```
120
```

NOTE : be careful with using recursion working to ensure that the function doesn't call infinitly himself. recursion is sometimes the most direct way to code an algorithm

lambda function :

this function create using 'lanbda' keyword A lambda function is a small anonymous function. it takes any number of arguments, but can only have one expression ***Syntax***- lambda arguments : expression

```
# lambda function:
a = lambda num : num +1; print(a(5))
square = lambda x : x*x
sum = lambda x, y, z : x+y+z
x = 3
y = 2
print(square(x))
print(sum(x,y,1))
```

```
6
9
6
```

Why Use Lambda Functions?

The power of lambda is better shown when you use them as an anonymous function inside another function. Say you have a function definition that takes one argument, and that argument will be multiplied with an unknown number. why we use: if you wwant to create any function with one expression or need to return function as so you can use it that time and also can use it anywhere

Example:

```
x = lambda a : a + 10
print(x(5))
```

```
15
```

```
x = lambda a,b : a * b
x(2,4)
```

```
8
```

```
x = lambda a, b, c : a + b + c
print(x(5, 6, 2))
```

```
13
```

lambda function inside of function:

Example:

```
def myfunc(n):
    return lambda a : a * n

mydoubler = myfunc(2)

print(mydoubler(11))
```

```
22
```

```
def myfunc(n):
    return lambda a : a * n

mytripler = myfunc(3)

print(mytripler(11))
```

```
33
```

excersize : with function

```
def max_num(num1,num2,num3):
    if num1 > num2 and num1 > num3:
        return num1
    elif num2 > num1 and num2 >num3:
        return num2
    else:
        return num3
m = max_num(14,52,89)
print("maximum no is :" + str(m))    # way 1
print(f"maximum no is : {m}")    # way 2
```

```
maximum no is :89
maximum no is : 89
```

```
def cel(f):
    return ((f-32)*(5/9))
def fah(c):
    return (c*(9/5)+32)

a, b = 450,500
x = cel(a)
y = fah(b)
print(f"farentite value of {a} is {x} F")
print(f"celsious  value of {b} is {y} C")
```

```
farentite value of 450 is 232.22222222222223 F
celsious  value of 500 is 932.0 C
```

```
print("hello")
print("how")
print("are")
print("you ?")

print("hello ", end= "")
print("how ", end= "")
print("are ", end= "")
print("you ?", end= "")
```

```
hello
how
are
you ?
hello how are you ?
```

```
def sum_n(n):
    sum = 0
    sum = sum_n(n) +sum_n(n-1)
    return sum
sum_n(5)
```

find greatest among numbers using **if-else and** function

```
def maximum(a,b,c):
    if a>b:
        if a>c:
            return a
        else:
            return c
    else:
        if b>c:
            return b
        else:
```

```
            return c
m = maximum(9100,2000,300)
print(m)
```

```
9100
```

```
def pattern(n):
    for i in range(n):
        print("*" * (n-i))
n = int(input("enter range of pattern : "))
pattern(n)
```

```
enter range of pattern : 6
******
*****
****
***
**
*
```

```
def conv_inch(cm):
    inch = cm/2.54
    return inch
def conv_cm(inch):
    cm = 2.54*inch
    return cm
x = conv_inch(25)
y = conv_cm(9.84251968503937)
print(x)
print(y)
```

```
9.84251968503937
25.0
```

strip():used to remove unneccessary

```
print("demonstration of removing whitespace and stripping of string :
")

def rmv_n_strip(string,word):
    newstr = string.replace(word,"")
    return newstr.strip()
str = input("enter a string:\n")
n = input("enter a word to strip from string:\n")
a = rmv_n_strip(str,n)
print(a)
```

```
demonstration of removing whitespace and stripping of string :
enter a string:
   i am rajendra  randa   .
enter a word to strip from string:
```

```
i am rajendraranda .
```

fun with code

```
print("please enter two number using space :")
num1, num2 = input("Enter two numbers : ").split()

num1 = float(num1)
num2 = float(num2)

sum = num1 + num2
diff = num1 - num2
prod = num1 * num2
quotient = num1 / num2;
average = (num1 + num2) / 2;

print("The sum, difference, product, quotient, average of is:",sum,"
",diff," ",prod," ",quotient," ",average)

if(num1 > num2):
  print("The sum of two numbers is: ",sum)
elif(num2 > num1):
  print("The difference of two numbers is: ",diff)
else:
  print("The quotient and average of two numbers is: ",quotient,"
",average)
```

```
please enter two number using space :
Enter two numbers : 12 13
The sum, difference, product, quotient, average of is: 25.0   -1.0
156.0   0.9230769230769231   12.5
The difference of two numbers is:  -1.0
```

