In [2]:
```
# 2.)float
x=1.20
y=1.0
z=-34.58
print(type(x))
print(type(y))
print(type(z))
a=36e3
b=11E4
print(type(a))
print(type(b))
```

```
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
```

Sequence type

1.)List

- to store multiple items in single variable
- ordered,changeable & allow duplicate values
- indexed
- **square** brackets

In [3]:
```
l=["apple","banana","cherry"]
print(l)
print(type(l))
l1=["abc",35,"xyz",True,False]
print(l1)
l2=[3,3,4,4,5,6,7,1,2]
print(l2)
```

```
['apple', 'banana', 'cherry']
<class 'list'>
['abc', 35, 'xyz', True, False]
[3, 3, 4, 4, 5, 6, 7, 1, 2]
```

2.)Tuple

- to store multiple items in a single variable
- ordered,unchangeable,allow duplicate values
- indexed
- **round** brackets

In [10]:
```
t=("apple","banana","cherry")
print(t)
print(type(t))
t1=("abc",35,"xyz",True,False)
print(t1)
t2=(12,2,3,3,4,5)
print(t2)
```

```
('apple', 'banana', 'cherry')
<class 'tuple'>
('abc', 35, 'xyz', True, False)
(12, 2, 3, 3, 4, 5)
```

3.)Range(x)

->it will give numbers from 0 to x-1

In [11]:
```
x=range(3)
print(x)
print(type(x))
```

```
range(0, 3)
<class 'range'>
```

In [12]:
```
for i in range(3):
    print(i)
```

```
0
1
2
```

range(start,end,step);where step is the number of digits it will skip

In [23]:
```
for i in range(0,5):
    print(i)
```

```
0
1
2
3
4
```

In [21]:
```
for i in range(1,10,2):
    print(i)
```

```
1
3
5
7
9
```

In [28]:
```
for i in range(2,10,-1):
    print(i)
```

In [27]:
```
for i in range(10,1,-2):
    print(i)
```

```
10
8
6
4
2
```

In [29]:
```
for i in range(-10,-1,-2):
    print(i)
```

In [30]:
```
for i in range(-10,-1,2):
    print(i)
```

```
-10
-8
-6
-4
-2
```

In [31]:
```
for i in range(-10,-1):
    print(i)
```

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

Mapping type

1.)dict

- ordered
- changeable
- does not allow duplicates
- key-value pairs
- **curly** brackets with key-value pairs

In [38]:
```
d={10:[1,2,3,4,5],20:"Arman",30:(2,3,3,2),20:"Aryan"}
print(d)
print(d[10])
print(type(d))
print(type(d[10]))
print(type(d[30]))
```

```
{10: [1, 2, 3, 4, 5], 20: 'Aryan', 30: (2, 3, 3, 2)}
[1, 2, 3, 4, 5]
<class 'dict'>
<class 'list'>
<class 'tuple'>
```

Set type

1.set

- unordered,unindexed
- does not allow duplicates
- **curly** brackets

In [41]:
```
s={"apple","banana","cherry","apple"}
print(s)
print(type(s))
```

```
{'banana', 'cherry', 'apple'}
<class 'set'>
```

2.frozenset()

- it is immutable version of set

Boolean type:

bool-> **True or False**

In [49]:
```
print("20>8:",20>8)
print("20==8:",20==8)
print("20<8:",20<8)
print("""bool("abc"):""",bool("abc"))
print("bool(""):",bool(""))
print("bool(123):",bool(123))
print("""bool(["abc","xyz"]):""",bool(["abc","xyz"]))
print("bool(0):",bool(0))
print("bool(0.0):",bool(0.0))
print("bool(1):",bool(1))
print("bool(" "):",bool(" "))
```

```
20>8: True
20==8: False
20<8: False
bool("abc"): True
bool(): False
bool(123): True
bool(["abc","xyz"]): True
bool(0): False
bool(0.0): False
bool(1): True
bool(): True
```

In [51]:
```
x=1,2,3  #by default it will be considered tuple
print(type(x))
```

```
<class 'tuple'>
```

