```
In [1]:

                     for i,j in
                     zip(list1,list2):
In [ ]: In [2]:      print(i,j)

                     Ram 25
                     Raheem 30
                     Robert 35


                     #{key:value}

In [ ]: In [3]:
                     d1={'Ram':25,
                     'Raheem':30,
list1=['Ram','Rahee  'Robert':35}

m','Robert']         d1

list2=[25,30,35]     #keys=
                     'Ram','Raheem','Rob
                     ert'
'Ram has age 25'     #values=25,30,35
'Raheem has age 30'
'Robert has age 35'

Out[3]: {'Ram': 25, 'Raheem': 30, 'Robert': 35}
                        #values:
                        'Ram','Raheem','Robert
In [4]:                 '
d2={25:'Ram',30:'Rahee
m',35:'Robert'} d2

#keys: 25,30,35

Out[4]: {25: 'Ram', 30: 'Raheem', 35: 'Robert'}
             'odd':[3,5,7]
             }
In [5]:
d3={'even':[2  d3
,4,6],

Out[5]: {'even': [2, 4, 6], 'odd': [3, 5, 7]}
                    fail

In [ ]: In [8]:
                    d4={(2,4,6):'even'
                    ,
                     (3,5,7):'odd'}
d4={[2,4,6]:'even'
,                   d4
 [3,5,7]:'odd'} #

Out[8]: {(2, 4, 6): 'even', (3, 5, 7): 'odd'}
In [9]:                 its':'Apple'}} d5
# {'key':<{}>}
d5={'item_list':{'frui

Out[9]: {'item_list': {'fruiits': 'Apple'}}


In [10]:
```

```
---------------------------------- -
TypeError Traceback (most recent call las
t)
Cell In[10], line 1
----> 1
d6={{'fruiits':'Apple'}:'item_list'}
 2 d6

TypeError: unhashable type: 'dict'
```

```
a=[1,2]
b=(1,2)
```

```
In [11]: In [18]:
```

```
a,b=[1,2]
c,d=(1,2)
d
```

```
d6={{'fruiits':'Apple'}:'item_list'}
d6

------------------------------------------
```

```
Out[18]: 2
       [19]:
       a=1,2 a
In

Out[19]: (1, 2)
                d1={'A':1,'B':2
                ,'A':1} d1
In [20]:

Out[20]: {'A': 1, 'B': 2}
                d1={'A':1,'B':2
                ,'A':3} d1
In [22]:

Out[22]: {'A': 3, 'B': 2}
```

Dictionary is a key:value pair
at values postition you can take any data type
at keys postion list and dictionary will fail
Duplicates are not allowed
If you will update a key value, latest value it will take

��������

```
In [23]: type(d1)

Out[23]: dict
```

str
list
dict
int
float
bool
complex

tuple

set

```
                              max(d1)
In [24]:
d1={'Ram':25,                 # this maximum we are getting
'Raheem':30,                  based on key or value
'Robert':35}

Out[24]: 'Robert'
                ,
                'Robert':3}
In [25]:
d1={'Ram':2
5,              max(d1)
'Raheem':30

Out[25]: 'Robert'
```

Maximum and minimum value based on key only

```
                 ord('m')
                 'Raheem',
In [26]:         ord('h')
min(d1)          'Robert'

'Ram',

Out[26]: 'Raheem'
             len(d1)
In [27]:

Out[27]: 3
                                                   [28]:
                                               In  d1

Out[28]: {'Ram': 25, 'Raheem': 30, 'Robert': 3}
In [29]: In [32]:
                                                    ---------------------------------- -
                                                    TypeError Traceback (most recent call las
                                                    t)
                                                    Cell In[29], line 1
                                                    ----> 1 sum(d1)

                                                    TypeError: unsupported operand type(s)
                                                    for +: 'int' and 'str'


                                                    d2={100:'2',300:'4'}
                                                    sum(d2)
sum(d1)

----------------------------------------

Out[32]: 400
```

