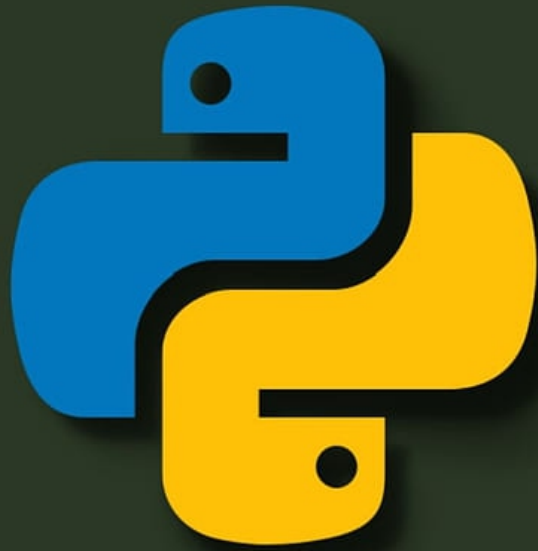


ISSN/ISBN : 2320-8821



**BASIC
OF
PYTHON PROGRAMMING**

Author : Mr. Vaibhav Narkhede



PREFACE

The ever-evolving landscape of “**BASIC OF PYTHON PROGRAMMING**” the need for a comprehensive and practical guide has never been more pressing. As I embarked on the journey to write this book, my goal was to provide technical readers with not only a deep understanding of the programming concept but also to understand the practical point of view while writing a code that can be applied in real-world scenarios.

This book is a culmination of years of coding and orientation, hands-on experience, and the valuable lessons learned along the way. It is intended to serve as both a reference for seasoned professionals and a learning tool for those new to the field. Each chapter has been carefully crafted to build on the previous ones, ensuring a logical progression of ideas and concepts. Throughout this book, you will find examples, case studies, and thought-provoking questions designed to challenge your understanding and encourage further exploration.

It is my sincere hope that this book will inspire you, provide clarity on complex topics, and serve as a trusted resource in your personal and professional development.

While writing this book I personally thank to my parents who always support me, my wife and child who has given me confidence to complete this book.

I also thank to my respected guide Dr. P. M. Jawandhiya Sir without which this book is not possible.

And last but not the least my friends who always encourage me to gain new knowledge in my field

- Mr. V. P. Narkhede

CHAPTER 1

INSTALLATION

Installation:

1]python. org →to download python.

2] Download install vs. code for your machine & install.

3] Extensions-

①)code runner

②Python (microsoft)

configure -

file> preferences > setting>run in terminal C check the checkbox).

4]In terminal or command prompt

python - -version → to check the version of python

pip→ info. about pip.installed.

5] shift + right click in folder (root directory)

→open in powershell→

Type code.←| =>to open vs code.

6]Modules & pip (python packet Manager)

(1)Right click on start menu→ powershell.

(2) To install module, in terminal/powershell

pip install <module-name>

(3) exit→ to exit powershell.

*** writing a program:**

- ① creating project file.
- ② creating python file to print "Hello World"

***calculations in powershell:**

>>>3+5 ←┘ =>8 (=> output)

***comments:**

- ① Single line comment starts with #
- ② Multiple line comment : " " "
(single|double quot") multiple lines" " "

***Use of (end=" ") in print:**

print ("Hi",end=" ")

Print ("vaibhav") =>Hi,vaibhav
 └──────────┘
 Single line display

***Escape Sequences:**

-insert characters that are illegal in a string with backslash(\).

\n=newline character

\+=one tab

\'=single quote (apostrophe')

\\=backslash

\b=backspace

Exercise-

(1) create the mountain pattern with the help of escape sequence.

