- intilaiztion
- type
- len
- min
- max
- sorted
- reveresd
- in
- for loop with in
- index
- for loop with index
- mutable immutable
- concatenation
- methods
- conditions

In [1]:
```
list1=[1,2,3,4,5]
list1
```

Out[1]:
```
[1, 2, 3, 4, 5]
```

In [2]:
```
type(list1)
```

Out[2]:
```
list
```

In [3]:
```
list2=['A','B','C','D']
list2
```

Out[3]:
```
['A', 'B', 'C', 'D']
```

In [4]:
```
list3=[1,2,3,'A','B','C']
list3
```

Out[4]:
```
[1, 2, 3, 'A', 'B', 'C']
```

In [5]:
```
list4=[1,2,'Apple','10.5',1.5,True,False,20+30j]
list4
```

Out[5]:
```
[1, 2, 'Apple', '10.5', 1.5, True, False, (20+30j)]
```

```
In [6]: list5=[10,10,10]
        list5
```

```
Out[6]: [10, 10, 10]
```

```
In [7]: list6=[1,2,3,['A','B','C']]
        list6
```

```
Out[7]: [1, 2, 3, ['A', 'B', 'C']]
```

```
In [10]: list7=[]
         list7
```

```
Out[10]: []
```

```
In [20]: list8=[_]
         list8
         # sepeate empty list form
         # Dataframe:[]
```

```
Out[20]: [[[[[[[[[[[]], [[]]]]]]]]]]]
```

```
In [21]: list1=[1,2,3,4,5]
         list2=['A','B','C','D']
         list3=[1,2,3,'A','B','C']
         list4=[1,2,'Apple','10.5',1.5,True,False,20+30j]
         list5=[10,10,10]
         list6=[1,2,3,['A','B','C']]
         list7=[]
         list8=[_]
```

- List represents with square brackets
- List can access the array of elements
- The values inside list we are calling as elements
- List can be have any data type
- List can have duplicates
- List in list possible , because list also a data type

```
In [24]: list1= [11,23,33,44,54]
         list2=['Apple','Banana','Cherry']
         list3=[10,20,30,'Apple','Banana']

         # len
         # max
         # min
         # revered
         # sorted
```

```
In [27]: len(list1),len(list2),len(list3)
```

```
Out[27]: (5, 3, 5)

In [28]: max(list1)

Out[28]: 54

In [29]: max(list2)  #

Out[29]: 'Cherry'

In [30]: max(list3)

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[30], line 1
----> 1 max(list3)

TypeError: '>' not supported between instances of 'str' and 'int'

In [31]: list3

Out[31]: [10, 20, 30, 'Apple', 'Banana']

In [32]: sorted(list1)

Out[32]: [11, 23, 33, 44, 54]

In [34]: sorted(list1,reverse=True)

Out[34]: [54, 44, 33, 23, 11]

In [35]: sorted(list3)

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[35], line 1
----> 1 sorted(list3)

TypeError: '<' not supported between instances of 'str' and 'int'

In [37]: reversed(list1)

Out[37]: <list_reverseiterator at 0x16dee601570>

In [38]: for i in reversed(list1):
             print(i)

54
44
33
23
11

In [40]: list(reversed(list1))

Out[40]: [54, 44, 33, 23, 11]

In [41]: list(reversed(list3)) # no need of compare
```

```
Out[41]: ['Banana', 'Apple', 30, 20, 10]
```

in

```
In [ ]: list object not callable
        restart and run
```

```
In [44]: list1=[1,2,3,4,'A','B','C']
         1 in list1
         2 in list1
         3 in list1

         #i in list1
```

```
Out[44]: True
```

```
In [45]: for i in list1:
             print(i,end=' ')
```

```
1 2 3 4 A B C
```

```
In [47]: for i in [1,2,3,4,'A','B','C']:
             print(i,end=' ')
```

```
1 2 3 4 A B C
```

```
In [49]: for i in '1234ABC':
             print(i,end=' ')
```

```
1 2 3 4 A B C
```

index

```
In [51]: list1=[11,22,33,44,'A','B','C']

         # -7  -6  -5  -4   -3  -2    -1
         # 11  22  33  44  'A'  'B'  'C'
         # 0   1   2   3    4    5     6
         list1[0]
         list1[1]
```

```
Out[51]: 22
```