if keys has numeric then we can do sum

```
In [35]: In [40]:
```

```
'a'*'b'

--------------------------------------------------------------------------- -
TypeError Traceback (most recent call las t)
Cell In[35], line 1
----> 1 'a'*'b'

TypeError: can't multiply sequence by non-int of type 'str'
```

type

len

max

min

sum

����

```
d1={'Ram':25,
'Raheem':30,
'Robert':35}


#'Ram':25 in d1 fail

'Ram' in d1 # works

#25 in d1 # Fails
```

```
Out[40]: True
```

```
In [39]: In  Raheem
             Robert


             str1='apple'
             'a' in str1

             l1=[1,2,3]
[38]:        1 in l1

             for i in l1:
             print(i)

             1
             2
             3


             ������
             ����


             l1=[10,20,30
             ,40] l1[0]

             str1='apple'
In [42]:     str1[0]
for i in d1:
print(i)

Ram

Out[42]: 'a'

                   d1['Ram']
In [44]:
d1={'Ram':25,      # can we get values
'Raheem':30,       using for loop
'Robert':35}

Out[44]: 25
                             The age of Raheem is 30
                             The age of Robert is 35
In [49]: In [51]:

                             for i in range(len(l1)):
                              print(i,l1[i])

                             0 10
                             1 20
                             2 30
                             3 40
for key in d1:
 print("The age of {} is
{}".format(key,d1[key]))

The age of Ram is 25
In [52]:
```

```
0
1
2
```

Creating a empty dictionary and update

```
s=''
```

```
In [54]: In [55]:
```

```
for i in 'apple':
 s=s+i

print(s)

apple

l=[i for i in range(10)]
```

```
for key in range(len(d1)):
 print(key)
 #print("The age of {} is
{}".format(key,d1[key])) # possibile / not
pos
```

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

l
```

```
Out[55]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
In [56]:
d1={}
         d1['fruite']='
         Apple' d1
```

```
Out[56]: {'fruite': 'Apple'}
```

```
In [57]:
d1={}
d1["First
name"]="Nikita"
d1["sir
          name"]="Parolekar
          "
          d1["Nativeplace"]
          ="Dhule"
          d1["State"]="Maha
          rashtra" d1
```

```
Out[57]: {'First name': 'Nikita',
          'sir name': 'Parolekar',
          'Nativeplace': 'Dhule',
          'State': 'Maharashtra'}
```

```
 In [ ]:
 # WAP create a dictionary
 based on two lists #
 names=['Ram','Raheem','Rob
 ert'] # age=[25,30,35]
 #
```

```
d1={'Ram':25,'Raheem':30,'
Robert':35}

names=['Ram','Raheem','Rob
ert']
age=[25,30,35]
```

```
In [58]:
names=['Ram','Raheem','Robert']
age=[35,25,30]
dict1={}
for i,j in zip(names,age):
 dict1[i]=j
dict1
```

```
###################################
##########################
dict1={name:age for name,age in
zip(names,age)}
```

```
Out[58]: {'Ram': 35, 'Raheem': 25, 'Robert': 30}
```

```
In [73]: In [68]:
```

```
########################################
################################
dict1={names[i]:age[i] for i in
range(len(names))}


Ram 25
Raheem 30
Robert 35


list1=[]
for i in range(len(names)):
 list1.append(i)
```

```
dict1={}
for i in range(len(names)):
 dict1[names[i]]=age[i]
```

```
list1=[i for i in range(len(names))]
list1
```

```
Out[68]: [0, 1, 2]
```

```
In [ ]:
```

```
{'even':[20,22],'odd':[19,2

1,23]}

# step-1: take empty
dictionary
# step-2: take two even and
odd list # step-3: import
random
# step-4: for i in
range(5):
# step-5:
num=random.randint(a,b) #
step-6: if <even>:
# step-7: append the values
in even list # step-8:
else:
# step-9: append the values
in odd list # step-10:
create dictinary
```

