2.Float

In [1]:
```
x=1.20
y=1.0
z=-34.58
print(type(x))
print(type(y))
print(type(z))
```

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

In [2]:
```
a=36e6  #36x10^6
b=11E4
print(a,b)
print(type(a))
print(type(b))
```

```
36000000.0 110000.0
<class 'float'>
<class 'float'>
```

Sequence type

List

```
    -> to store items in sigle veriable
    -> ordered, changeable & allow duplicate Value
    -> indexed
    -> square brackets
```

In [5]:
```
l=["apple","banana","cherry"]
print(l)
print(type(l))
l1=["abc",35,"xyz",True,False]
print(l1)
l2=[3,4,5,4,43,3,2,2,]
print(l2)
```

```
['apple', 'banana', 'cherry']
<class 'list'>
['abc', 35, 'xyz', True, False]
[3, 4, 5, 4, 43, 3, 2, 2]
```

Tuple

```
   -> to store items in sigle veriable
   -> ordered,unchangeable
   -> indexed
   -> round brackets
```

In [8]:
```
t=("A","B","C",5,6.67)
print(t)
print(type(t))
```

```
('A', 'B', 'C', 5, 6.67)
<class 'tuple'>
```

range()

range(start,end,stap)

In [11]:
```
x=range(3)
print(x)
print(type(x))
```
```
range(0, 3)
<class 'range'>
```

In [12]:
```
# eg.
for i in range(3):
    print(i)
```
```
0
1
2
```

range(2,10,2) # 2,4,6,8,9

In [18]:
```
for i in range(10,1,-2):
    print(i)
```
```
10
8
6
4
2
```

In [19]:
```
for i in range(-10,-4):
    print(i)
```
```
-10
-9
-8
-7
-6
-5
```

In [20]:
```
for i in range(10,4):# no output
    print(i)
```

Mapping

dict

In [21]:
```
d={10:"Lucky",20:"Arman",30:"Dhairya"}
print(d)
print(d[10])
print(type(d))
print(type(d[10]))
```
```
{10: 'Lucky', 20: 'Arman', 30: 'Dhairya'}
Lucky
<class 'dict'>
<class 'str'>
```

```
    --> Orered
    --> Changeble
    --> does not allow duplicates(if use then it overide the value)
    --> Key-Velue pairs
    --> curly brackets with key value pair
```

In [22]:
```
d={10:"Lucky",20:"Arman",30:"Dhairya",10:"Aryan"}
print(d)
```

```
{10: 'Aryan', 20: 'Arman', 30: 'Dhairya'}
```

set

```
--> unchanged,unindexed
--> does nor allow duplicates
--> curly brackets
```

In [23]:
```
s={"Apple","Bananan","Cherry","Apple"}
print(s)
print(type(s))
```

```
{'Apple', 'Bananan', 'Cherry'}
<class 'set'>
```

Boolean Type

bool--> True or False

In [63]:
```
print(20>8)
print(20==9)
print(20<8)
print(bool('abc')) # non zero string so its true
print(bool(""))
print(bool(123))
print(bool(["abc","xyz"])) # true
print(bool(0)) # False
print(bool(0.0))
print(bool(1))
print(bool(" "))
print(bool([])) # its zero so its False
print(bool("False")) #true
```

```
True
False
False
True
False
True
True
False
False
True
True
False
True
```

In [32]:
```
#key point
x=1,2,3
type(x)
```

Out[32]:
```
tuple
```

Global Variable vs Local Variable

In [33]:
```
a="Python"
def test(): # creation of function
```

```
    a="java"
    print(a)
test() # calling of function
print(a)
```

```
java
Python
```

In [34]:
```
a="Python"
def test():
    a="java"
    print(a)
print(a)
test()
```

```
Python
java
```

In [38]:
```
a="Python"
def test():
    global a
    a="java"
    print(a)
test()
print(a)
```

```
java
java
```