slice

```
In [52]: l=[11,22,33,44,55,66,77,88,99,100,'A','B','C']
         l[::]
```

```
Out[52]: [11, 22, 33, 44, 55, 66, 77, 88, 99, 100, 'A', 'B', 'C']
```

```
In [53]: l[::-1]
```

```
Out[53]: ['C', 'B', 'A', 100, 99, 88, 77, 66, 55, 44, 33, 22, 11]
```

```
In [ ]: l=[11,22,33,44,55,66,77,88,99,100,'A','B','C']
        l[2:12:2]
        l[2:12:-2]
        l[2:-12:2]
```

```
l[-2:12:2]
l[2:-12:-2]
l[-2:12:-2]
l[-2:-12:-2]
l[12:2:2]
l[12:2:-2]
l[12:-2:2]
l[12:-2:-2]
l[-12:2:2]
```

In [54]:
```
l=[10,20]
l[0]
```

Out[54]:
```
10
```

In [58]:
```
l=[[10]]
# how can you get 10
# in a list 'l' how many elements are there: one element
# How can I access one element using index: 0
l[0] # the output also a list
l[0][0]
```

Out[58]:
```
10
```

In [60]:
```
l=[1,2,[10]]
l[2][0]
```

Out[60]:
```
10
```

In [68]:
```
l=[1,
   2,
   3,
   [5,6,['Apple']]
  ]
len(l)
len(l[3])
l[3][2][0]
```

Out[68]:
```
'Apple'
```

In [71]:
```
l=[1,2,[3,4,['A'],['B']]]
l
# How many elements
len(l)
```

Out[71]:
```
3
```

In [77]:
```
l[2][3][0]
```

Out[77]:
```
'B'
```

In [85]:
```
l=[1,[2,[3,[4,[5,['Banana']]]]]]
l[1][1][1][1][1][0]
```

Out[85]:
```
'Banana'
```

In [80]:
```
len(l[1])
```

```
Out[80]:  2
```

```
In [95]:  l=[[[[[[[[['onion']]]]]]]]]
          l[0][0][0][0][0][0][0][0][0]
```

```
Out[95]:  'onion'
```

```
In [98]:  ' hello'
```

```
Out[98]:  ' hello'
```

```
In [97]:  print('hai')
```

```
hai
```

```
In [ ]:  dec 25th   ==== jan1
```

Methods

```
In [1]:  dir([])
```

```
Out[1]: ['__add__',
 '__class__',
 '__class_getitem__',
 '__contains__',
 '__delattr__',
 '__delitem__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__getstate__',
 '__gt__',
 '__hash__',
 '__iadd__',
 '__imul__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__mul__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__reversed__',
 '__rmul__',
 '__setattr__',
 '__setitem__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'append',
 'clear',
 'copy',
 'count',
 'extend',
 'index',
 'insert',
 'pop',
 'remove',
 'reverse',
 'sort']
```

```
In [ ]: 'append',
'clear',
'copy',
'count',
'extend',
'index',
'insert',
'pop',
'remove',
'reverse',
'sort'
```

In [ ]:
```
- clear

- copy

- count

- reverse

- sort
```

Clear

In [2]:
```
l1=[1,2,3,'A','B','C']
l1.clear()
```

In [3]:
```
l1
```

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

Copy

In [4]:
```
l1=[1,2,3,'A','B','C']
l2=l1.copy()
l1.clear()
```

In [5]:
```
l1
```

Out[5]:
```
[]
```

In [6]:
```
l2
```

Out[6]:
```
[1, 2, 3, 'A', 'B', 'C']
```

In [11]:
```
l1=[1,2,3,3,3,3,'A','A','A','B','C']
l1.count(3)
```

Out[11]:
```
4
```

In [9]:
```
s1='aaaaaaaaabbbbbbbccc'
s1.count('a',3)
```

Out[9]:
```
6
```

In [12]:
```
l1=[1,2,3,3,3,3,'A','A','A','B','C']
l1.count(3,2)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[12], line 2
      1 l1=[1,2,3,3,3,3,'A','A','A','B','C']
----> 2 l1.count(3,2)

TypeError: list.count() takes exactly one argument (2 given)
```

reverse

```
In [13]: l1=[1,2,3,3,3,3,3,'A','A','A','B','C']
         l1.reverse()
```

```
In [14]: l1
```

```
Out[14]: ['C', 'B', 'A', 'A', 'A', 3, 3, 3, 3, 2, 1]
```

```
In [15]: l1=[1,2,3]
         l2=[4,5,6]
         l1+l2
```

```
Out[15]: [1, 2, 3, 4, 5, 6]
```

```
In [16]: l1
```

```
Out[16]: [1, 2, 3]
```