***Variables :-**

*** Variable is created when u assign value to it.**

Ex.

```
x=51
```

```
y = "John"
```

```
print(x)
```

```
print (y).
```

*** Don't need to be set any type. To get the type of variable.**

```
print (type (x)) ←
```

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

```
↳ x is of int type.
```

***String Variables** → ① double quotes.

② single quotes.

```
X="vaibhav" or x='vaibhav'
```

***Names are case sensitive-**

```
[a]=4 (diff.) [A]="xyz".
```

****Variable name should not start with No.**

***Multi-words Names:-**

i) camel case-myvariableName

ii) Pascal case-MyVariableName

iii) Snake case-my_variable_name

*** Many values assignment:**

1) a, b, c = "apple", "banana", "Greepes"

print (a)

print (b)

print (c)

2) > x = y = z = "buldana"

print (x)

print (y)

print (z).

*** Operations Operations with variables:**

var 1 = "Hello world"

var 2 = 4

var 3 = 36.7

print (var 2 + var 3) => o/p 40.7

print (var 1+ var 2) => error

but,

var 5 = "How r u?"

Print (Var1+ var5) => o/p→No error

concatenation of strings,

var 6="32"

print (var1+ var 6) => o/p⇒ Hello world 32

*** Type Casting:-**

var 1 = "54"

var 2 = "46"

print (vael + vac2) => 5446

```
print (int (vart) + int (vask)) => 100
```

Funⁿ-

```
str( )
```

```
int( )
```

```
float ( )
```

***To print multiple times-**

```
print (10 * "Hello world")
```

```
"Hello world In".
```

```
print (10x int (var z) + int (var 2))
```

```
print (10 * str (int (var 1) + int (vari)))
```

```
=>?
```

***Quiz**

***Exercise**

***project.**

*** Talking i/ps from uses:**

```
print ("Enter your number")
```

```
inpnum = input( )
```

in the form of string only.

if print (inpnum + 10) => Error string can't be added with int 10.

```
print (str(inpnum)+10) => o/p
```

*** Quiz:**

simple program to add two number by taking i/p From user.

•Input can be taken like this.

```
∴ n1 = input("Enter your number")
```


CHAPTER 2

DATATYPES

*** Data Types-**

str, int, float, complex, list, tuple, range, dict, set, frozenset, bool, bytes, bytearray, memorview.

*** Numbers-**

int, x=7

Float, y=2.8

Complex, z=1j

*** Random Numbers:**

```
import random
print [random. nandrange (1, 10)]
```

*** Python strings-**

```
print("Hello world!")
      string /Text.
```

*** Multiline strings-**

```
a = " " " Lorem ipsum...
.....
..... " " "
print (a). or three" " .....
..... " "
```

*** Strings are arrays-**

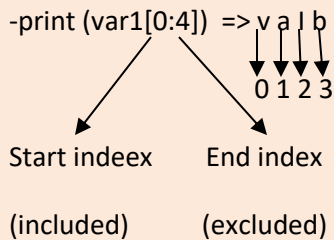
```
var i = "Vaibhav is a good boy"
      | | | | | | |
      v v v v v v v
index=> 0 1 2 3 4 5 6 7
      ∴ print (var1[0])v.
      print(var1[1])⇒a. &soon.
```

*** string length-**

print (length (var1)) => Total no. of characters in string. including space=>21

*** slicing:**

- can return a range of characters by using slice syntax.



For var 1

Length=> 21

index => 0→20.

-print (var1 [0:20]) =>?(will not print total string)

-print (var1 [0: 21]) => ? (full string)

Beyond length.

- print (var1 [78])=> Error

but

-print (var1[0:78]) => No Error

•Advanced slicing| Extended slice.

Print (var1 [0:5]) =>vaibh

Print (var1 [0:5:2]) =>vih

↳ **Skipping index.**

Print (var1 [0:]) =>full string.

↳ **length**

print (var1 [:5]) =>vaibh

↳ **consider to be 0**

print (var 1 [:]) =>full string.

Print (var1[: :]) =>full string

↳ Skip index by default taken as 1

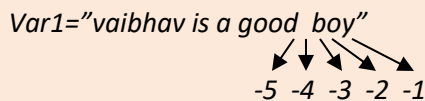
Print (var1[0:21:1]) =>full string.

Print 9var1[0:21:2]) =>skip one char.

Print(var1[: : 3])=>?

