In [7]:
```
# method-1

str1  = 'hello how are you'
# ans = ['Hello', 'How', 'Are', 'You']
l1 = str1.title().split()
l1
```

Out[7]:
```
['Hello', 'How', 'Are', 'You']
```

In [8]:
```
# method-2
ans = []
for i in str1.split():
    ans.append(i.title())
ans
```

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

In [9]:
```
l1 = ['Ramesh','Suresh','Sathish']
l2 = [20,30,40]

# 'Ramesh age is 20'
# 'Suresh age is 30'
# 'Sathish age is 40'
```

In [13]:
```
print(f'{l1[0]} age is {l2[0]}')
print(f'{l1[1]} age is {l2[1]}')
print(f'{l1[2]} age is {l2[2]}')

# what is common ? => (f'{l1[]} age is {l2[]}')
# what is changing? => i
```

```
Ramesh age is 20
Suresh age is 30
Sathish age is 40
```

In [12]:
```
for i in range(len(l1)):
    print(f'{l1[i]} age is {l2[i]}')
```

```
Ramesh age is 20
Suresh age is 30
Sathish age is 40
```

$$zip - method$$

- The zip() function in Python is used to combine multiple iterables (like lists, tuples, etc.) into one iterable, where each element is a tuple containing the elements from the input iterables that are at the same position. It is useful for iterating over multiple sequences in parallel.
- in zip there are two way
  - 1. is you can give two list like l1 and l2 one variable like i in zip
    - ex:- for i in zip(l1,l2): # here we use i only
    - here i become ramesh and i beacame 20
  - 2. is you can give two list like l1,and l2 two variables i,j in zip
    - ex: for i,j in zip(l1,l2): # here we use i,j

In [14]:
```
l1 = ['Ramesh','Suresh','Sathish']
l2 = [20,30,40]
l3 = ['Hyd', 'Mumbai', 'Pune']

for i,j,k in zip(l1,l2,l3):
    print(f'{i} has age is {j} and he is from {k}')
```

```
Ramesh has age is 20 and he is from Hyd
Suresh has age is 30 and he is from Mumbai
Sathish has age is 40 and he is from Pune
```

- two list work in one dictionary
- like l1 and l2 get combine with key value pair
- and form one dictionary dict1

Dictionary

- Dictionary represent in a curly braces with two values
- Dictionary has key:value pairs
  - One element called as **key**
  - another element called as **value**

Always remember dictionary required two values key and value

- if you will pass single value then it will became set

In [ ]:
```
l1 = ['Ramesh','Suresh','Sathish']
l2 = [20,30,40]

{key:value}
{key1:1value,key2:value2,key3:value3}
```

In [15]:
```
dict1 = {'Ramesh':20, 'Suresh':30, 'Sathish':40}
dict1
```

Out[15]:
```
{'Ramesh': 20, 'Suresh': 30, 'Sathish': 40}
```

In [16]:
```
type(dict1)
```

Out[16]:
```
dict
```

- intialization
- inbuilt functions
  - min
  - max

    - type
    - print
    - len
    - sum
    - sorted
    - reversed
- index
- mutable vs immutable
- concatenation
- slicing
- **Methods of Dict**

intialization

In [17]:
```
dict1 = {'Ramesh':20, 'Suresh':30, 'Sathish':40}
dict1
```

Out[17]:
```
{'Ramesh': 20, 'Suresh': 30, 'Sathish': 40}
```

In [18]:
```
d1 = {'Ramesh':20, 'Suresh':30, 'Sathish':40}
d1
```

Out[18]:
```
{'Ramesh': 20, 'Suresh': 30, 'Sathish': 40}
```

In [19]:
```
d2 = {'Ramesh':'20', 'Suresh':'30', 'Sathish':'40'}
d2
```

Out[19]:
```
{'Ramesh': '20', 'Suresh': '30', 'Sathish': '40'}
```

In [20]:
```
d3 = {20:'Ramesh', 30:'Suresh', 40:'Sathish'}
d3
```

Out[20]:
```
{20: 'Ramesh', 30: 'Suresh', 40: 'Sathish'}
```

In [22]:
```
d4 = {'Ramesh':20, 'Ramesh':30}
d4

# keys are important
# When same keys but different value
# It will take the latest value
```

