>>> 2 + 3

5

>>> 9 – 8

1

>>> 4 \* 6

24

>>> 8 / 4

2.0

>>> 5/2

2.5

>>> 5 // 2

2

>>> 8 +9- 10

7

>>> 8 +9 -

SyntaxError: invalid syntax

>>> 8+2\*3

14

>>> (8 + 2) \* 3

30

>>> 2 \* 2 \* 2

8

>>> 2 \*\* 3

8

>>> 10 // 3

3

>>> 10% 3

1

>>> 'navin'

‘navin’

>>> print ('navin')

Navin

>>> print ('navin's laptop’)

SyntaxError: invalid syntax

>>> print ("navin's laptop"

navin!s laptop

>>> print ('navin "laptop"')

navin "laptop"

>>> print ('navin’s "laptop"')

SyntaxError: invalid syntax

>>> print('navin\'s "laptop"')

navin's "laptop"

>>> 'navin' + 'navin'

'navinnavin'

>>> 10 \* 'navin'

‘navinnavinnavinnavinnavinnavinnavinnavinnavinnavin’

>>> print ('c:\docs\navin')

c:\docs

avin

>>> print (r'c:\docs\navin')

c:\docs\navin

>>>

>>> 2 + 3

5

>>> x = 2

>>> x + 3

5

>>> y = 3

>>> x + y

5

>>> x = 9

>>> x + y

12

>>> x

9

>>> abc

Traceback (most recent call last):

File "<pyshell#8>", line 1, in <module>

abc

NameError: name ‘abc’ is not defined

>>> x + 10

19

>>> \_ + y

22

>>> name = 'youtube'

>>> name

‘youtube'

>>> name + ' rocks'

‘youtube rocks'

>>>

>>> name ' rocks’

SyntaxError: invalid syntax

>>> name [0]

‘y’

-7 -6 -5 -4 -3 -2 -1

Y O U T U B E

0 1 2 3 4 5 6

>>> name [6]

‘e’

>>> name [8] I

Traceback (most recent call last):

File "<pyshell#18>", line 1, in <module>

name [8]

IndexError: string index out of range

>>> name[-1]

‘e’

>>> name[-2]

‘b’

>>> name [-7]

‘y’

>>> name[0:2]

‘yo’

>>> name[1:4]

‘out’

>>> name[1:]

‘outube'

>>> name[:4]

‘yout'

>>> name [3:10]

‘tube'

>>> name [3:10]

‘tube’

>>> name[0:3] = ‘my’

Traceback (most recent call last):

File "<pyshell#27>", line 1, in <module>

name[(0:3] = 'my'

TypeError: 'str' object does not support item assignment

>>> name[0] = 'R'

Traceback (most recent call last):

File "<pyshell#28>", line 1, in <module>

name[0] = 'R'

TypeError: 'str' object does not support item assignment

>>> 'my ' + name[3:]

‘my tube'

>>> myname = 'Navin Reddy'

>>> len (myname)

11

>>> nums = [25,12,36,95,14]

>>> nums

[25, 12, 36, 95, 14]

>>> nums [0]

25

>>> nums[54\]

SyntaxError: unexpected character after line continuation character

>>> nums [54]

Traceback (most recent call last):

File "<pyshell#4>", line 1, in <module>

nums [54]

IndexError: list index out of range

>>> nums [4]

14

>>> nums[2:]

(36, 95, 14)

>>> nums[-1]

14

>>> nums[-5]

25

**Lists**

List is a type of array. Lists are mutable (Where values are added and removed any time). We can perform various operatuons like append multiple list remove values pop values etc

>>> names = ['navin',’kiran’,’john’]

>>> names

['navin', 'kiran', 'john']

>>> mil = [nums, names]

>>> mil

[[25, 12, 36, 95, 14], ['navin', 'kiran', 'john']]

>>> nums.append (45)

>>> nums

(25, 12, 36, 96, 14, 45)

>>> nums. insert (2,77)

>>> nums

[25, 12, 77, 36, 95, 14, 45]

>>> nums. remove (14)

>>> nums

(25, 12, 77, 36, 95, 45)

>>> nums.pop (1)

12

>>> nums

(25, 77, 36, 95, 45]

>>> nums.pop()

45

>>> del nums[2:]

>>> nums

[25, 77]

>>> nums.extend (29, 12,14, 36)

Traceback (most recent call last):

File "<pyshell#26>", line 1, in <module>

nums.extend (29, 12,14, 36)

TypeError: extend() takes exactly one argument (4 given)

>>> nums.extend ([29,12,14,36])

>>> nums

[25, 77, 29, 12, 14, 36]

>>> min (nums)

12

>>> max (nums)

77

>>> sum (nums)

193

>>> nums.sort ()

>>> nums

(12, 14, 25, 29, 36, 77]

**Tuple [Immutable {We cannot change the Values}]**

Lists are added with square brackets where as Tuple is using round brackets ()

Tuple are faster than List.

>>> tup = (21,36,14,25)

>>> tup

(21, 36, 14, 25)

>>> tup[1)

36

>>> tup[1] = 33

Traceback (most recent call last):

File "<pyshell#3>", line 1, in <module>

tup[1] = 33

TypeError: 'tuple' object does not supbort item assignment

Set

Set is also one type of set . we define set using flower brackets. The output of set is not in sequence. This set is used to improve the speed.

Indexing is not supported in set. Since the sequence is not in a proper format and it will be changes randomly.