Print(var1[: : 555]) => v

***Negative Indexing:**



Print (var1 [-3])=>

Print (var1 [-3:-1]) =>bo

Print (var1 [: :-1]) =>reverse the string.

Print (var1 [: :-2])

-1 ① reverse the string

-1 ② skip one char.

***Functions on string:**

1)print [var1 . isalnum ()] =>>false

As spaces are there in string.

If we remove spaces =>>false

=>Boolean type

2)print (var1 . ends with ["boy"]) =>True

3)print (var1 . want ["b"]) =>1

4)print (var1 . capitalize []) =>first letter capitalize

5)print (var . find ["is"]) =>8

↳ From 8th index, is is string.

6)print (var1 . lower []) => lowercase string.

7)print (var1 . upper []) =>uppercase.

8)print (var1 . replace ["is", "are"]) =>?

9)print (var1 . strip []) =>remove white space before or after the actual text.(Not in the middle)

10)print (var1 . split []) => splits the string into substrings if it finds instances of separator.

(, □[single space])

11)Format () => combine strings & numbers.

① age=37

Text= "Myname is siya,I am"+ age

Print (text)

② age=37

Text="My name is siya,I am"=age

Print (text . format [age])

*****Format method takes unlimited number of arguments & placed into the respective placeholder.**

•We can use index numbers {0} to placed values in placeholders.

Quantity=4

Item=100

Price=500.50

myorder=" I have to pay {2} item {1}."

Print (myorder.format [quantity,itemno,price]).

***BOOLEAN:**

-Often need to know if expression is true or false.

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

-Bool () evaluate any value.(True/False print bool [15])

-Any string,list,tuple,set,dicf are True except empty.

-Any number is true,except 0.

-(),{ },[],",",0,None =>False.

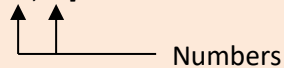
***List:**

-List is collection of items either string,number etc.

```
Items=['parlect', 'oil', 'deodorant', 'flavor']
```

```
Print (items) =>List.
```

```
Items =["parlect","oil","deodorant","flower",57,88]
```



```
Print (items)
```

To access members of list.

```
Print (items [0]) =>parlect
```

```
Print (items [1] =>oil
```

```
Print (items [7] => error.(out of range)
```

Lets,explore list methods.

```
Numbers=[2,7,9,11,3]
```

1)Numbers . sort () => to sort original list.

```
Print(numbers).
```

2)Numbers .reverse () . =>to reverse the list.

```
Print (numbers).
```

3)Numbers . insert (2,30) =>to insert any.item,without replacing existing values.



```
Print (numbers).
```

4)Numbers . oppned (40) =>To add an item tp end pf the list. Print (numbers)

5)List=["apple","banana"]

Numbers. Extend (List 1) =>To oppend elements from another list to current.

Print (numbers).

6)Numbers. Remove ("apple") =>remove specific item.

Print (numbers).

7)Numbers. Pop (1) =>remove specific item from index.

Print (numbers).

8)numbers. Clear () =>emptier the list.

Print (numbers).

9)len()=> to get length of list.

10)max ()=>maximum no.in list.

Min()=>minimum no. in list.

Index ()=>returns index of value.

***List slicing:**

Num=[2,7,9,11,3]

Print (num [1]) =>7

Print (num [0:5] =>full list

Print (num [0:4]) =>?

Print (num) => same list original

Print (num [: : 2] =>?

For negative indexing,

do not taken sl <-1 . most pwbably.

***Tuple:**

Mutable ->can change->list.

Immutable->can't change->tuple.

```
List 1= [2,3,4]
```

```
List [4]=8
```

```
Print (list 1)=> [2,3,4, ,8]
```

```
tup1= (2,3,4)
```

```
tup1 [4]=8
```

```
Print (tup1) => Error
```

•List & tuple allows duplicate values.

```
tup1=(1)
```

```
print (tup1)=>1=>tuple not created.
```

To create tuple with only one element add comma.

```
Tup1=(1,)
```

```
Print (tup1)=>(1,)=>tuple created.
```

•Slicing is done with tuple also like list.