IN PLACE

- whenever you want to overwrite the operation output
- in same variable we will use **inplace**
- In place means overwriting the existing elements
- some methods have inplace argument
- inplace= True means overwrite
- inplace = False means do not overwrites

```
In [17]: lst=[10,20,20,30,'apple','apple','banana','axe','ape']
         lst.reverse()
```

```
In [18]: lst
```

```
Out[18]: ['ape', 'axe', 'banana', 'apple', 'apple', 30, 20, 20, 10]
```

Sort

```
In [19]: lst=[10,20,20,30,'apple','apple','banana','axe','ape']
         lst.sort()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[19], line 2
      1 lst=[10,20,20,30,'apple','apple','banana','axe','ape']
----> 2 lst.sort()

TypeError: '<' not supported between instances of 'str' and 'int'
```

- sorted

- reveresd
- sort
- reverse

```
In [20]: l1=[1,22,33,3]
         l1.sort()
```

```
In [21]: l1
```

```
Out[21]: [1, 3, 22, 33]
```

```
In [23]: l1=[1,22,33,3]
         sorted(l1)
```

```
Out[23]: [1, 3, 22, 33]
```

```
In [24]: l1
```

```
Out[24]: [1, 22, 33, 3]
```

```
In [ ]: len([1,2,3])
        len('123')
```

```
In [25]: dir('')
```

```
Out[25]: ['__add__',
          '__class__',
          '__contains__',
          '__delattr__',
          '__dir__',
          '__doc__',
          '__eq__',
          '__format__',
          '__ge__',
          '__getattribute__',
          '__getitem__',
          '__getnewargs__',
          '__getstate__',
          '__gt__',
          '__hash__',
          '__init__',
          '__init_subclass__',
          '__iter__',
          '__le__',
          '__len__',
          '__lt__',
          '__mod__',
          '__mul__',
          '__ne__',
          '__new__',
          '__reduce__',
          '__reduce_ex__',
          '__repr__',
          '__rmod__',
          '__rmul__',
          '__setattr__',
          '__sizeof__',
          '__str__',
          '__subclasshook__',
          'capitalize',
          'casefold',
          'center',
          'count',
          'encode',
          'endswith',
          'expandtabs',
          'find',
          'format',
          'format_map',
          'index',
          'isalnum',
          'isalpha',
          'isascii',
          'isdecimal',
          'isdigit',
          'isidentifier',
          'islower',
          'isnumeric',
          'isprintable',
          'isspace',
          'istitle',
          'isupper',
          'join',
          'ljust',
          'lower',
          'lstrip',
          'maketrans',
          'partition',
          'removeprefix',
          'removesuffix',
          'replace',
          'rfind',
          'rindex',
          'rjust',
          'rpartition',
          'rsplit',
          'rstrip',
          'split',
          'splitlines',
          'startswith',
          'strip',
          'swapcase',
          'title',
          'translate',
          'upper',
          'zfill']
```

- keywords or inbuilt function str , list etc
- methods is resepctvive data types only

append

- Till now we just printed the output
- If you want to save the output in a list
- then we can use append method
- append means adding an element in a list
- the element will add at last, and the output will be overwrite

```
In [27]: l=[1,2,3,4]
         l.append(40)
         l
```

```
Out[27]: [1, 2, 3, 4, 40]
```

```
In [28]: l=[1,2,3,4]
         l.append(20)
         l.append(40)
         l
```

```
Out[28]: [1, 2, 3, 4, 20, 40]
```

```
In [29]: l.append(20,30)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[29], line 1
----> 1 l.append(20,30)

TypeError: list.append() takes exactly one argument (2 given)
```

