There are two types of Data Structures

1. Mutable Mutable is when something is changeable or have the ability to change , in python , 'mutable' is the ability to change values . These are often objects that stores a collections of data . For example:

- list[]
- sets{}
- Dictionaries {key:values}

2. Imutable It is not possible to alter the values of an object in python , it is known as imutable object.Once imutable object is created it's value remains permanent and unchangeable.

- numerical(int,float)
- strings
- tuples

In [ ]:
```
#imutable --> cannot modify --> int,float,string,bool,tuples
# mutable--> modify --> list,sets,dictionaries
```

In [1]:
```
a=10 #int
# built in function --> id
print(a)
id(a)
```

```
10
```

Out[1]:
```
140736464205000
```

In [2]:
```
a='gaurav'
print(a)
id(a)
```

```
gaurav
```

Out[2]:
```
1739992098752
```

In [3]:
```
print(a.upper())
id(a.upper())
```

```
GAURAV
```

Out[3]:
```
1739970570272
```

In [6]:
```
# list

a1=['tiger','cat','lion','pikachu','panda']
print(a1)
id(a1)
```

```
['tiger', 'cat', 'lion', 'pikachu', 'panda']
```

Out[6]: 1739992413312

In [7]:
```
a1.append('dog')
print(a1)
id(a1)
```

```
['tiger', 'cat', 'lion', 'pikachu', 'panda', 'dog']
```

Out[7]: 1739992413312

In [10]:
```
a1.insert(0,'horse') # 0 means first position and 1 is second then else other
print(a1)
id(a1)
```

```
['horse', 'horse', 'tiger', 'cat', 'lion', 'pikachu', 'panda', 'dog']
```

Out[10]: 1739992413312

In [13]:
```
a1.insert(3,'monkey')
print(a1)
id(a1)
```

```
['horse', 'horse', 'tiger', 'monkey', 'monkey', 'cat', 'lion', 'pikachu', 'pand
a', 'dog', 'monkey']
```

Out[13]: 1739992413312

string

- entrapped 'sentance',"" or ''' ''' (paragraph)
- duplicacy is allowed
- ordered
    - immutable

In [14]:
```
# duplicacy allowed

a='aaaaa'
print(a)
```

```
aaaaa
```

In [15]:
```
a='vipul'
a
```

Out[15]: 'vipul'

Methods

- upper()

- lower()
- swapcase()

In [16]:
```
a='vipul'
a.upper()
```

Out[16]: 'VIPUL'

In [17]:
```
a=a.upper() # permanent
print(a)
a.lower()
```

```
VIPUL
```

Out[17]: 'vipul'

In [18]:
```
'Vipul'=='vipul'
```

Out[18]: False

In [21]:
```
# swap case

a = ' hi THIS SIDE, vIPUL . i AM A dATA SCIENTIST . i AM FROM dELHI . aNSHUM S
print(a)
print(a.swapcase()) # change capital to small to capital word
```

```
 hi THIS SIDE, vIPUL . i AM A dATA SCIENTIST . i AM FROM dELHI . aNSHUM SIR IS
THE BEST TEACHER AT sKILLCIRCLE.
 HI this side, Vipul . I am a Data scientist . I am from Delhi . Anshum sir is
the best teacher at Skillcircle.
```

In [22]:
```
# proprocessing --> nip

# strip
# lstrip
# rstrip
# split
# replace
```

In [23]:
```
name1 = '  Anshul bhargav '
name2 = 'Anshul bhargav'
```

In [24]:
```
name1==name2
```

Out[24]: False

In [25]:
```
# strip
# print(name1.strip())
print('|'+name1.strip()+'|')
print('|'+name1.lstrip()+'|')
print('|'+name1.rstrip()+'|')
```

```
|Anshul bhargav|
|Anshul bhargav |
|  Anshul bhargav|
```

In [26]:
```
str1='chicken_biryani momos lassi paratha chole_bhature egg pizza'
```

In [29]:
```
str1[0:15] #hetric
```

Out[29]:
```
'chicken_biryani'
```

In [30]:
```
a1=str1.split()
print(a1)
a1[3]
```

```
['chicken_biryani', 'momos', 'lassi', 'paratha', 'chole_bhature', 'egg', 'pizz
a']
```

Out[30]:
```
'paratha'
```

In [33]:
```
a1=str1.split()
```

In [31]:
```
str2= ' dhaniya;dal;pudina;fruits;beetroots;chawal;milk'
str2.split(';')
```