```
In [1]: In [2]:
```

```
#WAP take 5 random numbers
# and create a dictionary

with even and odd # Output:
```

```
import os

os.getcwd()
```

```
Out[2]: 'C:\\Users\\omkar\\Documents'
```

dictionary methods

```
In [3]:  #
dir({})  list=[]
         #
# str='' dict={}

Out[3]: ['__class__',
          '__class_getitem__',
          '__contains__',
          '__delattr__',
          '__delitem__',
          '__dir__',
          '__doc__',
          '__eq__',
          '__format__',
          '__ge__',
          '__getattribute__',
          '__getitem__',
          '__getstate__',
          '__gt__',
          '__hash__',
          '__init__',
          '__init_subclass__',
          '__ior__',
          '__iter__',
          '__le__',
          '__len__',
          '__lt__',
          '__ne__',
          '__new__',
          '__or__',
          '__reduce__',
          '__reduce_ex__',
          '__repr__',
          '__reversed__',
          '__ror__',
          '__setattr__',
          '__setitem__',
          '__sizeof__',
          '__str__',
          '__subclasshook__',
          'clear',
          'copy',
          'fromkeys',
          'get',
          'items',
          'keys',
          'pop',
          'popitem',
          'setdefault',
          'update',
          'values']
```

���������� - �������� - ������������

```
In [4]:    'Robert':3
d1={'Ram': 5}
25,
'Raheem':3 d1
0,

Out[4]: {'Ram': 25, 'Raheem': 30, 'Robert': 35}
            items=d1.ite
            ms() items
In [8]:
#items

Out[8]: dict_items([('Ram', 25), ('Raheem', 30), ('Robert', 35)])
          type(items
          )
In [9]:

Out[9]: dict_items
           keys=d1.key
           s() keys
In [10]:
#keys
```

```
Out[10]: dict_keys(['Ram', 'Raheem', 'Robert'])
          type(keys
          )
In [11]:

Out[11]: dict_keys
              values=d1.val
              ues() values
In [12]:
# values

Out[12]: dict_values([25, 30, 35])
           type(value
           s)
In [13]:

Out[13]: dict_values
           5]
           l1.append(4
In [15]:   00) l1
l1=[25,30,3

Out[15]: [25, 30, 35, 400]
                                     values_list=list(values) # then
                                     you can apply list methods
In [20]:                             values_list
values # I want to convert into a
list

Out[20]: [25, 30, 35]
              (keys)
              keys_list
In [21]:
keys_list=list

Out[21]: ['Ram', 'Raheem', 'Robert']
In [ ]:                    # can you extract keys ans
                           values in a list



                           # I will give two list
                           keys and values # can you
                           create a dictionary

In [22]: In [24]:
                           d1={'Ram':25,
                           'Raheem':30,
                           'Robert':35}

                           keys=list(d1.keys())
                           values=list(d1.values())

# I will give the
dictionary                 keys,values

Out[24]: (['Ram', 'Raheem', 'Robert'], [25, 30, 35])
                      {i:j for i,j in
                      zip(keys,values)}
In [25]:

Out[25]: {'Ram': 25, 'Raheem': 30, 'Robert': 35}
                                       d1[keys[i]]=values[i] #
                                      d1[keys[0]]=values[0] d1['Ram']=25
In [27]:                              d1
d1={}
for i in range(len(keys)):
```

```
Out[27]: {'Ram': 25, 'Raheem': 30, 'Robert': 35}
                   dict(zip(keys,v
In [28]:           alues))

Out[28]: {'Ram': 25, 'Raheem': 30, 'Robert': 35}

In [ ]: In [32]:                            # {'first_name':['virat','Rohit','KL'],
                                           #
                                           'second_name':['kohli','sharma','rahul'
                                           ],
                                           # 'company':['blr','mumbai','lucknow']}


s1='virat.kohli@blr.com,Rohit.sharma@mu s1.split(',')
mbai.com,kl.rahul@lucknow.com'

 Out[32]: ['virat.kohli@blr.com', 'rohit.sharma@mumbai.com', 'kl.rahul@lucknow.com']
In [44]:                                   ,str1.find('.')+1)] for str1 in s
str1='virat.kohli@blr.com'                 f_name,s_name,c_name
d1={}
f_name=[str1[:str1.find('.')] for str1 in  d1['first_name']=f_name
s1.split(',')]                             d1['second_name']=s_name
s_name=[str1[str1.find('.')+1:str1.find('@'d1['company_name']=c_name
)] for str1 in s1.split(',')]
c_name=[str1[str1.find('@')+1:str1.find('.'d1

Out[44]: {'first_name': ['virat', 'rohit', 'kl'],
          'second_name': ['kohli', 'sharma', 'rahul'],
          'company_name': ['blr', 'mumbai', 'lucknow']}
                                            first_name.append(i[0:i.find('.')])