In [30]:
```
l=[1,2,3,4]
l.append(['Apple','Banana'])
l
```

Out[30]:
```
[1, 2, 3, 4, ['Apple', 'Banana']]
```

How to fill the values in a empty string

How to fill the values in a empty list

In [31]:
```
s=''
s=s+'apple'
s
```

Out[31]:
```
'apple'
```

In [33]:
```
l=[]
l=l+['apple']
l
```

Out[33]:
```
['apple']
```

In [34]:
```
l=[]
l.append('Apple')
l
```

Out[34]:
```
['Apple']
```

In [ ]:
```
#but if we need to add 10 element then we have to do 10 times append ?
l=[]
l.append(10)
l.append(10)
l.append(10)
l.append(10)
```

In [35]:
```
l=[]
for i in range(10):
    l.append(10)
l
```

Out[35]:
```
[10, 10, 10, 10, 10, 10, 10, 10, 10, 10]
```

In [36]:
```
l=[]
for i in range(1,11):
    l.append(i)
l
```

Out[36]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
In [39]: l=[]
         for i in range(1,6):
             l.append(i*i)
         l

Out[39]: [1, 4, 9, 16, 25]
```

Extend

```
In [ ]: # take 1 list = l1=[1,2,3,4]
        # take 2 list = l2=['A','B','C']
        # l1.append(l2)
        # l1+l2
        # l1.extend(l2)

        #
```

```
In [42]: l1=[1,2,3,4]
         l2=['A','B','C']
         l1.append(l2)
         print('l1:',l1)
         print('l2:',l2)

l1: [1, 2, 3, 4, ['A', 'B', 'C']]
l2: ['A', 'B', 'C']
```

```
In [43]: l1=[1,2,3,4]
         l2=['A','B','C']
         l1.extend(l2)
         print('l1:',l1)
         print('l2:',l2)

l1: [1, 2, 3, 4, 'A', 'B', 'C']
l2: ['A', 'B', 'C']
```

```
In [45]: l1=[1,2,3,4]
         l2=['A','B','C']
         print(l1+l2)
         print('l1:',l1)
         print('l2:',l2)

[1, 2, 3, 4, 'A', 'B', 'C']
l1: [1, 2, 3, 4]
l2: ['A', 'B', 'C']
```

```
In [3]: # WAP1) Ask the user enter 5 values get the evn numbers and odd numbers list
        # Idea : we alreday did this but we use print statement
        #       instead of print the values, append the values
        # take two list
        # even_list=[]
        # odd_list=[]
        # perform the operation
        even_list,odd_list=[],[]
        for i in range(5):
            num=eval(input('enter the number:'))
            if num%2==0:
                even_list.append(num)
            else:
                odd_list.append(num)
```

```
even_list,odd_list
```

Out[3]:
```
([6, 8], [3, 5, 7])
```

In [5]:
```
# Q2) WAP
# l=['hyd','mumbai','chennai']
# op=['HYD','MUMBAI','CHENNAI']

# Q3) WAP
# L=['hyd','mumbai','chennai']
# o=['Hyd','Mumbai','Chennai']

# Q4) Wap
# L=['hyd','mum#bai','chen#nai']
# o=[mum#bai','chen#nai']

# Q5) Wap
# L=['hyd','mum#bai','chen#nai']
# o=['hyd']

# Q6): string='hello hai how are you'
#   op= ['Hello','Hai','How','Are','You']

# Q7):
# input:
# string1='virat.kohli@rcb.com, Rohit.sharma@Mi.co,
#           Dhoni.Mahendra@csk.com'

# output
# fnames=['virat','Rohit','Dhoni']
# sname=['kohli,sharma,Mahendar']
# cname=['rcb','Mi','Csk']

# Q8) string1='can canner can you can not canner can be can'
# Get the count of each word in above string in a list
# o.p: ['can-5','canner-2','you-1','not-1','be-1']


# Q9)  get 5 randm numbers between 1 to 100
#      append in a list
#     Find the maximum and minimu value of a given list
#      with out using max and min functions