Global variable and Local variable

In [58]:
```
a="python" #->Global
def test():
    a="java" #->Local
    print(a)
test()
print(a)
```

```
java
python
```

In [61]:
```
a="python" #->Global
def test():
    global a
    a="java" #->Local
    print(a)
test()
print(a)
```

```
java
java
```

In [62]:
```
a="python" #->Global
def test():
    global a
    a="java" #->Local
    print(a)
print(a)
test()
```

```
python
java
```

Comments:

In [65]:
```
# This is comment
a=10
b=20
c=a+b   #addition
print(c) # print output

"""multiline
 comment"""
print("hello")
```

```
30
hello
```

Reading input from user

In [76]:
```
user=input("Enter username:")
print("The username is:",user)
```

```
Enter username:1ad
The username is: 1ad
```

In [73]:
```
a=input("Enter num1:")
b=input("Enter num2:")
```

```
c=a+b
print(c)
```

```
Enter num1:10
Enter num2:20
1020
```

In [74]:
```
a=int(input("Enter num1:"))
b=int(input("Enter num2:"))
c=a+b
print(c)
```

```
Enter num1:10
Enter num2:20
30
```

Type casting

1.)int

In [81]:
```
print(int(123.987))
print(int(True))
print(int(False))
print(int("10"))
#print(int("10.5"))
#print(int("ten"))
#print(int("0B1111"))
print(int(0B1111))
```

```
123
1
0
10
15
```

2.)float

In [84]:
```
print(float(123.987))
print(float(True))
print(float(False))
print(float("10"))
print(float("10.5"))
# print(float("ten"))
# print(float("0B1111"))
print(float(0B1111))
```

```
123.987
1.0
0.0
10.0
10.5
15.0
```

3.)bool

In [87]:
```
print("1.",bool(0))
print("2.",bool(1))
print("3.",bool(10))
print("4.",bool(10.5))
print("5.",bool(0.178))
print("6.",bool(0.0))
print("7.",bool(True))
print("8.",bool(False))
print("9.",bool("True"))
print("10.",bool("False"))
```

```
1 . False
2 . True
3 . True
4 . True
5 . True
6 . False
7 . True
8 . False
9 . True
10. True
```

4.)str

In [91]:
```
print(str(10))
print(str(10.5))
print(str(True))
print(str(False))
str(10)
```

```
10
10.5
True
False
```

Out[91]:
```
'10'
```

Python operators

1.)Arithmetic operators

- Addition(+) [strt->string concatenation]
- Subtraction(-)
- Multiplication(*) [str->string multiplication]
- Division(/)
- Modulus(%)
- Floor division(//)
- Exponent or power(**)

In [101…
```
a=int(input("Enter num1:"))
b=int(input("Enter num2:"))
print("Addition:",a+b)
print("Subtraction:",a-b)
print("Multiplication:",a*b)
print("Division:",a/b)
print("Modulus:",a%b)
print("Floor Division:",a//b)
print("Exponent or power:",a**b)
```

```
Enter num1:50
Enter num2:5
Addition: 55
Subtraction: 45
Multiplication: 250
Division: 10.0
Modulus: 0
Floor Division: 10
Exponent or power: 312500000
```

In [107…
```
print("5"*5)
```

```
55555
```

In [114…
```
print(12/5.0)
print(12//5)
print(12//5.0)
print(12.0//5)
print(12.0//5.0)
```