•To change tuple values,or add any item,or remove|update the values;

With tuple,you can't but there is an idea.

① First convert tuple->list.

② Update the list.

③ Convert list->tuple.

```
X=("banana","apple","mango")
```

```
Y=list(x)
```

```
Y[1]="dragonfruit"
```

```
X=tuple(y)
```

```
Print (x) =>tuple("banana","dragonfruit","mango")
```


***Methods:**

1)count ()->Returns no.of times specified value occurs in tuple.

```
tup1=(1,3,7,5,5,6,5,7)
```

```
x=tup1.count(5).
```

```
Print(x)=>3.
```

2)Index ()->searches a tuple for specified value & returns position

```
X=(1,3,7,5)
```

```
Y=(x.index(7))
```

```
Print (y)=>2
```

***Dictionary:**

-used to store data values in key:value pairs.

-ordered,changeable,not allowing duplicates.

Casse-sensitive (care).

```
d1={ }=>Empty dict.
```

```
Print (type (d1)) => <class,'dict'>
```

```
d2={ "Vaibhav":"Bhendi",
```

```
    "advik": "drumstick",
```

```
    "Priyanka",:"fish",
```

```
    "Payal"::"roti"
```

```
}
```

```
Print(d2)=>prints dict.d2
```

```
Print(d2["advik"])=>drumstick
```

```
Print(d2{"payal"})=>roti
```

```
d2={"vaibhav" : "bhendi",
```

```
    "advik" : "drumstick",
```

```
"payal" : "roti",
```

```
"shubham" : {
```

```
    "B" : "Maggie",
```

```
    "L" : "roti",
```

```
    "D" : "jamun"
```

```
}}
```

Print (d2["shubham"])=>prints dict of shubham.

Print (d2 ["shubham"] ["B"]=>Maggie.

-you can nest list,tuple,dict in dict.

***To add items:**

```
d2["yash"]="junkfood"
```

```
print (d2)=>In dict,yash is get added at last.
```

-key can be string or number(integer)

***To remove items from dict:**

```
1)del d2 ["yash"]=>delete item from dict.
```

***Functions:**

```
:.d3=d2
```

```
del d3 ["vaibhav"]=>deletes from both d3&d2
```

-doesn't mean copy is created.

To create copy use-

```
d3=d2.copy c=>produces copy.
```

```
Print (d3)
```

-print (d2.get ("vaibhav"))=>Bhendi.

-print (d2.update({"leena" : "chocos"}))=>done.

So run,

`d2.update ()->first.`

`then print (d2)=>leena is added.`

`-print (d2.keys ())=>print keys.`

`-print (d2.items) ())=>print items.`

`-print (len (d2))=>no.of items in dict.`

`-print (d2.acces ())=>return list of all values in dict.`

`-d2.pop("payal")=.removes item with specified key name.`

`Print(d2)`

`-d2.clear ()`

`Print=>clears the dict.`

Exercise-

Create a dictionary,take i/p from user,& return the meaning of the word from dictionary.

****Dictionaries creation // nd way:***

`dict`

`user=name='vaibhav',age=34`

`print(user)`

****How to access the items in dict:***

`Print (user[o])=>Error.as no indexing in dict.`

So,

`Print (user['name'])=>vaibhav.`

***Method:**

1)Fromkeys-

If we want to create `d={'name','Age','height':'unknown'}`

Like this so we can fromkeys()

① `d=dict.fromkeys(['name','Age','height'],'unknown')`

Print(d).

② `d=dict.fromkeys("abc","unknown")`.

③ `d=dict.fromkeys(range(1,11),'unknown')`.

2)Get()-

`User={'name':'vaibhav','age':34}`

Print (user.get("fav",'not found!'))

=>not found!

No duplicate are allows in dict

`User={'name':'vaibhav',, 'age':34,'age':24}`

Print (usee)

=>{'name':'vaibhav','age':24}

Second value overwrites first value.

***Sets:**

-stored multiple unique items in a single variable.

-Unordered & non indexed,immutable,noduplicates allowed.

`S1={'orio',"parle G","mac"},`

Print (s1)=>prints set.