# Q10) list_qns=['Who is PM of india','Who is ICT ODI captain',
#                'How many states are there in India']
#      list_ans= ['Modi','Rohit',29]
# step-1: Iterate each qn
# step-2: ask the user enter the answer for corresponding qn
# step-3: match that answer with list_ans
# Note:  Make sure first qns ans match with first index of the ans
# step-4: if both are match give the one mark
# step-5: Finally print how many qns are correct
#         how many marks got

list_qns=['Who is PM of india','Who is ICT ODI captain',
               'How many states are there in India']
list_ans= ['Modi','Rohit',29]
for i in list_qns:
    ans=input(i)
    if ans im list_ans:
```

In [ ]:
```
# attend the both
# enjoyed the both
# dont skip   3 ==
# good end to end
```

In [ ]:
```
TextLoader
Textloader
# starting === fun
# so much ==== fun

#
# Naresh it
# Sep 15th ====

# === workshops every weekend
# other trainer
# Langchain+RAG+FASTAPI ===
# Python part2
# statit

# workshop 750 +550
# 350
# rag : 68

# workshops
# youtube === pratcice
# like -share -comment
# ====
```

In [ ]:
```
# Q2) WAP
# l=['hyd','mumbai','chennai']
# op=['HYD','MUMBAI','CHENNAI']

# Q3) WAP
# L=['hyd','mumbai','chennai']
# o=['Hyd','Mumbai','Chennai']

# Q4) Wap
# L=['hyd','mum#bai','chen#nai']
# o=[mum#bai','chen#nai']

# Q5) Wap
# L=['hyd','mum#bai','chen#nai']
# o=['hyd']



# Q7):
# input:
# string1='virat.kohli@rcb.com, Rohit.sharma@Mi.co,
#           Dhoni.Mahendra@csk.com'

# output
# fnames=['virat','Rohit','Dhoni']
# sname=['kohli,sharma,Mahendar']
# cname=['rcb','Mi','Csk']
# Q8) string1='can canner can you can not canner can be can'
# Get the count of each word in above string in a list
# o.p: ['can-5','canner-2','you-1','not-1','be-1']


# Q9)  get 5 randm numbers between 1 to 100
#      append in a list
#    Find the maximum and minimu value of a given list
#     with out using max and min functions


# Q10) list_qns=['Who is PM of india','Who is ICT ODI captain',
#                'How many states are there in India']
#      list_ans= ['Modi','Rohit',29]
# step-1: Iterate each qn
# step-2: ask the user enter the answer for corresponding qn
# step-3: match that answer with list_ans
# Note:  Make sure first qns ans match with first index of the ans
# step-4: if both are match give the one mark
# step-5: Finally print how many qns are correct
#         how many marks got

# list_qns=['Who is PM of india','Who is ICT ODI captain',
#                 'How many states are there in India']
# list_ans= ['Modi','Rohit',29]
# for i in list_qns:
#     ans=input(i)
#     if ans im list_ans:
```

In [3]:
```
# Q4) Wap
# L=['hyd','mum#bai','chen#nai']
# o=[mum#bai','chen#nai']

L=['hyd','mum#bai','chen#nai']
o=[]
for i in L:
    if '#' in i:
        o.append(i)


o
```

Out[3]:
```
['mum#bai', 'chen#nai']
```

In [4]:
```
L=['hyd','mum#bai','chen#nai']
o=[]
for i in L:
    if '#' not in i:
        o.append(i)


o
```

Out[4]:
```
['hyd']
```

In [6]:
```
# Q6): string='hello hai how are you'
#  op= ['Hello','Hai','How','Are','You']

string='hello hai how are you'
lst=string.split()
```

```
o=[]
for i in lst:
    o.append(i.capitalize())
o
```

```
Out[6]: ['Hello', 'Hai', 'How', 'Are', 'You']
```

```
In [8]: string='hello hai how are you'
        string.title().split()
```

```
Out[8]: ['Hello', 'Hai', 'How', 'Are', 'You']
```

Join

```
In [10]: string='hello hai how are you'
         l=string.split()
         l
```

```
Out[10]: ['hello', 'hai', 'how', 'are', 'you']
```

```
In [12]: ' '.join(l)
```

```
Out[12]: 'hello hai how are you'
```

```
In [13]: '***'.join(l)
```

```
Out[13]: 'hello***hai***how***are***you'
```

- split and join together
- split works for strings
- join works for list

```
In [16]: # Q7):
         # input:
         # string1='virat.kohli@rcb.com, Rohit.sharma@Mi.co,
         #           Dhoni.Mahendra@csk.com'