Out[22]:
```
{'Ramesh': 30}
```

In [23]:
```
d5 = {'Ramesh':20, 'Suresh':20}
d5
```

Out[23]: {'Ramesh': 20, 'Suresh': 20}

In [25]:
```
d6 = {'Ramesh':20, 'Ramesh':20}
d6

# Duplicates are not allowed
```

Out[25]: {'Ramesh': 20}

In [26]:
```
d7 = {'fruit':'Apple', 'cost':10.5, 'count':100, 'avg':True}
d7
```

Out[26]: {'fruit': 'Apple', 'cost': 10.5, 'count': 100, 'avg': True}

In [28]:
```
d8 = {True:'Yes'}
d8
```

Out[28]: {True: 'Yes'}

In [30]:
```
d9 = {'age':[10,20,30]}
d9
# List can be a value
```

Out[30]: {'age': [10, 20, 30]}

In [33]:
```
d10 = {[10,20,30]:'age'}
d10
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[33], line 1
----> 1 d10 = {[10,20,30]:'age'}
      2 d10

TypeError: unhashable type: 'list'
```

In [32]:
```
d11 = {'age':(10,20,30)}
d11
```

Out[32]: {'age': (10, 20, 30)}

In [34]:
```
d12 = {(10,20,30):'age'}
d12
```

Out[34]: {(10, 20, 30): 'age'}

IMP

- In list not working but in tuple working why?
- because of tuple nature is immutable
- always remember key is superior than values
- so key will not change
    - for ex:- ramesh will be ramesh only

- but 1 year ago his age is 20 = value
- now he is 21 = value
- so value can be change but key will constant

- sir ans:
  - list is mutable and tuple is immutable
  - so list can not be a key

keys are important

In [35]:
```
d13 = {(10,20,30): ('A','B','C')}
d13
```

Out[35]:
```
{(10, 20, 30): ('A', 'B', 'C')}
```

In [36]:
```
d14 = {(10,20,30): ['A','B','C']}
d14
```

Out[36]:
```
{(10, 20, 30): ['A', 'B', 'C']}
```

In [37]:
```
d15 = {[10,20,30]: ('A','B','C')}
d15
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[37], line 1
----> 1 d15 = {[10,20,30]: ('A','B','C')}
      2 d15

TypeError: unhashable type: 'list'
```

List can't be a kye

In [38]:
```
d16 = {'Fruits':{'Apple':50}}
d16
```

Out[38]:
```
{'Fruits': {'Apple': 50}}
```

In [39]:
```
d17 = {{'Apple':50}:'Fruits'}
d17
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[39], line 1
----> 1 d17 = {{'Apple':50}:'Fruits'}
      2 d17

TypeError: unhashable type: 'dict'
```

inside dictionary{, dictionary not a key becaue of mutable}

In [ ]:
```
d1 = {'Ramesh':20, 'Suresh':30, 'Sathish':40} #w
d2 = {'Ramesh':'20', 'Suresh':'30', 'Sathish':'40'} #W
d3 = {20:'Ramesh', 30:'Suresh', 40:'Sathish'} #W
d4 = {'Ramesh':20, 'Ramesh':30} #W But Latest
d5 = {'Ramesh':20, 'Suresh':20} #W
d6 = {'Ramesh':20, 'Ramesh':20}#W no duplicates
d7 = {'fruit':'Apple', 'cost':10.5, 'count':100, 'avg':True} #W
d8 = {True:'Yes'} # W
d9 = {'age':[10,20,30]} #W
d10 = {[10,20,30]:'age'} #Not Works
d11 = {'age':(10,20,30)} #W
d12 = {(10,20,30):'age'} #W
d13 = {(10,20,30): ('A','B','C')} # W
d14 = {(10,20,30): ['A','B','C']} # W
d15 = {[10,20,30]: ('A','B','C')} # Not W
d16 = {'Fruits':{'Apple':50}} # W
d17 = {{'Apple':50}:'Fruits'} # Not W
```

min and max

In [40]:
```
d1 = {'Ramesh':20, 'Suresh':30, 'Sathish':40}
max(d1)
```

Out[40]: 'Suresh'