Print (type (s1))=><class,'set'>

-set can have strings,integers,orboolean values.

-set can't have list,dict.

```
Set1={"abc",34,True,40}.
```

```
Print (set1)=>prints set.
```

S=set() // set constructor.

Prints(s)=>creates set.

Easy tp construct set from list.

```
S=set ([1,2,3,4])
```

```
Print (s)=>{1,2,3,4}
```

```
l=[3,4,5,7]
```

```
S1=set(l)
```

```
Print(s1)=>{3,4,5,7}
```

***To add items in set:**

```
S=set( ).
```

- s.add(1)=>adds 1 to set.

```
s.add(2)=>adds 2 to set.
```

```
Print(s) =>{1,2}.
```

- s.remove (2)=>to remove 2from set.

```
S1={1,2,3}
```

- print(s.union (s1)=>?{1,2,3}

```
Print (s,s1)=.?
```

- print (s.intrsection (s1))

```
=>{1,2}
```

```
Print(s,s1)
```

- print (len (s))=>to know length of set.

- `print (type (s))=>`to know data type.

- `min (),max ()`.

- `s.update(s1)=>`to add items from another set into current set.

`Print (s)=>`{1,2,3}

- `s.discard(3)=>`to remove an item from set.


- `del s->`to delete the set.

`Print (s)=>`none | error.


- `s2=s.symmetric_difference(s1)`

`=>`Return set that contains all items from both sets,except that are present in both.

- `z=s1.isdisjoint(s2)`

`Print (z)`  Returns true if no items in set s1 is present in s2 .


- `z=s1.issubset (s2)`

`Print(z)`  Return true if all items set s1 are present in s2.

- `x={"a","b","c","d","e"}`

`Y={"a","b"}`

`Z=x.issuperset(y)`

`Print(z)`  Returns true if all items set y are present in x.

- `s1=s.copy ()->`To make a copy of set.

`Print (s1)`

- `union-set=s1|s2`

- `inte-set=s1 & s2`

***Frozenset ():**

->Frozenset () is an inbuilt function in python which takes an iterable object as input and makes them immutable.

->It freezes the iterable objects and makes them unchangeable.

```
#tuple of numbers
C=(1,2,3,4,5)

#converting it into frozenset

fnum=fmzenset (c)

print (fnum) =>frozenset({1,2,3,4,5})

print (type (fnum) )=><class,'frozenset'>
```

•As frozenset object are immutable,they are mainly used as key in dictionary or elements of other sets.

***If-----else-----:-/conditional statements:**

1)If----elif----else:-(elif=else if)

```
Var1=7
```

```
Var2=56
```

```
Var3=int (input ( ) )
```

```
If var3>var2: #pit brackets if necessary.
```

```
Print("Greater")
```

```
else:
```

```
print ("lesser")
```

```
⇒ 7 ←↓
```

```
Lesser
```

```
56 ←↓
```

```
Lesser (wrong one)
```

If var3 > var2:

Print("Greater")

If var3 = var2:

Print ("Equal")

Else:

Print ("lesser").

Here everytime second if is get checked which increase time of exeaction.so use elif offer first if.

Elif (var3 = var2):

Print ("Equal")

***In operator (keyword):**

-To check the item present in list ,set,tuple or dict, we use in operator.

List 1=[2,3,⑤.7]

If 5 in list 1:

Print ("yes,it is present")

=> Yes,it is present.

Print (5 in list 1) => Returns true if 5 is in list 1.

***Not in keyword:**

If 9 not in list1:

Print ("no,it isn't present ")

=> No , it isn't present.

Print (9 not in list 1)=> Returns true.if 9 is not present in list 1.

Exercise-

1) Take input from user as age.

2) Check if age >18 then display you can drive or else not.

3) If age=18, then ask the person to come physically @office for decision.

conditions:-

① $a = b$

② $a \neq b$

③ $a < b$, $a \leq b$

$a > b$, $a \geq b$

***Loops:**

•For loops:-

-Used to iterate over a sequence (list, tuple, dict, set or string).