File Handling (I/P & O/P file) :

File handling is an important part of any web application. Python has several functions for creating, reading, updating, and deleting files.

open the file :

read txt file :

To open the file, use the built-in open() function. The open() function returns a file object, which has a read() method for reading the content of the file: r : read mode to read the file rt : read mode to read the txt file

```
# way 1: default
f = open('demo1.txt','r')
data = f.read()
print(data)
f.close()

# way2 :use file path
f = open('C://Users//rajen//Downloads//Data_Science//Python
Programming//demo1.txt','r')
data = f.read()
print(data)
f.close()

# way3 : using 'with' keyword
with open('demo1.txt') as f:
    data = f.read()
    print(data)
```

```
this is text file line number 0
this is text file line number 1
this is text file line number 2
this is text file line number 3
this is text file line number 4
this is text file line number 5
this is text file line number 6
this is text file line number 7
```

```
---------------------------------------------------------------------------
-----
FileNotFoundError                         Traceback (most recent call
last)
~\AppData\Local\Temp\ipykernel_11964\2684226151.py in <module>
      6
      7 # way2 :use file path
----> 8 f = open('C://Users//rajen//Downloads//Data_Science//Python
Programming//demo1.txt','r')
      9 data = f.read()
     10 print(data)

FileNotFoundError: [Errno 2] No such file or directory:
```

```
'C://Users//rajen//Downloads//Data_Science//Python
Programming//demo1.txt'
```

you can read content using read() and in open() you can file name or file name with path and extention and mode of open()

```
# with loop
f = open('dummy.txt')
data = f.read()
for x in data:
    print(x)
```

```
1
1
1
1


2
2
2
2


3
3
3
3


4
4
4
4


5
5
5
5


6
6
6
6
```

```
f = open("dummy.txt", "r")
print(f.readline())
print(f.readline())
```

```
1111

2222
```

```
f = open('demo1.txt','r')
f1 = open('demo1.txt')
data1 = f.read()
data2 = f1.read()
print(data1)
f.close()
f1.close()
```

```
data = open("demo1.txt")
print(data.readline())
x = data.readline()
print(x)
y = data.readline()
print(y)
z = data.readline()
print(z)
data.close()
```

```
this is text file line number 0

this is text file line number 1

this is text file line number 2

this is text file line number 3
```

write in the file:

```
# write in file
f = open('another.txt','w')
data = f.write('you are welcome')
f.close()
```

```
# read that content
f = open('another.txt','r')
data = f.read()
print(data)
f.close()
```

```
you are welcome
```

```
# if you use write() function again then it'll rewrite content
f = open("another.txt","w")
data = f.write("this is text file where we'll write using python file
```

```
handling write() function.")
f.close()
f = open("another.txt","r")
data = f.read()
print(data)
```

```
this is text file where we'll write using python file handling write()
function.
```

```
f = open('another.txt','a')
data = f.write('this is content of second times.')
f.close()
f = open('another.txt', 'r')
data = f.read()
print(data)
```

```
this is text file where we'll write using python file handling write()
function.this is content of second times.
```

```
with open('demo.txt','w') as f:
    f.write('this is style 2.')
    f.close()
with open('demo.txt') as f:
    print(f.read())
    f.close()
```

```
this is style 2.
```

we can understand it in next chapter.