Comments

In [43]:
```
#This is Comment
a=10
b=20
c=a+b
print(c)
# if we write multiline comment at last is show in output
""" MultiLine
    comment"""
```

```
30
```

Out[43]:
```
' MultiLine \n    comment'
```

In [40]:
```
#This is Comment
a=10
b=20
c=a+b
print(c)
""" MultiLine
    comment"""
print(c)
```

```
30
30
```

Reading input from user

In [48]:
```
a=input("Enter username:")
print("The username is:"+a)
print(type(a))
```

```
Enter username:user
The username is:user
```

```
<class 'str'>
```

In [49]:
```
a=input("Enter number 1:")
b=input("Enter number 2:")
print(a+b)
```

```
Enter number 1:10
Enter number 2:20
1020
```

In [50]:
```
a=int(input("Enter number 1:"))
b=int(input("Enter number 2:"))
print(a+b)
```

```
Enter number 1:10
Enter number 2:20
30
```

Type Casting

int

In [54]:
```
print(int(123.987))
print(int(True))
print(int(False))
print(int("10"))
print(int(0B1111))
```

```
123
1
0
10
15
```

In [55]:
```
print(int("10.5"))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-55-24a9cbdf48a8> in <module>
----> 1 print(int("10.5"))

ValueError: invalid literal for int() with base 10: '10.5'
```

In [58]:
```
print(int("ten"))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-58-fe86acc464d3> in <module>
----> 1 print(int("ten"))

ValueError: invalid literal for int() with base 10: 'ten'
```

In [57]:
```
print(int("0B1111"))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-57-d69b2a855e10> in <module>
----> 1 print(int("0B1111"))

ValueError: invalid literal for int() with base 10: '0B1111'
```

float

In [62]:
```
print(float(123.987))
print(float(True))
print(float(False))
print(float("10"))
print(float("10.5"))
print(float(0B1111))
```

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

In [61]:
```
print(float("0B1111"))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-61-703a094a2758> in <module>
----> 1 print(float("0B1111"))

ValueError: could not convert string to float: '0B1111'
```

str

In [64]:
```
print(str(10))
print(str(10.5))
print(str(True))
print(str(False))
```

```
10
10.5
True
False
```

Python Operators

Arithnetic operator

```
+ -->Addition
- -->substraction
* -->multiplication
/ --> division
% --> modulus
// --> Floor Division
** --> exponent or power
```

In [70]:
```
a=int(input("Enter num1:"))
b=int(input("Enter num2:"))
print("Addition: ",a+b)
print("substraction: ",a-b)
print("multiplication: ",a*b)
print("division: ",a/b)   # ********it gives float*******
print("modulus: ",a%b)
print("Floor Division: ", a//b)
print("exponent or power:",a**b)
```

```
Enter num1:5
Enter num2:4
Addition:  9
substraction:  1
multiplication:  20
division:  1.25
modulus:  1
```

```
Floor Division:  1
exponent or power: 625
```

In [86]:
```
a=5
b=5
print(a/b) # not 1 its 1.0
print(a//b) # gives 1
print(12//5.0)
print(12//5)
print(12/5)
print(12.0//5)
```

```
1.0
1
2.0
2
2.4
2.0
```

In [72]:
```
"abc"*"abc"
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-72-d995279f0820> in <module>
----> 1 "abc"*"abc"

TypeError: can't multiply sequence by non-int of type 'str'
```