(list, tuple, dict, set or string)

List 1= ["bouborn", "orio", "crackjack", "parleG", "Monaco"]

Print= (list1 [0], list1 [1], list1 [2], list1 [3], list1 [4]) => To print all items of list.

For item in list1:

Print (item)=>?

Llist=> [["Advik", "Bouborn"],

["payal", "orio"]

["priyanka", "crackjack"],

["nishu", "parleG"],

["mani", "manaco"]]

For item,name in list1:

Print (name) =>?

Print inner list.

For name,item in list1: print (name,item)=> prints name & item together.

Dict 1=dict (list 1)=>Converts list into dictionary.

->for name,item in dict1:

Print (name,item)=>error.

->for name,item in dict1. Items ():

Print (name,item)=>o/p.

->for name in dict1:

Print (name) =>only keys are printed.

Exercise-

1)create one arbitrary list.

2)print only those numbers which >6.

Soln-

Items = ["A", "B", 5, 3, 3, 4, 7, 21, 19, 2, 23]

For item in items:

If str (items).isnumeric () and item > 6:

Print (item).

****While loop:***

i=0

while (i < 10):

print (i)

1)If executes,then continuous loop will run so.

2)Insert some updation code after print.

i = i + 1

=> this will print nos.from 0 to 9.

Next page->

0

1

2

:

:

9

CHAPTER 3

BREAK STATEMENT

***Break statement:**

-To stop the loop before it has looped through all items.

```
i=0
```

```
while (i < 10):
```

```
    print (i + 1, end = " ")
```

```
    i = i+1
```

```
⇒ 1 2 3 4 5 6 7 8 9
```

```
i=0
```

```
while (True):
```

```
    print (i + 1, end = " ")
```

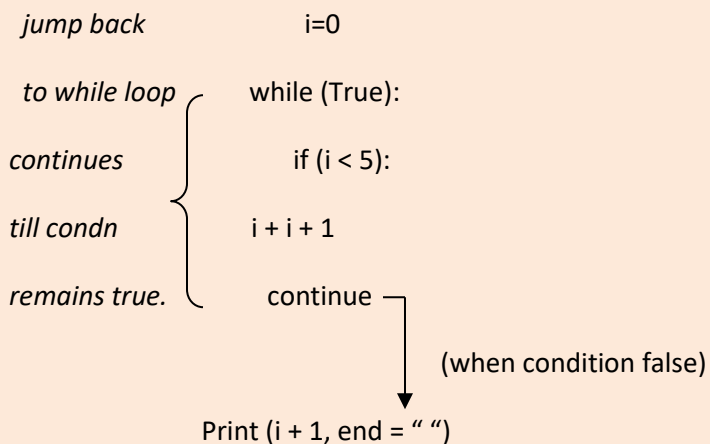
```
    if (i == 8):
```

```
        break # stop the loop.
```

```
    i = i + 1
```

```
## (control will jump here,when i=28)
```

*control will be taken off from loop & continue with next line of code in program.



If (i = 8):

Break

i = i + 1

#When (i < 5) condition remains false, then next line is get executed.

Exercise-

1) Take i/p from user.

2) If $i/p < 100$, then ask for another i/p. & loop continues.

3) If $i/p > 100$, then stop the loop. and congratulate him.

Exercise-

Guess the numbers.

1) $n = 18$ (any value).

2) Take i/p from user.

3) If $i/p < 18$, print (no is less)

$i/p > 18$, print (no is greater)

4) If $i/p = 18$, prints (congratulate).

5) Loop should continue until no of guesses = 9.

6) Print no. of guesses left.

7) If guess = 9, Game over.

CHAPTER OPERATORS

***Operators:**

-Don't use name of module on a name of file.

-Used to perform operations on variables and values.

1)Arithmetic Operators:

Print ("5+ 6 is", 5+6)=> 5+6 is 11

Print ("5-6 is" ,5-6)

Print ("5*6 is",5*6)

Print ("5/6 is", 5/6)

Print ("5/16 is", 5/16)=>floor division.

=>gives floor int.