In [30]:
s1='virat.kohli@blr.com,rohit.sharma@mu second_name.append(i[i.find('.')+1:i.fi
mbai.com,kl.rahul@lucknow.com'          nd('@')])
l1=s1.split(',')                        company.append(i[i.find('@')+1:i.find('
first_name=[]                           .',i.find('.')+1)])
second_name=[]                          d2['first_name']=first_name
company=[]                              d2['second_name']=second_name
d2={}                                   d2['company']=company
for i in l1:                            d2
 #print(i)

Out[30]: {'first_name': ['virat', 'rohit', 'kl'],
          'second_name': ['kohli', 'sharma', 'rahul'],
          'company': ['blr', 'mumbai', 'lucknow']}
```

����������-��������������������

```
                                        iterate through loop ==== > each
                                        word will print # step-3:
 In [ ]:                                list1.count(<word>): number
 str1='can can you canner can you       # step-4: make the dictionary
                                        In [46]:
 able to can canner'                    str1='can can you canner can you
                                        able to can canner.'
 #{'can':4,'you':2,'canner':2,'abl      l1=str1.split(' ') # step-1
                                        d1={}
 e':1,'to':1}
                                        for i in l1: # step-2
 # d={}                                  d1[i]=l1.count(i) # step-3
 # step-1: split the str1 =======
 > you will get a list # step-2:
```

```
 d1

Out[46]: {'can': 4, 'you': 2, 'canner': 2, 'able': 1, 'to': 1}
                                 you able to can canner.'
                                 str1.split()
In [47]:
str1='can can you canner can

Out[47]: ['can', 'can', 'you', 'canner', 'can', 'you', 'able', 'to', 'can', 'canne
         r.']
                 values=list(d1.v

In [55]: In
                 alues())

[56]: In [58]:
                 keys,values

keys=list(d1.key


s())


Out[58]: (['can', 'you', 'canner', 'able', 'to'], [4, 2, 2, 1, 1])
             )

In [60]: In
             values.index

[61]:
             (i)

i=max(values


Out[61]: 0
                     keys[values.index(ma
                     x(values))]
In [63]:

Out[63]: 'can'
                       :240}

In [ ]: In [ ]: In

                       d1={'a':20,'b':30,'c':4
                       0,'d':500}
                       d2={'a':50,'b':100,'c':
                       200}

[64]:
                       #
                       o/p={'a':70,'b':130,'c'
                       :240,'d':500}


d1={'a':20,'b':30,'c':4 d1 =
0}                     {'a':20,'b':30,'c':40,'
d2={'a':50,'b':100,'c': d':500} d2 =
200}                   {'a':50,'b':100,'c':200
                       } for i in d2:
#                       d1[i]+=d2[i]
o/p={'a':70,'b':130,'c' d1

Out[64]: {'a': 70, 'b': 130, 'c': 240, 'd': 500}
In [65]:
         d1 = {'a':20,'b':30,'c':40}
         d2 = {'a':50,'b':100,'c':200}
```

```
          for i in d1:
           d1[i]+=d2[i]
          d1

Out[65]: {'a': 70, 'b': 130, 'c': 240}
                                        d3[i]=d1[i]+d2[i]
                                        # d3['a']=d1['a']+d2['a']
In [67]:
d1={'a':20,'b':30,'c':40}
d2={'a':50,'b':100,'c':200}            d3

d3={}                                  # which ever is max length
if len(d1)==len(d2):                   iterate through that dictionary
 for i in d1:

