In [ ]: In [6]:

```
                    # we are splitting
                    # the output inthe
                    form list
                    str1='hai how are
List                you'
tuple               l1=str1.split()
string              l1
Dictionary
```

Out[6]: ['hai', 'how', 'are', 'you']

```
                        elements are there # we
                        want to join
In [7]:                 l1
# Now we have list of
```

Out[7]: ['hai', 'how', 'are', 'you']

```
          '
          '.join(l1)
In [12]:
```

Out[12]: 'hai how are you'

```
           for i in
           l1:  s=s+i
In [14]:   s
s=''
```

Out[14]: 'haihowareyou'

```
                            zip(names,age):
In [15]: In [21]:            print(f"{i} age is {j}")
```

```
                            Ramesh age is 20
                            Suresh age is 25
                            Sathish age is 30
                            In [23]: In [19]:
```

```
names=['Ramesh','Suresh','S
athish'] age=[20,25,30]

# Ramesh age is 20
# Suresh age is 25
# Sathish age is 30
for i in range(len(names)):
 print(names[i],age[i])

Ramesh 20
Suresh 25
Sathish 30

??????

#for i in names : i = Rames
i=sure i= sath #for i in
age : i 25 30 for i,j in
```

In [ ]: In [ ]:

In [ ]: In [24]:

```
num1,num2=10,20
print(num1)
print(num2)

10
20

def avg(n1,n2,n3):
 return(n1+n2+n3,
(n1+n2+n3)/3)
```

```
o1,o2=avg(10,20,30)
o1
o2

60
20.0

num1,num2=10,20
o1,o2=avg(10,20,30)
i,j in zip(names,age)

names=['Ramesh','Suresh','S
athish']
age=[20,25,30]

If both are together why
```

two lists ===== Dictionary

{<key>:<value>}

```
d1={"Ramesh":20,"Suresh":25
,"Sathish":30} d1

# Ramesh
# Suresh
# Sathish are keys

# 20 25 30 are values

# key:value pair together
```

```
Out[24]: {'Ramesh': 20, 'Suresh': 25, 'Sathish': 30}
         type(d1)

 In [25]:

Out[25]: dict
                        between
                        lists,
                        tuple,
In [ ]: In [26]:        set, and
                        dictionary?


                        d2={20:"Ramesh",25:"Sures
                        h",30:"Sathish"} d2
how can we distinguish

Out[26]: {20: 'Ramesh', 25: 'Suresh', 30: 'Sathish'}
In [27]:  } d3
d3={20:20

Out[27]: {20: 20}
           d4={'$':'30
           '} d4
In [34]:

Out[34]: {'$': '30'}
                        d1={"Ramesh":20,"Suresh":
                        25,"Sathish":30}
In [ ]:                 d2={20:"Ramesh",25:"Sures
                        h",30:"Sathish"}
                        d3={20:20}
                        d4={'$':'30'}

In [ ]: In [37]:
                        d5={$:'30'} # Fail

                        # spl cahracters should
                        not be a key


                        d6={'30':$}
                        d6

                        Cell In[37], line 1
                         d6={'30':$}
In [35]:                 ^
```

```
SyntaxError: invalid      }
syntax                    d7
                          # List can be value


d7={'names':['R','S','A']

Out[35]: {'names': ['R', 'S', 'A']}

  In [36]: In [38]:
```



```
  d8={['R','S','A']:'names'}
  d8

  # List can not be a key

  ---------------------------------------------------------------------------
  TypeError Traceback (most recent call las t)
  Cell In[36], line 1
  ----> 1 d7={['R','S','A']:'names'}
   2 d7

  TypeError: unhashable type: 'list'


  d9={('R','S','A'):'names'}
  d9

Out[38]: {('R', 'S', 'A'): 'names'}
In [39]:           'S','A')} d10
d10={'names':('R',

Out[39]: {'names': ('R', 'S', 'A')}
                     d11={('I','N','D'):('
                     R','S','A')} d11
In [40]:

Out[40]: {('I', 'N', 'D'): ('R', 'S', 'A')}
                pple'}
                d12={"Names":d}
In [41]:        d12
d={'Fruites':'A

Out[41]: {'Names': {'Fruites': 'Apple'}}


In [42]:
```

```
---------------------------------- -
TypeError Traceback (most recent call las
t)
Cell In[42], line 2
 1 d={'Fruites':'Apple'}
----> 2 d13={d:'names'}
 3 d13

TypeError: unhashable type: 'dict'
```

```
# dictionary and list can not be
represent as keys
```