>>> s = {22,25,14,21,5)}

>>> s

{5,14,21,22,25}

>>> s = {25,14,98,63,75,98}

>>> s

(98, 75, 14, 25, 63)

>>> s[2]

Traceback (most recent call last):

File "<pyshell#8>", line 1, in <module>

s(2]

TypeError: ‘set’ object does not support indexing

**HELP DOCS IN PYTHON**

help() // To see docs

help> topics //To get all topics etc etc

help> LISTS // To see docs about LISTS

if you are done use exit or just press enter.

>>> help('LISTS') //Without going to help menu and checking if you know you can use this syntax

>>> num = 5

>>> id (num)

1864841536

>>> name = 'navin'

>>> id (name)

90293952

>>> a = 10

>>> b=a

>>> a

10

>>> b

10

>>> id(a)

1864841616

>>> id(b)

1864841616

>>> id(10)

1864841616

>>> k = 10

>>> id (k)

1864841616

>>> a =9

>>> id(a)

18648416007

>>> id (b)

1864841616

>>> k=a

>>> id (k)

1864841600

>>> b=8

>>> PI = 3.14

>>> PI

3.14

>>> Type(PI)

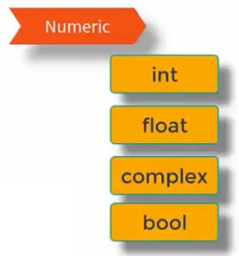
<class ‘float’>

**Data Types**



Dictionary is also called as Map (Some fellows also call it as hash/hastable/hashtab)

**None : Any variable which is not assigned and left free is considered as a None data type**



>>> num = 2.5

>>> type (num)

<class 'float'>

>>> num = 5

>>> type (num)

<class 'int'>

>>> num = 6+9j

>>> type (num)

<class 'complex'>

>>> a = 5.6

>>> b = int (a)

>>> type (b)

<class 'int'>

>>> b

5

>>> k

float (b)

>>> k

5.0

>> k= 6

>>> c = complex(b,k)

>>> c

(5+6j)

>>> b<k

True

>>> bool = b<k

>>> bool

True

>>> type (bool)

<class 'bool'>

>>> b > k

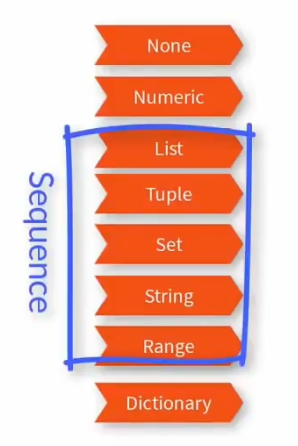
False

>>> int(True)

1

>>> int(False)

0



>>> 1st = [25,36,45,12]

>>> type (1st)

<class 'list'>

>>> s = (25,36, 45,15, 12,25}

>>> s

{36, 12, 45, 15, 25}

>>> type(s)

<class 'set'>

>>> t = (25,36, 4,57,12)

>>> type (t)

<class 'tuple'>

>>> str = "navin"

>>> st = ‘a’

>>> type (st)

<class 'str'>

>>> range (10)

range(0, 10)

>>> List (range (10))

[0, 1, 2, 3, 4, 5, 6, 7, 8 9]

>>> list (range (2,10,2))

(2, 4, 6, 8)

>>> type (range (10))

<class 'range'>

**Dictionary(Map)**

>>> d = {'navin':'samsung', 'rahul':'Iphone', 'kiran':'Oneplus'}

>>> d

{'navin':'samsung', 'rahul':'Iphone', 'kiran':'Oneplus'}

>>> d.keys ()

dict\_keys(['navin', 'rahul', 'kiran'])

>>> d.values()

dict\_values(['samsung', 'Iphone', 'Oneplus'])

>>> d['rahul']

'Iphone'

>>> d.get ("kiran")

'Oneplus'

Operators

* **Arithmetic Operators ( + - \* / // )**
* **Assignment Operator ( = )**
* **Relational Operators**
* **Logical Operators**
* **Unary Operators**

**Arithmetic Operators ( + - \* / // )**

>>> x = 2

>>> x = 3

>>> x + y

5

>>> x – y

-1

>>>x\*y

6

>>>x/y

0.6666666666666666

>>> x = x+2

>>> x

4

>>>x += 2

>>>x

6

>>> x \*=3

>>> x

18

>>> a,b = 5,6

>>> a

5

>>> b

6

Unary Operators

>>> n = 7

>>> n

7

>>> -n

-7

>>> n

7

>>> n = -n

>>> n

-7

Relational Operators

>>> a < b

True

>>> a > b

False

>>> a == b

False

>>> a = 6

>>> a == b

True

>>> a <= b

True

>>> a >= b

True

>>> a != b

False

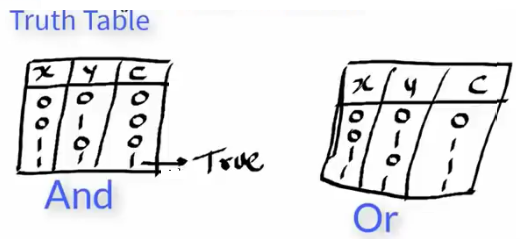
>>> b = 7

>>> a != b

True

Logical Operators

And or not



>>>a=5

>>>b=4

>>>a<8 and b<5

True

>>>a<8 and b<2

False

>>>a<8 or b<2

True

>>> x= True

>>> x

True

>>> not x

False

>>>x=not x

>>>x

False