file handling with "with" keywords :

```
# create txt file
with open("with.txt","r") as f:
    data = f.read()
    f.close()

# write txt file
with open("with.txt","w") as f:
    data = f.write('this is text file.')
    print(data)
    f.close()

# read txt file
with open("with.txt","r") as f:
    data = f.read()
    print(data)
    f.close()

# write data in already written file
with open('with.txt','w') as f:
    data = f.write('this li line 2, ')
```

```
    data = f.write('this is line 3')
    data = f.close

with open("with.txt","r") as f:
    data = f.read()
    print(data)
    f.close()
```

```
---------------------------------------------------------------------------
-----
FileNotFoundError                         Traceback (most recent call
last)
~\AppData\Local\Temp\ipykernel_12680\2641714934.py in <module>
      1 # create txt file
----> 2 with open("with.txt","r") as f:
      3     data = f.read()
      4     f.close()
      5

FileNotFoundError: [Errno 2] No such file or directory: 'with.txt'
```

```
# find words in string
word = input("enter word to check weather is exist in poem 'the
star' : ")
with open('poem.txt','r') as f:
    data = f.read()
    if word in data:
        print("word matched...")
    else:
        print("word doesn't exist...")
```

```
enter word to check weather is exist in poem 'the star' : twinkle
word matched...
```

```
# generate table 2-20 and write it in diff diff text files
for i in range(2,21):
    with open(f"table of {i}.txt",'w') as f:
        for j in range(1,11):
            f.write(f"{i} X {j} = {i*j}")
            if j!= 10:
                f.write('\n')

# check text files in folders

# a file contain word "donkey" multiple time,WAP toreplace it with
###### by updating some files
with open('sample.txt','r') as f:
    data = f.read()
    data = data.replace('donkey', '########')
    with open('sample.txt','w') as f:
        data = f.write(data)
```

```
with open('sample.txt','r') as f:
    data = f.read()
    print(data)
```

NOTE : file handling using keywords 'with' can work without 'close()'

```
# hide matched words of collection in text file

words = ['donkey','monkey','kaddu','budddu','mote']
with open('sample.txt','r') as f:
  data = f.read()

for word in words:
  data = data.replace(word,'#######')
  with open('sample.txt','w') as f:
    f.write(data)

def game(x):
    return x

# x = int(input("enter your score : "))
score = game(200)

with open('hiscore.txt','r') as f:
    newscore = f.read()

if int(newscore) < score :
    with open('hiscore.txt','w') as f:
        data = f.write(score)
elif newscore == '':
    with open('hiscore.txt','w') as f:
        data = f.write(score)
```

```
---------------------------------------------------------------------------
-----
FileNotFoundError                         Traceback (most recent call
last)
~\AppData\Local\Temp\ipykernel_12680\2156176863.py in <module>
      5 score = game(200)
      6
----> 7 with open('hiscore.txt','r') as f:
      8     newscore = f.read()
      9

FileNotFoundError: [Errno 2] No such file or directory: 'hiscore.txt'
```

```
with open('log.txt') as f:
    data = f.read().lower()
if 'python' in data:
    print('string present...')
else:
    print("string doesn't present...")
```

```
string present...
```

```
data = True
i = 1
with open('log.txt') as f:
    while data:
        data = f.readline()
        if 'python' in data.lower():
            print(data)
            print(f'\nstring present in file on line no,: {i}')
            i = i+1
```

```
03/22 08:51:06 INFO   :....mailbox_register: mailbox allocated for
rsvp python has stopped working.....


string present in file on line no,: 1
03/22 08:51:11 INFO   :.....mailbox_register: mailbox allocated for
route        python has stopped working.....


string present in file on line no,: 2
03/22 08:52:50 TRACE  :......rsvp_event_establishSession: local node
will send   python has stopped working.....


string present in file on line no,: 3
03/22 08:52:50 TRACE  :........router_forward_getOI:         route
handle:   7f5251c8 python has stopped working...


string present in file on line no,: 4
03/22 08:52:50 TRACE  :.......event_establishSessionSend: found
outgoing if=9.67.116.98 through python has stopped working.....


string present in file on line no,: 5
03/22 08:52:50 INFO   :.......init_policyAPI: RegisterWithPolicyAPI:
Entering   python has stopped working.....

string present in file on line no,: 6
```

make a copy of text file

```
with open('sample.txt') as f:
    data = f.read()
with open('copy.txt','w') as f:
    data = f.write(data)
```

```
file1 = 'sample.txt'
file2 = 'copy.txt'
with open('copy.txt','r') as f:
    data1 = f.read()
with open('sample.txt','r') as f:
    data2 = f.read()
if data1 == data2:
    print('files are identical...')
else:
    print("files aren't identical...")
```

```
files are identical...
```

rename text file:

```
import os
oldname = 'font.txt'
newname = 'rename.txt'

with open(oldname,'r') as f:
    data = f.read()
with open(newname,'w') as f:
    data = f.write(data)
os.remove(oldname)
```

delete the file

To delete a file, you must import the OS module, and run its os.remove() function

```
import os
os.remove('delete.txt')
```

check if file exist then delete if not then show msg

```
import os
if os.path.exists('delete.txt'):
    os.remove('delete.txt')
    print("file deleted...")
else:
    print("file doesn't exist...")
```

```
file doesn't exist...
```

delete folder

```
import os
os.rmdir('delete')
```