In [ ]: In [ ]:

```
################## Pass
##################
d1={"Ramesh":20,"Suresh":25,"Sathish":30}
d2={20:"Ramesh",25:"Suresh",30:"Sathish"}
d3={20:20}
d4={'$':'30'}
d5={'names':['I','N','D']}
d6={'names':('I','N','D')}
d7={('I','N','D'):'names'}
d8={"Names":{'Fruites':'Apple'}}
```

In [ ]: In [45]:

```
############### Fail ###################
d9={$:'30'}
d10={['I','N','D']:'names'}
d11={{'Fruites':'Apple'}:"Names"}
```

```
d={'Fruites':'Apple'}
d13={d:'names'}
d13
```

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

```
d3={True:True}
d3
```

```
Out[45]: {True: True}
```

In [46]:

```
d1={"Ramesh":20,"Ram # Duplicate keys are
esh":25} d1          not allowed
```

```
Out[46]: {'Ramesh': 25}
```

```
                    resh":20} d2
                    # Duplicate values
In [47]:            are allowed
d2={"Ramesh":20,"Su
```

```
Out[47]: {'Ramesh': 20, 'Suresh': 20}
```

max
min
len
sum

Watch only keys

In [48]:

```
d1={"Ramesh":20,"Suresh":
25,"Sathish":30} max(d1)
```

```
                        # Are you seeing values
# Are you seeing keys

Out[48]: 'Suresh'
                        d2={20:"Ramesh",25:"Sures
                        h",30:"Sathish"} max(d2)
In [50]:

Out[50]: 30
                                                 max(d2)

In [ ]: In [ ]:                                  ----------------------------------------
                                                 ---------------------------------- -
                                                 TypeError Traceback (most recent call las
                                                 t)
                                                 Cell In[51], line 2
In [51]:                                          1 d2={20:"Ramesh",'Suresh':30}
d2={20:"Ramesh",'Suresh':30} # Fail              ----> 2 max(d2)

max([25,30,35]) # 35                             TypeError: '>' not supported between
max(['A','B','C']) # 'c'                         instances of 'str' and 'int'
max([25,'A']) # error


d2={20:"Ramesh",'Suresh':30}
In [52]:                 25,"Sathish":30} min(d1)
d1={"Ramesh":20,"Suresh":


Out[52]: 'Ramesh'
                        d2={20:"Ramesh",25:"Sures
                        h",30:"Sathish"} min(d2)
In [53]:

Out[53]: 20
                        d1={"Ramesh":20,"Suresh":
                        25,"Sathish":30} len(d1)
In [54]:


Out[54]: 3
          sorted(d1
          ) #
In [55]:

Out[55]: ['Ramesh', 'Sathish', 'Suresh']

In [57]: In [58]:                                Sathish
                                                 Suresh
                                                 Ramesh


                                                 d1={"Ramesh":20,"Suresh":25,"Sathish":30}
                                                 sum(d1)

                                                 ----------------------------------------
                                                 ---------------------------------- -
                                                 TypeError Traceback (most recent call las
                                                 t)
                                                 Cell In[58], line 2
                                                  1
                                                 d1={"Ramesh":20,"Suresh":25,"Sathish":30}
                                                 ----> 2 sum(d1)

In [59]:                                         TypeError: unsupported operand type(s)
d1={"Ramesh":20,"Suresh":25,"Sathish":30}        for +: 'int' and 'str'
for i in reversed(d1):
 print(i)
```

```
                                                    sum(d2)

d2={20:"Ramesh",25:"Suresh",30:"Sathish"}
```

```
Out[59]: 75
```

```
                    ','Sathish']
                    l2=[20,25,30]
In [61]: In [62]:

                    sum(l2)
                    sum(l1) # fail
l1=['Ramesh','Suresh
```

```
Out[62]: 75
```

```
 In [ ]: In [ ]: In [ ]: In [ ]:          len(d2) # 2


                                          - how to read

                                          - different ways to read

                                          - type / min/ max/
                                          len/sum/sorted/reversed

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