In [42]:
```
d1 = {'Ramesh':20, 'Suresh':20, 'Sathish':20}
max(d1)

# Keys are important
# when we are check max and min it will check keys
```

Out[42]: 'Suresh'

In [43]:
```
d1 = {'Ramesh':20, 'Suresh':30, 'Sathish':40}
min(d1)
```

Out[43]: 'Ramesh'

In [44]:
```
d = {10:'a',20:'b', 30:'c'}
max(d), min(d)
```

Out[44]: (30, 10)

sum

In [45]:
```
d1 = {10:'a',20:'b', 30:'c'}
sum(d1)
```

Out[45]: 60

In [46]:
```
d1 = {10:'a',20:'b', 30:'c'}
sum(d1,start=100)
```

Out[46]: 160

In [47]:
```
d1 = {'Ramesh':20, 'Suresh':30, 'Sathish':40}
```

```
sum(d1)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[47], line 2
      1 d1 = {'Ramesh':20, 'Suresh':30, 'Sathish':40}
----> 2 sum(d1)

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

In [49]:
```
d1 = {10:'a',20:'b', 'a':'c'}
sum(d1)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[49], line 2
      1 d1 = {10:'a',20:'b', 'a':'c'}
----> 2 sum(d1)

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

In [50]:
```
30+'a'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[50], line 1
----> 1 30+'a'

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

len

In [52]:
```
d1 = {10:'a',20:'b', 'a':'c'}
len(d1)

# here key value pair lenght will calculated
```

Out[52]: 3

In [3]:
```
d1 = {10:'a',20:'b', 'a':}
len(d1)
```

```
  Cell In[3], line 1
    d1 = {10:'a',20:'b', 'a':}
                             ^
SyntaxError: expression expected after dictionary key and ':'
```

sorted

In [54]:
```
d1 = {'Ramesh':20, 'Suresh':30, 'Sathish':40}
sorted(d1)
```

Out[54]: ['Ramesh', 'Sathish', 'Suresh']

In [55]:
```
d1 = {'Ramesh':20, 'Suresh':30, 100:40}
sorted(d1)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[55], line 2
      1 d1 = {'Ramesh':20, 'Suresh':30, 100:40}
----> 2 sorted(d1)

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

- why error?
- all keys are not same data types some are str and some are int
- so sorted will compare and give answer thats why error

reversed

In [56]:
```
d1 = {'Ramesh':20, 'Suresh':30, 'Sathish':40}
reversed(d1)
```

Out[56]:
```
<dict_reversekeyiterator at 0x19a9c019260>
```

In [57]:
```
list(reversed(d1))
```

Out[57]:
```
['Sathish', 'Suresh', 'Ramesh']
```

- sorted and reversed answer comes in a list form

In [58]:
```
for i in reversed(d1):
    print(i)
```

```
Sathish
Suresh
Ramesh
```

In [59]:
```
d1 = {'Ramesh':20, 'Suresh':30, 100:40}
reversed(d1)
```

Out[59]:
```
<dict_reversekeyiterator at 0x19a9c0186d0>
```

In [60]:
```
list(reversed(d1))
```

Out[60]:
```
[100, 'Suresh', 'Ramesh']
```

- in reversed we got answer because reversed only reverse the keys
- reversed no need to data types
- but sorted will compare with data types

Note

- max/min/sorted can wrok for homogenious
- sum is only work for numbers ony

- reversed can work any one

index

In [61]:
```
l = [10,20,30]
l[0]
```

Out[61]: 10

In [62]:
```
s = 'python'
s[0]
```

Out[62]: 'p'

In [63]:
```
d1 = {'Ramesh':20, 'Suresh':30}
d1[0]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[63], line 2
      1 d1 = {'Ramesh':20, 'Suresh':30}
----> 2 d1[0]

KeyError: 0
```