Out[31]:
```
[' dhaniya', 'dal', 'pudina', 'fruits', 'beetroots', 'chawal', 'milk']
```

In [34]:
```
# REPLACE

name1='  Vipul Pandey '
name2= 'Vipul Pandey'

name1.strip()
```

Out[34]:
```
'Vipul Pandey'
```

In [35]:
```
name2='!!vi!pul!!@'
name2.strip('!@')
```

Out[35]:
```
'vi!pul'
```

In [36]:
```
name2.replace('!','').replace('@','')
```

Out[36]:
```
'vipul'
```

In [38]:
```
sent1='Anshul is an introvert'
print(sent1)
```

```
Anshul is an introvert
```

In [39]:
```
sent1.replace('Anshul','Gaurav')
```

Out[39]:
```
'Gaurav is an introvert'
```

In [41]:
```
# is upper
# is lower
# is alpha
# is numeric
# is alnum
# is space
# is printable
# startwith
# endwith

name='GULSHAN'
name = 'GULshan'
name.isupper()
```

Out[41]: False

In [42]:
```
actor='akshay'
actor.lower()
```

Out[42]: 'akshay'

In [43]:
```
# isalpha --> alphabetical /a-z/A-Z
actor.isalpha()
actor2='tushar kapoor'
actor2.isalpha()
```

Out[43]: False

In [44]:
```
# isnumeric
a='10'
# int(a)
a.isnumeric()
```

Out[44]: True

In [45]:
```
#isspace
val1='  '
print(val1)
val1.isspace()
```

Out[45]: True

In [46]:
```
# isprintable
a = 'hello\ngaurav'
a.isprintable()
```

Out[46]: False

In [49]:
```
sent1= 'Gaurav is a good boy'
sent1.startswith('Gaurav')
sent1.endswith('boy')
```

Out[49]: True

In [3]:
```
# index
# count

sent1= 'anshum likes lassi.anshum can do anything.anshum keeps on asking xyz.'
print(sent1.count('anshum'))
print(sent1.count(' '))
print(sent1.count('.'))
```

```
3
9
3
```

In [5]:
```
name = 'Prabhakarna Sripalawardhana Attapattu Jayasuriya Laxmansriramkrishna S

print(name)
```

```
Prabhakarna Sripalawardhana Attapattu Jayasuriya Laxmansriramkrishna Shivavenka
ta Rajasekara Srinivasana Trichipalli Ekkaparampeer Perambadur Chinnaswami Muth
uswami Venugopal Iyer
```

In [11]:
```
# index
name.index('Rajasekara')
print(name[81:81+11])
print(name[81:92])
```

```
Rajasekara
Rajasekara
```

lists

- mutable
- multiple data types are allowed
- ordered
- duplicacy is allowed

In [2]:
```
l1=['Gaurav','anushu','varsha','Aditya']
l1
```

Out[2]: ['Gaurav', 'anushu', 'varsha', 'Aditya']

In [3]:
```
# typecasting
str1='abcd'
list(str1)
```

Out[3]: ['a', 'b', 'c', 'd']

In [4]:
```
# ordered
l1=['Gaurav','anshu','varsh','Aditya']
```

```
print(l1)

l1[0]
```

```
['Gaurav', 'anshu', 'varsh', 'Aditya']
```

Out[4]:
```
'Gaurav'
```

In [5]:
```
# multiple data types

l1=[1,2,3,4.3,'varsha',[1,23,34,2]]
l1
```

Out[5]:
```
[1, 2, 3, 4.3, 'varsha', [1, 23, 34, 2]]
```

In [6]:
```
# methods
#append
l1=['anshu','anshu','gaurav','aditya','anshul']
l1
```

Out[6]:
```
['anshu', 'anshu', 'gaurav', 'aditya', 'anshul']
```

In [7]:
```
l1.append('varsha')
l1
```

Out[7]:
```
['anshu', 'anshu', 'gaurav', 'aditya', 'anshul', 'varsha']
```

In [8]:
```
# insert
l1.insert(0,'vipul')
```

In [9]:
```
print(l1)
```

```
['vipul', 'anshu', 'anshu', 'gaurav', 'aditya', 'anshul', 'varsha']
```

