Objects

in python, we can treat any thing as a object

scenario -1

sequence/ordered objects
npn-sequence/unordered objects

scenario -2

iterable objects
non-iterable objects

scenario -3

immutable objects
mutable objects

sequence/ordered objects

any object which follow the insertion order is preserved concept (both input order and output order will be same), that type of objects are called sequence/ordered objects.

ex: str,list,tuple,dict(from python3.5+ versions onwards),
range,.......

ex:

```
>>> x="siva"
>>> x
'siva'
>>> type(x)
<class 'str'>

>>> y =[7,3,8,2]
>>> y
[7, 3, 8, 2]
>>> type(y)
<class 'list'>
```

non-sequence/unordered objects:

any object which dont follow the insertion order is preserved concept(both input order and output order will not be same),that type of objects are called non-sequence/unordered objects.

ex: set,frozenset,dict(2.x and 3.0 to 3.4.x)

```
ex:
---
>>> z={9,7,8,5}
>>> z
{8, 9, 5, 7}
>>> type(z)
<class 'set'>

iterable objects:
----------------
        we can use any objects as a repeated purpose that type of objects are
called iterable objects.

                        (or)

        any object which allows the iterations, that type of objects are called
iterable objects.

        ex: str,list,tuple,set,frozenset,dict,range,......

ex:
---
>>> x=[5,3,7,2]
>>> y="hai siva"
>>> for ele in x:
          print(y)

hai siva
hai siva
hai siva
hai siva

non- iterable objects:
---------------------
                we can't use any object as a repeated purpose that type of objects
are called non-iterable objects.

                        (or)

                any object which dont allows the iterations, that type of objects
are called non-iterable objects.

        ex: int,float,complex,bool

ex:
---
>>> x=4562
>>> y="hai siva"
>>> for ele in x:
```

```
          print (y)

Traceback (most recent call last):
   File "<stdin>", Line 1, in <module>
TypeError: 'int' object is not iterable
```

Immutable objects:
----
        any object which dont allows to modify(insert/delete/update) the data, that type of objects are called immutable objects.

        ex: int,float,complex,bool,str,tuple,frozenset,range,bytes,....

ex:
--
```
>>> x=(6,3,7,2)
>>> x
(6, 3, 7, 2)
>>> type(x)
<class 'tuple'>
>>> id(x)
1660397582064
>>> x[1]
3
>>> x[1]=30
TypeError: 'tuple' object does not support item assignment
```

mutable objects:
----
        any object which allows to modify(insert/delete/update) the data, that type of objects are called mutable objects.

        ex: list,set,dict,bytearray,...

ex:
--
```
>>> y=[5,3,7,2]
>>> y
[5, 3, 7, 2]
>>> type(y)
<class 'list'>
>>> id(y)
1660397749056
>>> y[1]
3
>>> y[1]=30
>>> y
[5, 30, 7, 2]
>>> id(y)
```

```
1660397749056

note:
----
different immutable objects having the same data but referencing to same object in
the memory location.

ex1:
---

        sample.py
        --------
x=10
y=10
z=10
print(id(x))
print(id(y))
print(id(z))

        output
        ------
C:\Users\DELL\Desktop>python sample.py
2276633870864
2276633870864
2276633870864

ex2:
---

        sample.py
        ---------
x="hai siva krishna"
y="hai siva krishna"
z="hai siva krishna"
print(id(x))
print(id(y))
print(id(z))

        output
        ------
C:\Users\DELL\Desktop>python sample.py
3027894658480
3027894658480
3027894658480

ex3:
---
        sample.py
        ---------
```

```
x=(1,2,3,4)
y=(1,2,3,4)
z=(1,2,3,4)
print(id(x))
print(id(y))
print(id(z))

        output
        ------
C:\Users\DELL\Desktop>python sample.py
1905125810928
1905125810928
1905125810928
```

different mutable objects having the same data but referencing to different objects in the memory location.

```
ex1:
---
        sample.py
        ---------
x=[1,2,3,4]
y=[1,2,3,4]
z=[1,2,3,4]
print(id(x))
print(id(y))
print(id(z))

        output
        ------
C:\Users\DELL\Desktop>python sample.py
2544447143424
2544447092864
2544447128512

ex2:
---
        sample.py
        ---------
x=[1,2,[3,4]]
y=[1,2,[3,4]]
z=[1,2,[3,4]]
print(id(x))
print(id(y))
print(id(z))
print('='*20)
print(id(x[0]))
print(id(y[0]))
print(id(z[0]))
print('='*20)
print(id(x[2]))
print(id(y[2]))
print(id(z[2]))
print('='*20)
print(id(x[2][0]))
print(id(y[2][0]))
print(id(z[2][0]))

        output
        ------
C:\Users\DELL\Desktop>python sample.py
2047431374976
2047431414976
2047431371392
==================
2047430033648
2047430033648
2047430033648
==================
2047431425536
2047431410624
2047431416640
==================
2047430033712
2047430033712
2047430033712
```