```
                                          d1={"Ramesh":20}
                                          d2={'Suresh':30}
                                          d1+d2
```

```
                                          ---------------------------------------
                                          ------------------------------- -
                                          TypeError Traceback (most recent call las
                                          t)
                                          Cell In[1], line 3
                                           1 d1={"Ramesh":20}
                                           2 d2={'Suresh':30}
                                          ----> 3 d1+d2
```

```
 d1={"Ramesh":20,"Suresh":25,"Sathish":30} TypeError: unsupported operand type(s)
 sum(d1) # Error                          for +: 'dict' and 'dict'
 max(d1) # Sure
 min(d1) # Rame
 len(d1) # 3

                                          d1*2 # d1+d1
 d2={20:"Ramesh",25:"Suresh",30:"Sathish"} ---------------------------------------
 sum(d2) # 75                             ------------------------------- -
 max(d1) # 30                             TypeError Traceback (most recent call las
 min(d1) # 20                             t)
 len(d1) # 3                              Cell In[2], line 1
                                          ----> 1 d1*2

 d2={20:"Ramesh",'Suresh':30}
 max(d2) # E                              TypeError: unsupported operand type(s)
 min(d2) # E                              for *: 'dict' and 'int'
 sum(d2) # E
```

```
        ����
 In [8]:                  :25,"Sathish":30} # Keys
 # Keys are important     are important
 d1={"Ramesh":20,"Suresh" 'Ramesh' in d1 # True
```

```
20 in d1 # False
```

```
Out[8]: False
```

```
In [5]:
```

```
In [16]:
```

```
hish":30}
# Normal index will not work for
dictionary #d1[0] # Error

# If you want retrive the value,
by using key we can get
d1['Ramesh']
d1['Suresh']
d1['Sathish']

d1[i]

for i in d1:
 print(i,d1[i])
```

```
Ramesh 20
Suresh 25
Sathish 30
```

```
In [17]:
```

```
In [19]:
```

```
names=['Ramesh','Suresh','Sathis
h']
age=[20,25,30]
for i,j in zip(names,age):
 print(f"{i} age is {j}")
```

```
Ramesh age is 20
Suresh age is 25
Sathish age is 30
```

```
for i in d1:
 print(i)
```

```
Ramesh
Suresh
Sathish
```

```
d1={"Ramesh":20,"Suresh":25,"Sat
hish":30}
for i in d1:
 print(f"{i} age is {d1[i]}")
```

```
Ramesh age is 20
Suresh age is 25
Sathish age is 30
```

```
d1={"Ramesh":20,"Suresh":25,"Sat
```

```
In [ ]:
```

```
In [25]:
```

```
 print(i) # Only keys


for i in d1:
 print(i,d1[i]) # keys
and values


d1,l1,s1={},[],''
s1=s1+'a' # ''+'a'
s1=s1+'b'
s1=s1+'c'
s1
```

```
d1={"Ramesh":20,"Suresh":
25,"Sathish":30}

for i in d1:
```

```
Out[25]: 'abc'
```

```
   l1=[]
   l1.append(1
```

```
In [28]:
```

```
0)          0) l1
l1.append(2
```

```
Out[28]: [10, 20]
```

```
                30,40]
                d1['odd']=[31,3
In [32]:        3] d1
d1={}
d1['even']=[20,
```