In [10]:
```
#remove-->element name
#pop --> index

l1.remove('gaurav')
l1
```

Out[10]:
```
['vipul', 'anshu', 'anshu', 'aditya', 'anshul', 'varsha']
```

In [11]:
```
l1.pop(0)
```

Out[11]:
```
'vipul'
```

In [12]:
```
l1
```

Out[12]:
```
['anshu', 'anshu', 'aditya', 'anshul', 'varsha']
```

In [14]:
```
# extend
marks1=[99,98,92]
marks2=[90,91,93]
```

```
#append // wrong
marks1.extend(marks2)
print(marks1)
marks1[-1]
```

```
[99, 98, 92, 90, 91, 93]
```

Out[14]: 93

In [15]:
```
# clear()
marks=[98,91,92,94,91]
print(marks)
marks.clear()
print(marks)
```

```
[98, 91, 92, 94, 91]
[]
```

In [16]:
```
# sort()
# diffrence b/w sort and sorted?
l1=[2,3,11,234,24,22,4,3234,234,32,45,354]
# sorted --> built in function
print('sorted\t',sorted(l1),'temorary change') #temporary
print('real\t',(l1))
# sort --> list method
l1.sort()
print('sort\t',(l1),'for permanent change')
l1[::-1]
```

```
sorted          [2, 3, 4, 11, 22, 24, 32, 45, 234, 234, 354, 3234] temorary chan
ge
real         [2, 3, 11, 234, 24, 22, 4, 3234, 234, 32, 45, 354]
sort         [2, 3, 4, 11, 22, 24, 32, 45, 234, 234, 354, 3234] for permanent c
hange
```

Out[16]: [3234, 354, 234, 234, 45, 32, 24, 22, 11, 4, 3, 2]

In [17]:
```
# copy

# what is deep copy and what is shallow copy?
l1=['anshu','dog','cat','tiger']# original
l2=l1.copy()#shallow copy
l1.remove('anshu')
print('l1\t',l1)
print('l2\t',l2)
```

```
l1          ['dog', 'cat', 'tiger']
l2          ['anshu', 'dog', 'cat', 'tiger']
```

In [ ]:
```
#index
#count
```

In [18]:
```
l1.index('tiger')
l1.count('tiger')
```

Out[18]: 1

tuples

- immutable
- multiple data types are allowed
- ordered
- duplicacy is allowed
- but,entraped is()

In [19]:
```
t1=(1,2,3,4,5,5.56,'gaurav',4,4)
t1
```

Out[19]: (1, 2, 3, 4, 5, 5.56, 'gaurav', 4, 4)

methods

- index
- count

In [20]:
```
t1.index('gaurav')
t1.count(4)
```

Out[20]: 3

dictionaries

- key value pairs and are entrapped in {}
- multiple data types
- ordered
- duplicacy is not allowed

In [21]:
```
dict1={
    'happiness':'the state of being happy','extraordinary':'very unusual or re
}
dict1
```

Out[21]:
```
{'happiness': 'the state of being happy',
 'extraordinary': 'very unusual or remarkable.'}
```

In [22]:
```
dict1
```

Out[22]:
```
{'happiness': 'the state of being happy',
 'extraordinary': 'very unusual or remarkable.'}
```

In [23]:
```
dict2={
    'name':'Vipul',
    'course':'Data analytics','salary':100000,'age':20
}
dict2
```

Out[23]:
```
{'name': 'Vipul', 'course': 'Data analytics', 'salary': 100000, 'age': 20}
```

In [24]:
```
# methods
dict3={
    'teeth':30,
    'marks':90
}
```

In [25]:
```
print(dict3,dict2)
```

```
{'teeth': 30, 'marks': 90} {'name': 'Vipul', 'course': 'Data analytics', 'salar
y': 100000, 'age': 20}
```

In [26]:
```
# update
dict2.update(dict3)
```

In [27]:
```
dict2
```

Out[27]:
```
{'name': 'Vipul',
 'course': 'Data analytics',
 'salary': 100000,
 'age': 20,
 'teeth': 30,
 'marks': 90}
```

In [31]:
```
# value change
dict2['name']='Pandey'
dict2
```

Out[31]:
```
{'name': 'Pandey',
 'course': 'Data analytics',
 'salary': 100000,
 'age': 20,
 'teeth': 30,
 'marks': 90}
```