- Why error?
- because we are learing a pairing in dictionary
- above l and s are single thats why not error

in dict we can not use index same like str,list,tupel

- like l[0]
- str[0]
- tup[0]
- because key and value pair together

In [65]:
```
d1['Ramesh'] #use like this
```

Out[65]: 20

In [5]:
```
d1 = {'Ramesh':20, 'Suresh':30, 'Sathis':40}
d1['Ramesh'] #20
d1['Suresh'] #30
d1['Sathis'] #40

# What is common? => d1[]
# What is changing => i
```

Out[5]: 40

In [7]:
```
for i in range(len(d1)):
    print(i)
```

```
0
1
2
```

In [76]:
```
for i in d1:
    #print(i,d1[i])
    print(f'{i} has age : {d1[i]}')
```

```
Ramesh has age : 20
Suresh has age : 30
Sathis has age : 40
```

- whenever we iterate dictionary using for loop in operator it will give keys
- if want to get values we will access through keys only
- Key will work as index

Note

- using **range** in dictionary we can find only index
  - for i in range(len(d1)):
    - print(i)
- and using **in** operator in dictionary we get key and value
  - for i in d1:
    - print(f'{i} has age : {d1[i]}')

The below things are very very important

In [77]:
```
d1 = {'fruits':['Apple','Banana','Cherry']}
d1['fruits'][2]
```

Out[77]: 'Cherry'

In [78]:
```
d1 = {'fruits':{'Apple':['Sweet','Green','Custred']}}
len(d1)
```

Out[78]: 1

In [80]:
```
d1['fruits']['Apple']
```

Out[80]: ['Sweet', 'Green', 'Custred']

In [81]:
```
d1['fruits']['Apple'][1]
```

Out[81]: 'Green'

In [82]:
```
d1 = {'fruits':[{'cost':[100,200,300]}]}
d1
```

Out[82]:
```
{'fruits': [{'cost': [100, 200, 300]}]}
```

In [85]:
```
d1['fruits'][0]['cost'][2]
```

Out[85]:
```
300
```

In [88]:
```
d1 = {'fruits':['Apple'],'cost':[{'1kg':[300,500],'2kg':['none']}]}
d1
```

Out[88]:
```
{'fruits': ['Apple'], 'cost': [{'1kg': [300, 500], '2kg': ['none']}]}
```

In [93]:
```
d1['cost'][0]['2kg'][0]
```

Out[93]:
```
'none'
```

In [98]:
```
d1 = {'fruits':{'Mango':{'Nagpur':{"MH":{'King':{"Shivaji":{"Sambhaji"}}}}}}}
d1
```

Out[98]:
```
{'fruits': {'Mango': {'Nagpur': {'MH': {'King': {'Shivaji': {'Sambhaji'}}}}}}}
```

In [107…
```
d1['fruits']['Mango']['Nagpur']['MH']['King']['Shivaji'][0]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[107], line 1
----> 1 d1['fruits']['Mango']['Nagpur']['MH']['King']['Shivaji'][0]

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

In [ ]:
```
d1 = {
    'fruits': {
        'Mango': {
            'Nagpur': {
                'MH': {
                    'King': {
                        'Shivaji': {'item1', 'item2', 'item3'}
                    }
                }
            }
        }
    }
}
```

- Why error?
- if you notice carefully after shivaji we are passing {} but not passing key and value pair
- so without key value pair it is a set and you can't use an index to access its elements.

- Then, accessing d1['fruits']['Mango']['Nagpur']['MH']['King']['Shivaji'][0] will give you the error because 'Shivaji' is a set, and you can't use an index to access its elements.

In [108…
```
dir(dict)
```

Out[108…
```
['__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']
```

Methods Dict

1. clear
2. copy
3. fromkeys

4. get
5. items
6. keys
7. pop
8. popitem
9. setdefault
10. update
11. values

How to converts list to dictionary

In [110…
```
names = ['Ramesh', 'Suresh', 'Sathish']
age = [20,30,50]