```
Out[32]: {'even': [20, 30, 40], 'odd': [31, 33]}
```

```
                    d1['CITY']=city
                    d1['AGE']=age
In [31]:            d1
d1={}
name=input("enter   enter the
the name:")         name:python
city=input("enter   enter the
the city")          cityHyderabad
age=eval(input("ente enter the age10
r the age"))
d1['NAME']=name
```

```
Out[31]: {'NAME': 'python', 'CITY': 'Hyderabad', 'AGE': 10}
```

```
In [ ]: In [33]:          random number 10,100 #
                         step-5: if <condition>:
                         # step-6: append the values
                         in eve # step-7: else:
                         # step-8: append the values
                         in odd list # step-9:
                         create the dictionary


                         import random
                         even_list,odd_list,d1=[],[]
                         ,{}

                         for i in range(10):
                          num=random.randint(10,100)
# WAP ask the user generate  if num%2==0:
10 random numbers # Find   even_list.append(num)
the even number and odd    else:
number # output:           odd_list.append(num)
#{'Even':[20,30,40],'Odd':[
35,45]}                  d1['Even']=even_list
# step-1: import random    d1['Odd']=odd_list
# step-2:                  d1
even_list,odd_list=[],[]
# step-3: for loop 10 times
# step-4: num=generate
```

```
Out[33]: {'Even': [38, 52], 'Odd': [81, 95, 31, 23, 27, 57, 43, 33]}
```

```
                                        d1={}
                                        for i in l1: # iterator through list
In [43]:                                 if i not in l2:
# word frequency                        d1[i]=l1.count(i) # key=i value=
s1='hello how how are you, hello how im l1.count(i)
good, hello how what are you doing' #    l2.append(i)
{'hello':3,'how':4,......}
# split the data                        d1
l1=s1.split() # list of words
l2=[]
```

```
Out[43]: {'hello': 3,
          'how': 4,
          'are': 2,
```

```
            'you,': 1,
            'im': 1,
            'good,': 1,
            'what': 1,
            'you': 1,
            'doing': 1}
```

In [42]:
```
d1={}
d1['hello']=3
d1[i]=l1.count(i)
```

Out[42]:
```
{'hello': 3}
```

In [46]:
```
d1={'Fruites':['Apple
','Banana']} #
```

```
Retrive Banana
d1['Fruites'][0],d1['
Fruites'][1]
```

Out[46]:
```
('Apple', 'Banana')
```

In [115]:
```
d1={'Fruites':[{'Apple':50
},{'Banana':20}]} len(d1)
```

```
d1['Fruites'] # List
d1['Fruites'][1] # dict
d1['Fruites'][1]['Banana']
```

Out[115]:
```
20
```

In [58]:
```
d1={'Fruites':[{'Apple':[50,150
```

```
,300]},{'Banana':20}]}
d1['Fruites'][0]['Apple'][2]
```

Out[58]:
```
300
```

In [67]:
```
d1={'Fruites':
[{'Apple':[{50:'Norma
```

```
l',  150:'Average',
300:'Good'}]},
{'Banana':20}]}
d1['Fruites'][0]['App
le'][0][150]
```

Out[67]:
```
'Average'
```

In [86]:
```
d1={'Fruites':
 [{'Apple':[{50:
 [
 {'Normal':[100,300,400]  }
 ]
 }
 ]
 }
```

```
 ]
 }
len(d1['Fruites'])
len(d1['Fruites'][0])
len(d1['Fruites'][0]['Apple'
])
len(d1['Fruites'][0]['Apple'
][0])
d1['Fruites'][0]['Apple'][0]
[50][0]['Normal'][2]
```

Out[86]:
```
400
```