Print ("5**3 is",5**3)=>5**3 is 125.

Print ("5%5 is", 5%5)=>5%5 is 0.

2)Assignment Operator:

x=5

Print(x) =>5

x=x+7 or x+=7

print(x) =>12

:x -=7

Print(x) =>5

x* =5 means x + x*5

print(x) =>25

x/=5 means x= x/5

print(x) =>5

$x\%=5$ means $x = x\%5$

print(x)=>0

x=5

$\therefore x11=3$ means $x=x113$

Print (x)=>1

$X^{**}=3$ means $x=x^{**}3$

Print (x) =>125

$X\&=3$ means $x=x\&3$

Print (x) =>1

101

011

001=>1

$x|=3$ mean $x=x|3$

Print (x) =>7

101

110

111=>7

$x^{\wedge}=3$ or $x=x^{\wedge}3$ (XOR)

print (x) =>6

101

^ 011

110=>6

$x>>=3$ or $x=x >>3$ (right shift)

print (x) =>0 (00001)

x

$x \gg 3$ or $x \ll 3$ (left shift)

print (x) => 40 (10100)

3) Comparison Operators:

i=8

print (i == 5) => false

print (i != 5) => true

print (i < 5) => false

print (i > 5) => true

print (i <= 5) => false

print (i >= 5) => true

4) Logical Operators:

a=True

b=False

print (a and b) => false

print (a or b) => true

print (not (a or b)) => false

reverse the result

5) Identity Operations:

-Used to compare the objects.

Print (5 is 5) => true

Print (5 is not 7) => true

Print (5 is not 5) => false

6)Membership Operators:

-Used to test if a sequence/item is present in object.

```
List =[2,3,4,5]
```

```
Print (5 in list) =>true
```

```
print (325 in list )=>false
```

```
print (325 not in list )=>true
```

7) Bitwise Operators:

-Used with binary numbers.

0-00

1-01

2-10

3-11

```
Print (1&2)=>0
```

```
Print (2|3)=>3
```

```
Print ( 1 ^ 2)=>3
```

```
Print (~(1&2) )=>1
```

```
Print (1<<2) =>4
```

```
Print (1>>2) =>0
```

***Short hand if_else Notation:**

****One liner notation-***

```
a=int (input ("enter a:\n"))
```

```
b=int (input ("enter b:\n"))
```

```
if a>b : print ("A is grater than B")
```

```
print ("A is grater than B") if (a>b)
    else print ("A is smaller than B")
```

***Functions:-**

-Function is a black of code which runs when called.

-can pass data, as parameters.

-can return data as a result.

****Creating a function:***

•Type of functions-

① Built in functions->predefined.

② User defined functions.

```
a=7
```

```
b=8
```

```
c=sum( a,b )=>Buit in function.
```

```
Print (c)
```

#press control on sum to open built in functions or doctype.

CHAPTER 5

FUNCTIONS

***Creating a function:**

```
def function 1 ( ):
    print ( "this is user's function")
function1 ( ) =>To call a function.
```

***Arguments/parameters:**

-Information can be passed into functions as arguments.

```
def func 1(a,b):
    print ("Addition of a & b is",a+b)
func1 (7,8)=>"Addition of a&b is 15
```

if we write,

```
print (func1 (7,8) )=>None this happens can function returns value.
```

If we want to store the value of addition in some other variable.

```
def function2(a,b):
    arg=(a+b)/2
    print (arg)
function2(5,7)=>6.0
```

if,

```
v=function2 (5,7)
```

```
print(v) =>None is returned. So if in function2,we include one line after print, return arg
```

then execute,

```
variable v will printed with value
```

```
=>6.0
```

Important properly of function. **code Rewability**

*** Docstrings:**

-information of function.

```
def func1 (a, b):
```

```
    """This is a function which will calculate average of two numbers"""
```

```
        Avg= (a+b)/2
```

```
    return avg
```

```
V= func1(5,7)
```

```
print (V) ⇒ 6.0-
```

```
print (func1.__ doc__)
```

```
    ⇒returns docstring of func1.
```