Out[67]: {'a': 70, 'b': 130, 'c': 240}

                                   d1[list(d1.keys())[i]]+=d2[li
In [68]:                           st(d2.keys())[i]] d1
for i in
range(min(len(d1),len(d2))):

Out[68]: {'a': 70, 'b': 130, 'c': 240}
                                   for i in
In [69]:                           range(min(len(d1),len(d2))):
d1 =
{'a':20,'b':30,'c':40,'d':500 d1[list(d1.keys())[i]]+=d2[li
}                                  st(d2.keys())[i]] d1
d2 = {'a':50,'b':100,'c':200}

Out[69]: {'a': 70, 'b': 130, 'c': 240, 'd': 500}
                            python
In [ ]: In [ ]:             # statistics will
                            start (10days) # EDA
                            with python



In [ ]: In [71]:


                            d1 =
                            {'a':20,'b':30,'c':40,
In [72]:                    'd':500} d1['a']=200
# strings
/list/dictionary  #
Python developer            d1


# this week finish

Out[72]: {'a': 200, 'b': 30, 'c': 40, 'd': 500}
In [81]:            s+=i
s=''
for i in            s
reversed('123'):

Out[81]: '321'
       [80]:
       int(s)
In

Out[80]: 321
                      s1='hai how kumar'
In [93]:              l=s1.split(' ')
                      l1=[]
```

```
for i in l:         ize())
                    ' '.join(l1)
l1.append(i.capital

Out[93]: 'Hai How Kumar'
  In [1]: dir({})

 Out[1]: ['__class__',
          '__class_getitem__',
          '__contains__',
          '__delattr__',
          '__delitem__',
          '__dir__',
          '__doc__',
          '__eq__',
          '__format__',
          '__ge__',
          '__getattribute__',
          '__getitem__',
          '__getstate__',
          '__gt__',
          '__hash__',
          '__init__',
          '__init_subclass__',
          '__ior__',
          '__iter__',
          '__le__',
          '__len__',
          '__lt__',
          '__ne__',
          '__new__',
          '__or__',
          '__reduce__',
          '__reduce_ex__',
          '__repr__',
          '__reversed__',
          '__ror__',
          '__setattr__',
          '__setitem__',
          '__sizeof__',
          '__str__',
          '__subclasshook__',
          'clear',
          'copy',
          'fromkeys',
          'get',
          'items',
          'keys',
          'pop',
          'popitem',
          'setdefault',
          'update',
          'values']
```

����������-��������

In [3]: In [6]:

```python
# take one dictionary
# take second dictionary
# copy first dict into second
# clear first dict
# and print both

d1 = {'a':20,'b':30,'c':40,'d':500} #
d2=d1.copy()
d1.clear()
print(d1) # {}
print(d2) # {'a':20,'b':30,'c':40,'d':500}
```

```
{}
{'a': 20, 'b': 30, 'c': 40, 'd': 500}
```

������-��������������-������

what is pop
pop will remove specify key
return value
what is popitem

```
- remove the last value default

- return the pair
```

what is del

- del is a keyword

- it can delete an item by providing specific key d2

```
Out[6]: {'a': 20, 'b': 30, 'c': 40, 'd': 500}
          d2.pop('a
          ')
In [7]:

Out[7]: 20
          d2.popitem
          ()
In [9]:

Out[9]: ('d', 500)
             #del is a
In [13]: In  keyword del
             d2['c']


[14]:        d2




Out[14]: {'b': 30}
In [10]: In [11]:     access by index l1



l1=[10,20,30,40,50]
del l1[1] # list is


Out[11]: [10, 30, 40, 50]

  In [15]: In [16]:
```

```
In [17]: In [19]:
```

```
del(d2)


d2

--------------------------------------------------------------------------- -
NameError Traceback (most recent call las t)
Cell In[16], line 1
----> 1 d2