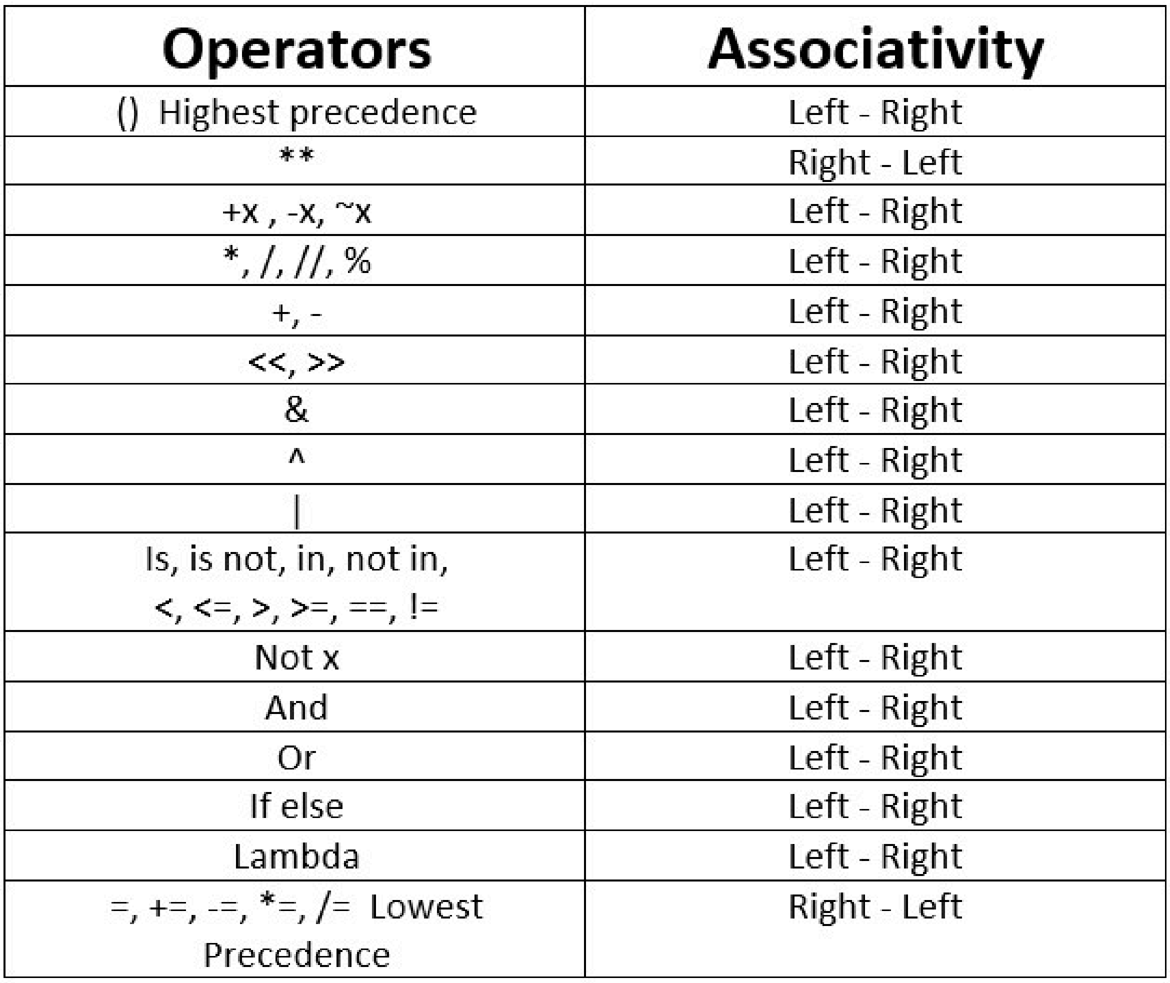
In [73]:
```
"abc"+10
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-73-5dbb00701316> in <module>
----> 1 "abc"+10

TypeError: can only concatenate str (not "int") to str
```

In [79]:
```
"abc_"*5
```

Out[79]:
```
'abc_abc_abc_abc_abc_'
```



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 [87]:
```
print(6+3*4+6)
```
```
24
```

In [88]:
```
print((6+3)*(4+6))
```
```
90
```

In [89]:
```
print(3**2*2**3)
```
```
72
```

In [90]:
```
print(3**(2*2)**3)
```
```
3433683820292512484657849089281
```