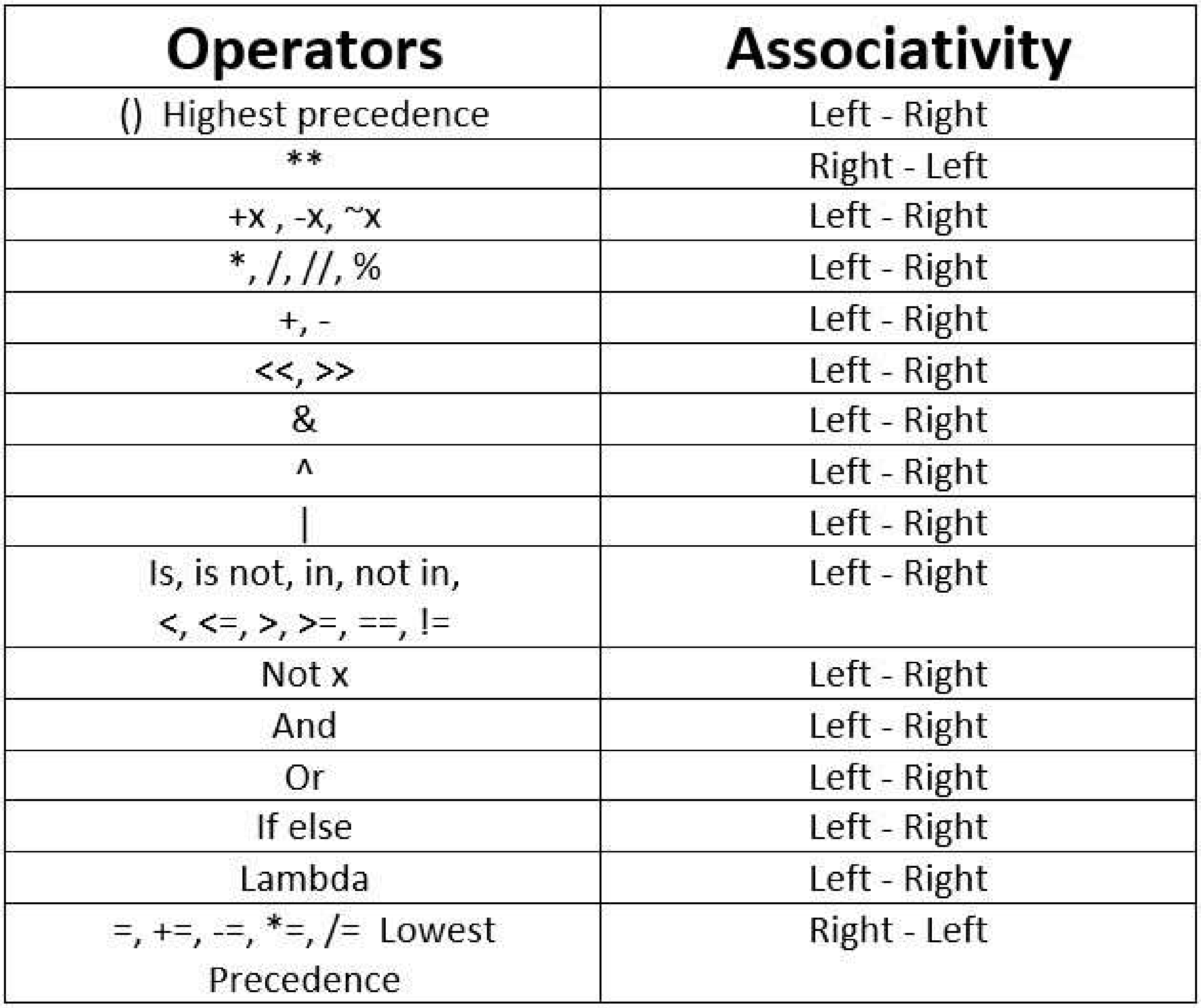
```
2.4
2
2.0
2.0
2.0
```

In [115…
```
(6+3)*(4+6)
```

Out[115…
```
90
```

In [116…
```
6+3*4+6
```

Out[116…
```
24
```



Operators	Associativity
() Highest precedence	Left - Right
**	Right - Left
+x , -x, ~x	Left - Right
*, /, //, %	Left - Right
+, -	Left - Right
<<, >>	Left - Right
&	Left - Right
^	Left - Right
|	Left - Right
Is, is not, in, not in, <, <=, >, >=, ==, !=	Left - Right
Not x	Left - Right
And	Left - Right
Or	Left - Right
If else	Left - Right
Lambda	Left - Right
=, +=, -=, *=, /= Lowest Precedence	Right - Left

In [117…
```
3**(2*2)**3
```
Out[117…
```
3433683820292512484657849089281
```
In [ ]:
```
```