         # output
         # fnames=['virat','Rohit','Dhoni']
         # sname=['kohli,sharma,Mahendar']
         # cname=['rcb','Mi','Csk']

         string1='virat.kohli@rcb.com, Rohit.sharma@Mi.co, Dhoni.Mahendra@csk.com'
         l=string1.split()
         l[0]
         first_dot_index=l[0].index('.')
         second_dot_index=l[0].index('.',1+first_dot_index)
         index_at_the_rat=l[0].index('@')
         first_name=l[0][:first_dot_index]
         sname=l[0][first_dot_index+1:index_at_the_rat]
         cname=l[0][index_at_the_rat+1:second_dot_index]
         first_name,sname,cname
```

```
Out[16]: ('virat', 'kohli', 'rcb')
```

```
In [18]: string1='virat.kohli@rcb.com, Rohit.sharma@Mi.co, Dhoni.Mahendra@csk.com'
         l=string1.split()

         fname=[]
         sname=[]
         cname=[]
         for i in range(len(l)):
             first_dot_index=l[i].index('.')
             second_dot_index=l[i].index('.',1+first_dot_index)
             index_at_the_rat=l[i].index('@')
             f1=l[i][:first_dot_index]
             s1=l[i][first_dot_index+1:index_at_the_rat]
             c1=l[i][index_at_the_rat+1:second_dot_index]
             fname.append(f1)
             sname.append(s1)
             cname.append(c1)
         fname,sname,cname
```

```
Out[18]: (['virat', 'Rohit', 'Dhoni'],
          ['kohli', 'sharma', 'Mahendra'],
          ['rcb', 'Mi', 'csk'])
```

```
In [39]: # Q8) string1='can canner can you can not canner can be can'
         # Get the count of each word in above string in a list
         # o.p: ['can-5','canner-2','you-1','not-1','be-1']
         string1='can canner can you can not canner can be can'
         l=string1.split() #[can,canner,can,you, can, not, canner, can, be, can]
         l1=[]
         o=[]
         for i in l:
             if i not in l1:
                 o.append(f"{i}-{l.count(i)}")
                 l1.append(i)


         # step-1: i='can'    'can' not in []    can=5    l1=['can']
         # step-1: i='canner'    'canner' not in ['can']    canner=2    l1=['can','canner']
         # step-1: i='can'    'can' not in ['can','canner']    False

         o
```

```
Out[39]: ['can-5', 'canner-2', 'you-1', 'not-1', 'be-1']
```

```
In [24]: string1='can canner can you can not canner can be can'
         string1.count('e')
```

```
Out[24]: 3
```

```
In [28]: l=['priyanka','priyanka']
         l.count('priyanka')
```

```
Out[28]: 2
```

```
In [29]: s='priyanka priyanka'
         s.count('priyanka')
```

```
Out[29]: 2
```

In [40]:
```
# Take some paragraph from wikipedia
# Print how many words count
# each word and its count

# Condition: aviod stop words
# ['a','an','the']
para="""Data science is "a concept to unify statistics, data analysis, informati
     "understand and analyze actual phenomena" with data.[5] It uses techniques
     statistics, computer science, information science, and domain knowledge.
     [6] However, data science is different from computer science and informatio
     Turing Award winner Jim Gray imagined data science as a "fourth paradigm" o
     (empirical, theoretical, computational, and now data-driven)
     and asserted that "everything about science is changing because of the impa
    """

# Idea: make a new paragraph with out punctuations with out numbers
# idea: make a another paragreaph with out  'a' ,'an','the'
# count words