```
len(d1)
```

In [87]:

Out[87]:
```
1
```

In [94]:
```
len(d1['Fruites']) # 1 ele
in list index=0
```

```
len(d1['Fruites'][0])
d1['Fruites'][0]['Apple']
```

Out[94]:
```
[{50: [{'Normal': [100, 300, 400]}]}]
```

```
In [95]:                    dir('') dir(())
dir({}) # dir([])

Out[95]: ['__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 [ ]: In [96]:          'get',
                          'items',
                          'keys',
                          'pop',
                          'popitem',
                          'setdefault',
                          'update',
                          'values'

                          ����������-���
                          ����������-��
                          ���������

                          d1={'Ramesh':20,'Suresh'
'clear',                  :25,'Piyush':27} d1
'copy',
'fromkeys',

Out[96]: {'Ramesh': 20, 'Suresh': 25, 'Piyush': 27}
          d1.keys()

In [97]:

Out[97]: dict_keys(['Ramesh', 'Suresh', 'Piyush'])
           d1.values(
           )
In [98]:

Out[98]: dict_values([20, 25, 27])
           d1.items(
           )
In [99]:

Out[99]: dict_items([('Ramesh', 20), ('Suresh', 25), ('Piyush', 27)])
                          type(d1.keys()) #
                          dict_keys
In [103]:                 # so convert dict_keys
d1.keys() # to check the  type to list type
type of output

Out[103]: dict_keys
             list(d1.keys
             ())
In [108]:

Out[108]: ['Ramesh', 'Suresh', 'Piyush']
                    list(d1.keys())[0]
                    .lower()
In [107]:

Out[107]: 'ramesh'
          get
          pop
  In [ ]: popitem
   clear
   copy
```

```
In [ ]: In [110]:          # l2=l1.copy()
                           # l1.clear()
                           # print(l1),print(l2)
                           l1=['A','B','C']
                           l2=l1.copy()
                           l1.clear()
                           l1,l


                           d1={'Ramesh':20,'Suresh':
                           25,'Piyush':27}
                           d1.clear()
                           d1
list1.clear()
list1

# Intialise the list l1

Out[110]: {}
                           d2=d1.copy()
                           d1.clear()
In [111]:                  d2,d1
d1={'Ramesh':20,'Suresh':
25,'Piyush':27}

Out[111]: ({'Ramesh': 20, 'Suresh': 25, 'Piyush': 27}, {})
   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']
          fromkeys
          setdefaul
          t update
```

```
In [ ]:
pop
popitem
```

```
In [4]:
d1={'Ramesh':20,'Suresh'
:25,'Piyush':27}
```

```
                d1.pop('Ramesh') # key
return values d1
```

```
Out[4]: {'Suresh': 25, 'Piyush': 27}
```

```
                :25,'Piyush':27}
                d1.popitem() # LIFO
(key,value) d1
```

```
In [6]:
d1={'Ramesh':20,'Suresh'
```

```
Out[6]: {'Ramesh': 20, 'Suresh': 25}
```

```
          ??????
```

```
                :25,'Piyush':27}
                d1.get('Ramesh')
```

```
In [7]:
d1={'Ramesh':20,'Suresh'
```

```
Out[7]: 20
```

```
                        d1['Ramesh'] # By reference as a
                        key you an get the value
```

```
In [8]:
```

```
Out[8]: 20
```

```
          ??????????????
```

```
          d2={'Suresh'
          :30}
          d1.update(d2
          ) d1
```

```
In [9]:
d1={'Ramesh'
:20}
```

```
Out[9]: {'Ramesh': 20, 'Suresh': 30}
```

```
          d2={'Suresh'
          :30}
          d2.update(d1
          ) d2
```

```
In [10]:
d1={'Ramesh'
:20}
```

```
Out[10]: {'Suresh': 30, 'Ramesh': 20}
```

```
In [ ]:
```