NameError: name 'd2' is not defined
```

```
d1={'a':1,'b':2,'c':3}
################################
del d1['c']

d1.pop('c')

d1.popitem() # LIFO
########################################
del (d1)
```

items/values/keys

if i provide two list you know how to make dict

if i provide dict , you know how to genertae list
create an empty string/list/dictionary
counter
summ
function
pop/popitem/del
copy/clear

����������������-������

```
d1 = {'a':20,'b':30,'c':40,'d':500}

d1['b']
```

```
Out[19]: 30
```

```
In [20]: d1.get('b'
         )
```

```
Out[20]: 30
```

```
In [21]: In [22]:
```

```
------------------------------------ -
KeyError Traceback (most recent call las
t)
Cell In[22], line 1
----> 1 d1['z']

KeyError: 'z'
```

how many ways we can do
what output it is returning
what is the difference
we can access the values by providing key as index and key in get method but if the key is not present in dictionary , key as index will give key error get method will not retutrn anything and no error

```
# it creates a new dictionary

# create a new dictionary from the given

sequence of elements # create new

dictionary with irerable key
```

```
In [ ]: In [34]:
```

```
d1.get('z')

d1['z']
```

```
------------------------------------------
```

```
d2={}.fromkeys('Raheem',25)
d2
```

```
Out[34]: {'R': 25, 'a': 25, 'h': 25, 'e': 25, 'm': 25}
```

```
In [25]:
d1
```

```
Out[25]: {'a': 20, 'b': 30, 'c': 40, 'd': 500}
            name='sourav'
In [28]: In new=name.capita

            lize() name,new
[29]:

Out[29]: ('sourav', 'Sourav')
                d3={}.fromkeys([1,3
In [31]:        ,5],'odd') d3

Out[31]: {1: 'odd', 3: 'odd', 5: 'odd'}
In [32]:                  ��������������
                          ��

                          returns value of the key
                          if key is not present
                          returns default

                          d1={}
In [ ]: In [43]:          d1.setdefault('nareshit','D
                          s')
                          d1.setdefault('city')
                          d1.setdefault('nareshit',['
for i in [1,3,5]:         DS','MLops','DE'])
 print(i)                 d1['nareshit']=['DS','MLops
                          ','DE']
1                         d1.setdefault('city','hyd')
3                         d1
5

������ ����

Out[43]: {'nareshit': ['DS', 'MLops', 'DE'], 'city': None}

In [39]:

                          In [ ]: In [44]:

                          d2={}
```

```
d2['nareshit']='DS'
d2['city'] #
```

```
---------------------------------------------------------------------------
KeyError Traceback (most recent call las
t)
Cell In[39], line 3
 1 d2={}
 2 d2['nareshit']='DS'
----> 3 d2['city']

KeyError: 'city'
```

```
# it inserts the particular items to the
dictionary

d1 = {'a': 1, 'b': 4, 'c': 5}
d2 = {'b': 7, 'd': 9}
d1.update(d2)
print(d1)

{'a': 1, 'b': 7, 'c': 5, 'd': 9}

{'a': 1, 'b': 7, 'c': 5, 'd': 9}
```

```
�������������

Out[44]: {'a': 1, 'b': 7, 'c': 5, 'd': 9}
In [48]: In [47]:    a:1 c:5 {'b': 4,

                     'd': 9, 'a': 1, 'c':

                     5}
```

```
d1 = {'a': 1, 'b':
4, 'c': 5} d2 =
{'b': 7, 'd': 9}
d2.update(d1)
print(d2) # b:4 d:9
```

```
l1=[1,4,5]
l2=[4,9]
l1.extend(l2)
l1
```

```
Out[47]: [1, 4, 5, 4, 9]
```

Update method will update a dictionary with elements from another dictionary.
It modifies the dictionary.

```
In [50]:
t1=[(1,2),(2,3)]
#dict/iterable d1={}
d1.update(t1)
d1

Out[50]: {1: 2, 2: 3}
```

```
In [ ]:
```

```
In [ ]: In [ ]:
```

- strings

- list
- dictionary

if else

for

print

append = []

- tuple

write an article on tupe

- how to read
- type/max/min/len/sum
- index/mutable
- in /for
- slice
- methods

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```