4.logical operators:

(a)logical AND:

A	B	A and B
0	0	0
1	0	0
0	1	0
1	1	1

Example:

```
>>> a=45>12 and 56>23

>>> print(a)

True

>>> a=34>90 and 56>34

>>> print(a)

False
```

(b) logical OR:

A	B	A or B
0	0	0
1	0	1
0	1	1
1	1	1

Example:

```
>>> 23>90 or 45>12

True

>>> 67>12 or 6>23
```

True

>>>

(c)logical NOT:

A	notA
0	1
1	0

Example:

>>> a=not(0)

>>> print(a)

True

>>> a=not(1)

>>> print(a)

False

(5)membership operators:

These operators test whether a value is a member of a sequence. The sequence may be a list, a string or a tuple.

in: it checks if a value is a member of a sequence

Example:

>>> names=["sushma","sowmya","anil","mani"]

>>> "anil" in names

True

>>> "chanti" in names

False

not in: it checks if a value is not a member of sequence.

Example:

>>> list=["sushma","mani","anil"]

>>> "mani" not in list

False

>>> "chanti" not in list

True

6.identity operators:

is: returns true if both variables are the same object.

Is not: returns true if both variables are not the same object.

Example:

>>> x=5

>>> type(x) is int

True

>>> x=90.345

>>> type(x) is int

False

>>> x=5

>>> type(x) is not float

True

>>> y=3.45

>>> type(y) is not float

False

Bitwise operators:

1.bitwise AND:

It returns 1 if both the bits are 1, otherwise zero.

Example:

>>> 10&7

2

Explanation:

A=10➔1010(binary)

B=7 ➔ 0111(binary)

A&B= 0 010

= 2(decimal)

2. bitwise OR:

It returns 1 if any of the bits 1, if both the bits are zero, then it returns zero.

Example:

>>> 10|7

15

Explanation:

A=10➔1010

B=7 ➔ 0111

A|B = 1111

=15(decimal)

3.bitwise XOR(^):

A	B	A^B
0	0	0
1	0	1
0	1	1
1	1	0

Example:

```
>>> 10^7
13
>>>
```

Explanation:

A=10➔1010

B=7 ➔0111

A^B= 1101

=13(decimal)

4.bitwise one compliment(~):

Ones complement of a number ‘A’ is equal to –(A+1)

Example:

```
>>> ~10
-11
>>>
```

Explanation:

A=10➔1010

~A=~1010

=-(1010+1)

=-(1011)

=-11

5.sift left operator(<<):

It shifts the left operand bits towards the left side for the given number of times in the right operand. In simple terms , the binary number is appended with 0's at the end.

Example:

```
>>> 10<<2
40
>>>
```

Explanation:

A=10➔1010

A<<2=1010<<2

=101000

=40(decimal)

6.bitwise right shift(>>):

It is exactly opposite of the left shift operator. Then left side operand bits are move towards the right side for the given number of times. In simple terms, the right side bits are removed.

Example:

>>> 10>>2

2

>>>

Explanation:

A=10➔1010

A>>2=1010>>2

=10

=2(decimal)

Constants in python:

→A constant is a type of variable whose value cannot be changed.

→in python, constants are usually declared and assigned in a module .

→the module is a new file containing variables,functions etc……

→inside the module, constants are written in all capital letters and underscores separating the words.

Example:

Constants.py:

pi=3.14

gravity=9.8

main.py:

```
import constant

print(constant.pi)

print(constant.gravity)
```

Taking input from keyboard

1.input():

This function automatically identifies whether user entered a string or list. If the input provided is not correct then either syntax error or exception is raised by python.

What ever you entered as input, input function convert it into string. If you enter an integer value still input() function convert it into integer in your code by using 'type casting'

Type casting:

Converting one data type into another datatype is called type casting.

Example:

```
>>> name=input("enter name:")

enter name:sushma

>>> print(name)

sushma

>>> print(type(name))
```

```
<class 'str'>

>>> id=10

>>> id=input("enter a number:")

enter a number:34

>>> print(type(id))

<class 'str'>

>>> sal=45000.2345

>>> sal=input("enter a salary:")

enter a salary:67000.2345

>>> print(type(sal))

<class 'str'>
```

1.typecasting the input to integer:

Example:

```
>>> id=int(input("enter is:"))

enter is:78

>>> print(type(id))

<class 'int'>
```

2.typecasting the input to float:

Example:

```
>>> sal=67000.234

>>> sal=float(input("enter salary:"))
```

```
enter salary:78000.234

>>> print(type(sal))

<class 'float'>

>>> sal=float(input("enter salary"))

enter salary78000

>>> print(type(sal))

<class 'float'>
```

3.typecasting the input to string:

Example:

```
>>> name=str(input())

sushma

>>> print(type(name))

<class 'str'>

>>>
```