# How many times Data repetaed
# like each and evry word
```

In [41]:
```
para.split()
```

```
Out[41]: ['Data',
           'science',
           'is',
           '"a',
           'concept',
           'to',
           'unify',
           'statistics,',
           'data',
           'analysis,',
           'informatics,',
           'and',
           'their',
           'related',
           'methods"',
           'to',
           '"understand',
           'and',
           'analyze',
           'actual',
           'phenomena"',
           'with',
           'data.[5]',
           'It',
           'uses',
           'techniques',
           'and',
           'theories',
           'drawn',
           'from',
           'many',
           'fields',
           'within',
           'the',
           'context',
           'of',
           'mathematics,',
           'statistics,',
           'computer',
           'science,',
           'information',
           'science,',
           'and',
           'domain',
           'knowledge.',
           '[6]',
           'However,',
           'data',
           'science',
           'is',
           'different',
           'from',
           'computer',
           'science',
           'and',
           'information',
           'science.',
           'Turing',
           'Award',
           'winner',
 'Jim',
 'Gray',
 'imagined',
 'data',
 'science',
 'as',
 'a',
 '"fourth',
 'paradigm"',
 'of',
 'science',
 '(empirical,',
 'theoretical,',
 'computational,',
 'and',
 'now',
 'data-driven)',
 'and',
 'asserted',
 'that',
 '"everything',
 'about',
 'science',
 'is',
 'changing',
 'because',
 'of',
 'the',
 'impact',
 'of',
 'information',
 'technology"',
 'and',
 'the',
 'data',
 'deluge.[7][8]']
```

In [42]:
```
import string
```

In [43]:
```
string.punctuation
```

Out[43]:
```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

In [44]:
```
string.digits
```

Out[44]:
```
'0123456789'
```

In [48]:
```
# Q10) list_qns=['Who is PM of india','Who is ICT ODI captain',
#                'How many states are there in India']
#      list_ans= ['Modi','Rohit',29]

list_qns=['Who is PM of india','Who is ICT ODI captain',
            'How many states are there in India']

list_ans= ['Modi','Rohit',29]
for qn in list_qns:
    ans=input(qn+':')
    if list_ans[<qn index>]
```

```
In [52]: list_qns=['Who is PM of india','Who is ICT ODI captain',
                     'How many states are there in India']
         list_ans[list_qns.index('Who is PM of india')]

Out[52]: 'Modi'

In [53]: list_ans[list_qns.index('Who is ICT ODI captain')]

Out[53]: 'Rohit'

In [54]: list_ans[list_qns.index('How many states are there in India')]

Out[54]: 29

In [ ]: # First get each qns
        # get the index of the each qns
        # pass that index to the list ans ==== > original answer
        # compare user answer with original answer

In [62]: list_qns=['Who is PM of india','Who is ICT ODI captain',
                     'How many states are there in India']

         list_ans= ['Modi','Rohit',29]
         for qn in list_qns:
             user_ans=input(qn+':')
             id=list_qns.index(qn)
             original_ans=list_ans[id]
             if original_ans.casefold()==user_ans.casefold():
                 print('correct')

         # Type check you finish

        correct
        correct
        ---------------------------------------------------------------------------
        AttributeError                            Traceback (most recent call last)
        Cell In[62], line 9
              7 id=list_qns.index(qn)
              8 original_ans=list_ans[id]
        ----> 9 if original_ans.casefold()==user_ans.casefold():
             10     print('correct')

        AttributeError: 'int' object has no attribute 'casefold'

In [ ]: saturday azure session 5.30pm
        sunday azure session 10am
        ================================
        naresh it
        this Ds course =====

        Azure + Langchain RAG+ FastAPI
        placement
        FLASK model deployment ====
        OOPS
        EDA 2
        Maths

In [1]: dir([])
```

```
Out[1]: ['__add__',
         '__class__',
         '__class_getitem__',
         '__contains__',
         '__delattr__',
         '__delitem__',
         '__dir__',
         '__doc__',
         '__eq__',
         '__format__',
         '__ge__',
         '__getattribute__',
         '__getitem__',
         '__getstate__',
         '__gt__',
         '__hash__',
         '__iadd__',
         '__imul__',
         '__init__',
         '__init_subclass__',
         '__iter__',
         '__le__',
         '__len__',
         '__lt__',
         '__mul__',
         '__ne__',
         '__new__',
         '__reduce__',
         '__reduce_ex__',
         '__repr__',
         '__reversed__',
         '__rmul__',
         '__setattr__',
         '__setitem__',
         '__sizeof__',
         '__str__',
         '__subclasshook__',
         'append',
         'clear',
         'copy',
         'count',
         'extend',
         'index',
         'insert',
         'pop',
         'remove',
         'reverse',
         'sort']
```

```
In [ ]: 'insert',
        'pop',
        'remove',
```

```
In [3]: l=['Apple','Ball','Cat']
        l.append('Dog')
        l
```

```
Out[3]: ['Apple', 'Ball', 'Cat', 'Dog']
```

```
In [4]: l=['Apple','Ball','Cat']
        #   0       1      2
        #   A       B       D     C
        #   0       1       2      3
        l.insert(2,'Dog')
        l

Out[4]: ['Apple', 'Ball', 'Dog', 'Cat']

In [ ]: # Remove
        # pop

In [6]: l=['Apple','Ball','Cat','Dog']
        l.remove('Cat')

In [7]: l

Out[7]: ['Apple', 'Ball', 'Dog']

In [8]: l=['Apple','Ball','Cat','Cat','Dog']
        l.remove('Cat')
        l

Out[8]: ['Apple', 'Ball', 'Cat', 'Dog']

In [9]: l.remove('z')

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[9], line 1
----> 1 l.remove('z')

ValueError: list.remove(x): x not in list

In [11]: l=['Apple','Ball','Cat','Dog','Cat']
         l.pop()

Out[11]: 'Cat'

In [12]: l

Out[12]: ['Apple', 'Ball', 'Cat', 'Dog']

In [13]: l=['Apple','Ball','Cat','Dog','Cat']
         l.remove('Cat')
         # It will not return

In [14]: l=['Apple','Ball','Cat','Dog','Cat']
         l.pop() # It will return

Out[14]: 'Cat'

In [19]: l=['Apple','Ball','Cat','Dog','Cat']
         cat_index=l.index('Cat')
         l.pop(cat_index)
         l
```

```
Out[19]:  ['Apple', 'Ball', 'Dog', 'Cat']
```

```
In [18]:  l1=[1,2,3,4,5,6,7,8,9,10,11,2,5,7,6,2,3]
          i1=l1.index(2)
          i2=l1.index(2,1+i1)
          l1.pop(i2)
          l1
```

```
Out[18]:  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 5, 7, 6, 2, 3]
```

```
In [23]:  l1=[1,2,3,4,5,6,7,8,9,10,11,2,5,7,6,2,3]
          i1=l1.index(2)
          i2=l1.index(2,1+i1)
          i3=l1.index(2,1+i2)
          l1.pop(i3)
          l1
```

```
Out[23]:  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 2, 5, 7, 6, 3]
```