# {}
dict1 = {}
for i,j in zip(names,age):
    print(i,j)
```

```
Ramesh 20
Suresh 30
Sathish 50
```

In [10]:
```
names = ['Ramesh', 'Suresh', 'Sathish']
age = [20,30,50]
dict1 = {}
dict1['Ramesh'] = 20 # ['Ramesh'] is key and 20 is value
dict1['Suresh'] = 30
dict1['Sathish'] = 50

dict[i] = j
# What is common? dict1[]
# What is changing i,j => dict[i] = j
```

In [12]:
```
names = ['Ramesh', 'Suresh', 'Sathish']
age = [20,30,50]
dict1 = {}

for i,j in zip(names,age):
    dict1[i]=j

dict1
```

Out[12]:
```
{'Ramesh': 20, 'Suresh': 30, 'Sathish': 50}
```

In [ ]:
```
# Very Important
# strings to convert list (split())
# list to convert strings (''.join())
# list to convert dict
# dict to convert list
```

How to converts dictionary to list

In [14]:
```
d = {'Ramesh': 20, 'Suresh': 30, 'Sathish': 50}
# iterate it print i ===== Key
```

```
# using key how to get value
# Take two empty list and append it
```

In [17]:
```
d = {'Ramesh': 20, 'Suresh': 30, 'Sathish': 50}
l1,l2 =[],[]
for key in d:
    l1.append(key)
    l2.append(d[key])

l1,l2
```

Out[17]: (['Ramesh', 'Suresh', 'Sathish'], [20, 30, 50])

In [ ]:
```
# How to converts list to dictionary

names = ['Ramesh', 'Suresh', 'Sathish']
age = [20,30,50]
dict1 = {}

for i,j in zip(names,age):
    dict1[i]=j

dict1
#############################################################################
# How to converts dictionary to list

l1,l2 =[],[]
for key in d:
    l1.append(key)
    l2.append(d[key])

l1,l2
```

In [18]:
```
d = {'Ramesh': 20, 'Suresh': 30, 'Sathish': 50}
l1,l2 =[],[]
for key in d:
    l1.append(key)
    l2.append(d[key])

print(l1)
l2
```

```
['Ramesh', 'Suresh', 'Sathish']
```

Out[18]: [20, 30, 50]

In [20]:
```
# Here we can apply list comprehension

l1=[key for key in d]
l2=[d[key] for key in d]

l1,l2
```

Out[20]: (['Ramesh', 'Suresh', 'Sathish'], [20, 30, 50])

Dictionary comprehension

In [ ]:
```
# syntax
```

```
{key:value for key,value in zip(<var1>,<var2>)}
```

In [26]:
```
names = ['Ramesh', 'Suresh', 'Sathish']
age = [20,30,50]

dict1= {i:j for i,j in zip(names,age)}
dict1
```

Out[26]:
```
{'Ramesh': 20, 'Suresh': 30, 'Sathish': 50}
```

comprehension

- in list we required square brackets []
  - output i
- in dictionary we required curly braces {}
  - output key:value means i:j

Mutable - Immutable

- list are mutable
- string are immutable
- tuple are immutable
- dictionary also mutable we can modifiy the value of a dictionary

In [9]:
```
d = {'Ramesh': 20, 'Suresh': 30}
d['Suresh'] = 50
d
```

Out[9]:
```
{'Ramesh': 20, 'Suresh': 50}
```

- dictionary is similar to database operations
- keys always unique but values can be change

concatenation

In [27]:
```
d1 = {'Ramesh': 20}
d2 = {'Fruits':'Apple'}
d1+d2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[27], line 3
      1 d1 = {'Ramesh': 20}
      2 d2 = {'Fruits':'Apple'}
----> 3 d1+d2

TypeError: unsupported operand type(s) for +: 'dict' and 'dict'
```

In dictionary concatenation will failed

- by method update we can do
- but using + operand can't do

items-values-keys

In [28]:
```
d = {'Ramesh':20, 'Suresh': 30, 'Sathis': 40}
d.items
```

Out[28]:
```
<function dict.items>
```

In [29]:
```
d = {'Ramesh':20, 'Suresh': 30, 'Sathis': 40}
d.items()
```

Out[29]:
```
dict_items([('Ramesh', 20), ('Suresh', 30), ('Sathis', 40)])
```

In [30]:
```
d = {'Ramesh':20, 'Suresh': 30, 'Sathis': 40}
d.values()
```

Out[30]:
```
dict_values([20, 30, 40])
```

In [31]:
```
d = {'Ramesh':20, 'Suresh': 30, 'Sathis': 40}
d.keys()
```

Out[31]:
```
dict_keys(['Ramesh', 'Suresh', 'Sathis'])
```

- The above output look like list but it is a dictionary
  - dict_keys
  - dict_items
  - dict_values

In [32]:
```
d = {'Ramesh':20, 'Suresh': 30, 'Sathis': 40}
key = d.keys()
type(key)
```

Out[32]:
```
dict_keys
```

In [33]:
```
len(key)
```

Out[33]:
```
3
```

In [34]:
```
key[0] # here error
       # because it not a list
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[34], line 1
----> 1 key[0]

