Advance Data Types

In [5]:

```
# list []

x = [1,3,6,["str",[10,52,True],"like",25.6,False],None]

print(x)
print(type(x))
print(x[1])
print(x[1])
print(x[3][1])
print(x[3][1][2])
```

```
[1, 3, 6, ['str', [10, 52, True], 'like', 25.6, False], None]
<class 'list'>
3
3
[10, 52, True]
True
```

In [27]:

```
k =[]  # empty list
print(k)
print(id(k))
print(type(k))
```

```
[]
2027985036672
<class 'list'>
```

In [28]:

```
k =[1,2],"string",56.3  # list
print(k)
print(id(k))
print(type(k))
```

```
([1, 2], 'string', 56.3)
2027984362368
<class 'tuple'>
```

In [11]:

```
x =[1,4.6,True,"hi"]
print(x)
x.append("good morning")  # add elements in the end
print(x)
x.append("good morning")
print(x)
```

```
[1, 4.6, True, 'hi']
[1, 4.6, True, 'hi', 'good morning']
[1, 4.6, True, 'hi', 'good morning', 'good morning']
```

In [12]:

```
# tupple ()
t = (1,2,3,4)
print(t)
print(id(t))
print(type(t))
```

```
(1, 2, 3, 4)
2027984268128
<class 'tuple'>
```

In [13]:

```
t = (1,2,"string",True)
print(t)
print(id(t))
print(type(t))
```

```
(1, 2, 'string', True)
2027984267728
<class 'tuple'>
```

In [15]:

```
print(t)
print(t.append(53))  # throws an error
```

```
(1, 2, 'string', True)

---------------------------------------------------------------------------
-
AttributeError                            Traceback (most recent call las
t)
Cell In[15], line 2
      1 print(t)
----> 2 print(t.append(53))  # throws an error

AttributeError: 'tuple' object has no attribute 'append'
```

Tupple mein modification nhi kar skte -> immutable List mein modification kar skte h -> mutable

In [18]:

```
# set {}
# set is un-ordered whereis -> List and Tupples are ordered

s = {1,2,3,4,5}
print(s)
print(id(s))
print(type(s))
```

```
{1, 2, 3, 4, 5}
2027978961888
<class 'set'>
```

In [20]:

```
p = {1,2,3,4,5}
print(p)
print(id(p))
print(type(p))
# access is not be able element wise
print(p[1])
```

```
{1, 2, 3, 4, 5}
2027978963456
<class 'set'>

---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
Cell In[20], line 6
      4 print(type(p))
      5 # access is not be able element wise
----> 6 print(p[1])

TypeError: 'set' object is not subscriptable
```

In [21]:

```
p = {1,2,3,5,4,8,9,3,3}
print(p)
print(id(p))
print(type(p))
```

```
{1, 2, 3, 4, 5, 8, 9}
2027984929280
<class 'set'>
```

In [23]:

```python
p = {1,2,3,4,5,"hello"}
print(p)
print(id(p))
print(type(p))
```

```
{1, 2, 3, 4, 5, 'hello'}
2027984931072
<class 'set'>
```

In [26]:

```python
# Set is mutable
p = {1,2,3,4,5,"hello"}
p.add("world")  # to add elements in set
print(p)
print(id(p))
print(type(p))
```

```
{1, 2, 3, 4, 5, 'world', 'hello'}
2027978963456
<class 'set'>
```

In [30]:

```python
# frozenset is immutable (freezed set)
s= {5,3,4,2,5,1}
print(s)
print(type(s))
f = frozenset(s)  # after declaring frozenset we cannot update anything in set
print(f)
print(type(f))
```

```
{1, 2, 3, 4, 5}
<class 'set'>
frozenset({1, 2, 3, 4, 5})
<class 'frozenset'>
```

In [31]:

```python
# Dictionary (key pair values)

d = {1:"vasima",2:"swati",3:"ravi"}
print(d)
print(id(d))
print(type(d))
```

```
{1: 'vasima', 2: 'swati', 3: 'ravi'}
2027984973696
<class 'dict'>
```

In [32]:

```
d = {"python":"vasima","physics":"swati","data":[1,4,6]}
print(d)
print(id(d))
print(type(d))
```

```
{'python': 'vasima', 'physics': 'swati', 'data': [1, 4, 6]}
2027984201792
<class 'dict'>
```

In [33]:

```
d["data"]
```

Out[33]:

```
[1, 4, 6]
```

In [35]:

```
d["data"]=[1,"hello"]
print(d)
```

```
{'python': 'vasima', 'physics': 'swati', 'data': [1, 'hello']}
```

only (list, set, dictionary) are mutable

whenever int,float,string,complex,boolian,tuple,frozenset are immutable

In [38]:

```
# immutability of integer
x=7
print(id(x))
y=7
print(id(y))
x=99
print(id(x))
```

```
140724185240552
140724185240552
140724185243496
```

In [39]:

```
# mutability of list
l=[7,5,5.6]
print(l)
print(id(l))
l.append(88)
print(l)
print(id(l))
```

```
[7, 5, 5.6]
2027985001024
[7, 5, 5.6, 88]
2027985001024
```

In [40]:

```
# None datatype
x = None
print(x)
print(id(x))
print(type(x))  # it works like a holder, later on any of datatype will be store in it
```

```
None
140724183853808
<class 'NoneType'>
```

In [43]:

```
# unicode
chr(65)
```

Out[43]:

```
'A'
```

In [45]:

```
ord("G")
```

Out[45]:

```
71
```

In [46]:

```
chr(0)
```

Out[46]:

```
'\x00'
```

In [44]:

```python
print("\N{grinning face}")
```

```
😀
```

In [47]:

```python
print("\N{slightly smiling face}")
```

```
🙂
```

In [49]:

```python
# range f() -> it is a f() (considered as data type)

for i in range(0,10):
    print(i,end=" ")
```

```
0 1 2 3 4 5 6 7 8 9
```

In [51]:

```python
for i in range(0,20,2): # 0 to (n-1) ,skiprange
    print(i,end=" ")
```

```
0 2 4 6 8 10 12 14 16 18
```

In [52]:

```python
# write a program to convert pi value as integer

import math
print(math.pi)
x = math.pi
print(x)
i = int(math.pi)
print(i)
print(type(i))
```

```
3.141592653589793
3.141592653589793
3
<class 'int'>
```

In [53]:

```python
# write a program to convert boolian value as integer

t = True
f = False
print("t =",t,"f =",f)
print(type(t))
```

```
t = True f = False
<class 'bool'>
```