In [32]:
```
# clear()--> dictionary khali ho jati hai
dict1.clear()
dict1
```

Out[32]:
```
{}
```

In [33]:
```
# keys
# values
```

```
# items
print(dict2.keys())
print(dict2.values())
print(dict2.items())
```

```
dict_keys(['name', 'course', 'salary', 'age', 'teeth', 'marks'])
dict_values(['Pandey', 'Data analytics', 100000, 20, 30, 90])
dict_items([('name', 'Pandey'), ('course', 'Data analytics'), ('salary', 10000
0), ('age', 20), ('teeth', 30), ('marks', 90)])
```

In [34]:
```
# get
dict2.get('name')
dict2['name']
```

Out[34]:
```
'Pandey'
```

In [35]:
```
# dict2['beful']
dict2.get('feef')
```

In [36]:
```
# setdefault
dict2.setdefault('name')
```

Out[36]:
```
'Pandey'
```

In [39]:
```
# unknown
dict2.setdefault('wage')
```

In [40]:
```
dict2
```

Out[40]:
```
{'name': 'Pandey',
 'course': 'Data analytics',
 'salary': 100000,
 'age': 20,
 'teeth': 30,
 'marks': 90,
 'wage': None}
```

sets

- enterapped{}
- duplicacy is not allowed
- unordered
- mutable

In [1]:
```
# making
s1 = { 1,2,3,4,5,6}
s1
type(s1)
```

Out[1]:
```
set
```

In [2]:
```
# unordered
s2={'gaurav','varsha','roshan','aditya','khusbhoo'}
# s2[0]
s2
```

Out[2]:
```
{'aditya', 'gaurav', 'khusbhoo', 'roshan', 'varsha'}
```

In [3]:
```
# duplicacy is not allowed
s3={1,2,1,1,1,1,1,1}
s3
```

Out[3]:
```
{1, 2}
```

In [4]:
```
# methods
#union
#intersection
#difference
```

In [5]:
```
a={1,2,3,4,5}
b={4,5,6,7,8}
# common elements 4,5
# union
a.union(b)
```

Out[5]:
```
{1, 2, 3, 4, 5, 6, 7, 8}
```

In [6]:
```
a.intersection(b)
```

Out[6]:
```
{4, 5}
```

In [9]:
```
# difference
a.difference(b)
```

Out[9]:
```
{1, 2, 3}
```

In [10]:
```
b-a
```

Out[10]:
```
{6, 7, 8}
```

In [12]:
```
a-b #(difference)
```

Out[12]:
```
{1, 2, 3}
```

In [13]:
```
print('a',a)
print('b',b)
```

```
a {1, 2, 3, 4, 5}
b {4, 5, 6, 7, 8}
```

In [17]:
```
# update
# intersection_update
# difference_update
# union--> update
a={1,2,3,4,5}
b={4,5,6,7,8}
a.update(b)
a
```

Out[17]: {1, 2, 3, 4, 5, 6, 7, 8}

In [18]:
```
# intersection_update
a={1,2,3,4,5}
b={4,5,6,7,8}
a.intersection_update(b)
a
```

Out[18]: {4, 5}

In [21]:
```
# difference update
a={1,2,3,4,5}
b={4,5,6,7,8}
a.difference_update(b)
a
```

Out[21]: {1, 2, 3}

In [22]:
```
a={1,2,3,4,5}
b={4,5,6,7,8}
b.difference_update(a)
b
```

Out[22]: {6, 7, 8}

In [24]:
```
# copy
# list--> copy
# deep copy vs shallow copy?

s1={'yashika','prince','aqram','gaurav'} # original
# deep copy
s2=s1 # duplicate set
#copy method
s2=s1.copy()# shallow
s2.add('harsh')
print('s2',s2)
print('s1',s1)
```

```
s2 {'yashika', 'aqram', 'gaurav', 'harsh', 'prince'}
s1 {'yashika', 'aqram', 'prince', 'gaurav'}
```

In [25]:
```
# remove
# discard
s1.remove('gaurav')
```

```
s1
```

Out[25]: {'aqram', 'prince', 'yashika'}

In [27]:
```
s1.discard('aqram')
s1
```

Out[27]: {'prince', 'yashika'}

In [28]:
```
# s1.remove('sdfdgr') #error
s1.discard('fssfr')# error free
```

In [29]:
```
# add
s1.add('anshum')
s1
```

Out[29]: {'anshum', 'prince', 'yashika'}

In [31]:
```
# clear
s1.clear()
s1
```

Out[31]: set()

In [32]:
```
# empty set
s1={}
type(s1)
s2=set()
type(s2)
s2.add('yashika')
s2
```

Out[32]: {'yashika'}

In [ ]:
```

```