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

- dict_keys is a type has keys represented in a list

- It is looks like list but not List
- We can not use list methods here
- For that we need to convert dict_keys to list

In [10]:
```
d = {'Ramesh':20, 'Suresh': 30, 'Sathis': 40}
key = d.keys() # dict_keys type
list(key)      # list      type
```

Out[10]:
```
['Ramesh', 'Suresh', 'Sathis']
```

In [12]:
```
d = {'Ramesh':20, 'Suresh': 30, 'Sathis': 40}
values = d.values() # dict_values type
list(values)        #  list type
```

Out[12]:
```
[20, 30, 40]
```

In [11]:
```
d = {'Ramesh':20, 'Suresh': 30, 'Sathis': 40}
items = d.items() # dict_items type
list(items)       # list type
```

Out[11]:
```
[('Ramesh', 20), ('Suresh', 30), ('Sathis', 40)]
```

In [ ]:
```
# Without methods

d = {'Ramesh': 20, 'Suresh': 30, 'Sathish': 50}
keys,values =[],[]
for key in d:
    keys.append(key)
    values.append(d[key])

keys, values
#####################################################################
# List comprehension

d = {'Ramesh': 20, 'Suresh': 30, 'Sathish': 50}
keys=[key for key in d]
values=[d[key] for key in d]
keys,values

#####################################################################

# With methods

d = {'Ramesh': 20, 'Suresh': 30, 'Sathish': 50}
keys = list(d.keys())
values = list(d.values())
keys,values
```

- Que:- When we are using keys list, values list, then what is the use of items
- items : for both are comming
- values : values

- keys : keys only

In [13]:
```
# Que:-

d = {'Ramesh': 20, 'Suresh': 30, 'Sathish': 50}
# d = {'ramesh': 20, 'suresh': 30, 'sathish': 50}
# ans like above
```

In [14]:
```
d = {'Ramesh': 20, 'Suresh': 30, 'Sathish': 50}
keys = [i.lower() for i in list(d.keys())]
values = [str(i) for i in list(d.values())]
{key:value for key,value in zip(key,values)}
```

Out[14]:
```
{'Ramesh': '20', 'Suresh': '30', 'Sathis': '50'}
```

get

In [19]:
```
d = {'Ramesh': '20', 'Suresh': '30', 'Sathis': '50'}
d.get('Ramesh')
```

Out[19]:
```
'20'
```

In [42]:
```
d = {'Ramesh': '20', 'Suresh': '30', 'Sathis': '50'}
d.get('Ramesh') # ans
d.get('Ram') # No error - no ans
```

In [44]:
```
d['Ramesh'] # Ans normal way
d['Ram'] # Error comes
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[44], line 2
      1 d['Ramesh'] # Ans normal way
----> 2 d['Ram'] # Error comes

KeyError: 'Ram'
```

update

In [48]:
```
d = {'Ramesh': '20', 'Suresh': '30', 'Sathis': '50'}
d1 = {'Fruits':'Apple'}
d.update(d1)
d
```

Out[48]:
```
{'Ramesh': '20', 'Suresh': '30', 'Sathis': '50', 'Fruits': 'Apple'}
```

In [50]:
```
d.update('Apple'=200)
```

```
  Cell In[50], line 1
    d.update('Apple'=200)
                    ^
SyntaxError: expression cannot contain assignment, perhaps you meant "=="?
```

In [51]:
```
d.update('Apple':200)
```

```
  Cell In[51], line 1
    d.update('Apple':200)
                     ^
SyntaxError: invalid syntax
```