- remove will expect a value (direct element)
- pop will expect an index of the value
- if duplicate values are there, then remove will delete the first value only
- if no index given in pop , it will remove the last element by default
- if no value present remove will give value error
- if no valid index pop will give index error

del

```
In [24]:  l=['Apple','Ball','Cat','Dog','Cat']
          del(l)
```

```
In [25]:  l
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[25], line 1
----> 1 l

NameError: name 'l' is not defined
```

```
In [26]:  l=['Apple','Ball','Cat','Dog','Cat']
          del(l[0])
          l
```

```
Out[26]:  ['Ball', 'Cat', 'Dog', 'Cat']
```

```
In [ ]:  - append / extend/ concat / insert

         - remove/pop/del

         - index
- count

- reverse

- sorted

- clear/copy
```

In [ ]:
```
- intialization

- type

- min

- max

- len

- sorted

- reveresd

- in

- in for loop

- index

- index for loop

- mutable immutable

- slice

- concat

- methods
```

In [ ]:
```
# Assignment on tuple and set
# string : I explained
# List : You(99%)+ Omkar sir(1%)
# Tuple : you(100%)
```

In [27]:
```
#dir('')
#dir([])
dir(())
```

```
Out[27]: ['__add__',
          '__class__',
          '__class_getitem__',
          '__contains__',
          '__delattr__',
          '__dir__',
          '__doc__',
          '__eq__',
          '__format__',
          '__ge__',
          '__getattribute__',
          '__getitem__',
          '__getnewargs__',
          '__getstate__',
          '__gt__',
          '__hash__',
          '__init__',
          '__init_subclass__',
          '__iter__',
          '__le__',
          '__len__',
          '__lt__',
          '__mul__',
          '__ne__',
          '__new__',
          '__reduce__',
          '__reduce_ex__',
          '__repr__',
          '__rmul__',
          '__setattr__',
          '__sizeof__',
          '__str__',
          '__subclasshook__',
          'count',
          'index']
```

```
In [ ]:
```