Q. Declare a float value and store it in a variable.

Check the type and print the id of the same.

```
In [1]: a=20.56
        print(a)
        print(type(a))
        print(id(a))

        20.56
        <class 'float'>
        2565312966544
```

Q. Arithmetic Operations on float

Take two different float values.

Store them in two different variables.

Do below operations on them:-

Find sum of both numbers

Find difference between them

Find the product of both numbers.

Find value after dividing first num with second number

Find the remainder after dividing first number with second number

Find the quotient after dividing first number with second number

Find the result of the first num to the power of the second number.

In [2]:
```python
a=20.56
b=30.44
print(a+b)
print(a-b)
print(a*b)
print(a/b)
print(a%b)
print(a//b)
print(a**b)
```

```
51.0
-9.880000000000003
625.8464
0.6754270696452036
20.56
0.0
9.298726587413567e+39
```

Q. Comparison Operators on float

Take two different float values.

Store them in two different variables.

Do below operations on them:-

Compare these two numbers with below operator:-

Greater than, '>'

Smaller than, '<'

Greater than or equal to, '>='

Less than or equal to, '<='

Observe their output(return type should be boolean)

In [3]:
```python
a=20.56
b=30.44
print(a > b)
print(a < b)
print(a >= b)
print(a <= b)
```

```
False
True
False
True
```

Q. Equality Operator

Take two different float values.

Store them in two different variables.

Equate them using equality operators (==, !=)

Observe the output(return type should be boolean)

In [5]:
```python
a=20.56
b=30.44
print(a==b)
print(a!=b)
```

```
False
True
```

Q. Logical operators

Observe the output of below code

Cross check the output manually

In [6]:
```python
print ( 10.20 and 20.30 ) #both are true and second value taken -> Output is 20.3
print ( 0.0 and 20.30 ) #First is false so first value taken -> Output is 0.0
print ( 20.30 and 0.0 ) #Goes to till second and second value is false so second
print ( 0.0 and 0.0 ) #First is false so first value is taken-> Output is 0.0
print ( 10.20 or 20.30 ) #First is True so first value is taken>Output is 10.2
print ( 0.0 or 20.30 ) #Goes to till second and second is true second value is to
print ( 20.30 or 0.0 ) #First is True so first value is taken->Output is 20.3
print ( 0.0 or 0.0 ) #Goes to till second and second is also false and second val
print ( not 10.20 ) #-Not of true is false->Output is False
print ( not 0.0 ) #Not of false is True>Output is True
```

```
20.3
0.0
0.0
0.0
10.2
20.3
20.3
0.0
False
True
```

Q. What is the output of expression inside print statement.

Cross check before running the program.

```
In [7]: a = 10.20
        b = 10.20
        print (a is b) #True or False? True 10.20<256
        print (a is not b) #True or False? False
        # Why the Id of float values are different when the same value is
        # assigned to two different variables
        # ex: a = 10.5 b=10.5. but id will be same if I assign the variable
        # having float i.e. a=c then both a and c's Id are same
```

```
False
True
```

Q. Bitwise operation is not applicable between instances of float.

Why the Id of float values are different when the same value is assigned to two different variables

ex: a = 10.5 b=10.5. but id will be same if I assign the variable having float i.e. a=c then both a and c's Id are same

Object Reusability concept is not applicable on float values.

```
In [8]: a = 10.5
        b= 10.5
        print(id(a))
        print(id(b))
        # Both have different Ids.
```

```
2565314826768
2565314826832
```

Q. Membership operation

in, not in are two membership operators and return boolean value

```python
In [10]: print ( '2.7' in 'Python2.7.8' ) #True
         print ( 10.20 in [ 10 , 10.20 , 10 + 20j , 'Python' ]) #True
         print ( 10.20 in ( 10 , 10.20 , 10 + 20j , 'Python' )) # True
         print ( 20.30 in { 1 , 20.30 , 30 + 40j }) # True
         print ( 2.3 in { 1 : 100 , 2.3 : 200 , 30 + 40j : 300 }) # True
         print ( 10 in range ( 20 )) # True
```

```
True
True
True
True
True
True
```