In [53]:
```
d.update(Apple=200)
d
```

Out[53]:
```
{'Ramesh': '20',
 'Suresh': '30',
 'Sathis': '50',
 'Fruits': 'Apple',
 'Apple': 200}
```

In [56]:
```
t = (200,'marks')
l = [t]
d.update(l)
d
```

Out[56]:
```
{'Ramesh': '20',
 'Suresh': '30',
 'Sathis': '50',
 'Fruits': 'Apple',
 'Apple': 200,
 200: 'marks'}
```

In [57]:
```
d.update([(200, 'marks')])
```

In [58]:
```
d
```

Out[58]:
```
{'Ramesh': '20',
 'Suresh': '30',
 'Sathis': '50',
 'Fruits': 'Apple',
 'Apple': 200,
 200: 'marks'}
```

- concatenation of two dictionary is fails
- But we can combined two dictionary using update method

- 3 ways you can update
  - 1. one is directly
    - d.update(d1) # dict in dict
  - 2. key = value
    - d.update(Apple=200)
  - 3. list of tuples
    - d.update([(200, 'marks')])

In [ ]:
```
d = {'Ramesh': '20', 'Suresh': '30', 'Sathis': '50'}
d1 = {'Fruits':'Apple'}
d.update(d1)
d.update(Apple=200)
d.update([(200, 'marks')])
```

pop-popitems

In [59]:
```
d = {'Ramesh': '20', 'Suresh': '30', 'Sathis': '50'}
d.pop('Ramesh')
```

Out[59]: '20'

In [60]:
```
d
```

Out[60]: {'Suresh': '30', 'Sathis': '50'}

In [21]:
```
d = {'Ramesh': '20', 'Suresh': '30', 'Sathis': '50'}
d.pop('r')
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[21], line 2
      1 d = {'Ramesh': '20', 'Suresh': '30', 'Sathis': '50'}
----> 2 d.pop('r')

KeyError: 'r'
```

In [ ]:
```
l = [12,32,123,234,213,234]
l.pop(3) # here we need index

d = {'Ramesh': '20', 'Suresh': '30', 'Sathis': '50'}
d.pop('Ramesh') # here we need key
```

- Differnce between list pop and dict pop
  - in list pop required a number
    - l = [12,32,123,234,213,234]
    - l.pop(3) # here we need index
  - in dict pop required a key
    - d = {'Ramesh': '20', 'Suresh': '30', 'Sathis': '50'}
    - d.pop('Ramesh') # here we need key

popitems

In [62]:
```
d = {'Ramesh': '20', 'Suresh': '30', 'Sathis': '50'}
d.popitem()
```

Out[62]: ('Sathis', '50')

In [63]:
```
d
```

```
Out[63]: {'Ramesh': '20', 'Suresh': '30'}
```

- pop will excpect a key, corresponding value will be removed
  - pop will return the **value which is removing**
- pop item has no arguments, last key-value pair will be removed
  - pop item return last **key-value pair**

del

```
In [64]: l = [100,200,33,444]
         del l[0]
         l
```

```
Out[64]: [200, 33, 444]
```

```
In [66]: l = [100,200,33,444]
         del l # here l list is gone
         l     #  thats why error will coming
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[66], line 3
      1 l = [100,200,33,444]
      2 del l # here l list is gone
----> 3 l     #  thats why error will coming

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

```
In [67]: d = {'Ramesh': '20', 'Suresh': '30', 'Sathis': '50'}
         del d['Ramesh']
         d
```

```
Out[67]: {'Suresh': '30', 'Sathis': '50'}
```

```
In [69]: d = {'Ramesh': '20', 'Suresh': '30', 'Sathis': '50'}
         del d
         d
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[69], line 3
      1 d = {'Ramesh': '20', 'Suresh': '30', 'Sathis': '50'}
      2 del d
----> 3 d

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

difference of following

- count in strings also in list
- index in string also in list
- find in strings but not in list

- pop in list also in dictionary
- remove inl ist but in dictionary pop items
- concatenation in list ===== extend in list
- access value using key in dict === get method
- del for list and dictionary
- sorted and reversed for all data types
- strings to list
- list to strings
- list to dictionary
- dictionary to list

In [ ]: