list

In [1]:
```
# anything that can be written inside a sqaure bracket and elements are se
```

In [2]:
```
my_list = [1,2,3,4,5,6,7]
print(type(my_list))
print(my_list)
```

```
<class 'list'>
[1, 2, 3, 4, 5, 6, 7]
```

Creation of list

In [3]:
```
# 1. you already know the element

fruits = ['apple','mango','orange']
print(fruits)
```

```
['apple', 'mango', 'orange']
```

In [7]:
```
# 2. taking the user input with eval

data = eval(input('enter the list'))
print(type(data),data)
```

```
enter the list[1,2,3,4,5]
<class 'list'> [1, 2, 3, 4, 5]
```

In [9]:
```
# 3. type casting
# Note:- you can only convert iterable object to a list
```

In [ ]:
```
# rules to find out if object is iterable or not

1. len function should work on that
or
2. dir(object) should have a __iter__ method in it
```

In [10]:
```
a = 10
# print(len(a)) # non iterable

b = 10.2
print(dir(b)) # non iterable

c = 'afsan'
print(len(c)) # iterable
print(dir(c))
```

...

In [11]:
```
a = 10
b = 100.2
c = 'data'

list(c)
```

Out[11]:
```
['d', 'a', 't', 'a']
```

In [12]:
```
#4. split()

a = 'learnbay insititute'
a.split()
```

Out[12]:
```
['learnbay', 'insititute']
```

In [13]:
```
print(list(a))
```

```
['l', 'e', 'a', 'r', 'n', 'b', 'a', 'y', ' ', 'i', 'n', 's', 'i', 't', 'i',
't', 'u', 't', 'e']
```

In [14]:
```
# # properties of list

# 1. seq data type
# 2. +ve and -ve indexing will be supported
# 3. operations - indexing, slicing, concatination, repetition, memebershi
# 4. allows duplicate elememt
# 5. it can contain any element of any data type
# 6. it is mutable
```

In [15]:
```
# 1. seq data type
# 2. +ve and -ve indexing will be supported

a = [1,2,3,4,5]
a[2]
```

Out[15]:
```
3
```

In [16]:
```
a[-1]
```

Out[16]:
```
5
```

In [17]:
```
a[-5]
```

Out[17]:
```
1
```

In [18]:
```
# allows duplicate elememt

data = [1,1,1,1,1,2,3,4,5,6,7,7,7,7,7]
print(data)
```

```
[1, 1, 1, 1, 1, 2, 3, 4, 5, 6, 7, 7, 7, 7, 7]
```

In [19]:
```
# it can contain any element of any data type

data = [1,'string',11.2,None,True,[1,2,3],50+7j]
print(data)
```

```
[1, 'string', 11.2, None, True, [1, 2, 3], (50+7j)]
```

operations

In [20]:
```
# 1. concatination
"""
op - +
operants - both should be list
"""

a = [1,2,3,4]
b = [100,200,300,400]
print(a+b)
```

```
[1, 2, 3, 4, 100, 200, 300, 400]
```

In [21]:
```
#2. repetition
"""
op- *
operands - on should be a list and other should be a int
"""

a = [100,200,300]
b = 5
print(a*b)
```

```
[100, 200, 300, 100, 200, 300, 100, 200, 300, 100, 200, 300, 100, 200, 300]
```

In [22]:
```
# indexing

a = ['vikcy','viknesh','arpana','mousami','afsan']
print(a[-1])
print(a[3])
```

```
afsan
mousami
```

In [23]:
```
# slicing
a = ['vikcy','viknesh','arpana','mousami','afsan']
print(a[-4:-1:1])
```

```
['viknesh', 'arpana', 'mousami']
```

In [24]:
```
print(a[-1::-1])
```

```
['afsan', 'mousami', 'arpana', 'viknesh', 'vikcy']
```

In [25]:
```
print(a[1:100:-1])
```

```
[]
```

In [26]:
```
# memebership

# it tells of the element is present in a list or not
a = ['vikcy','viknesh','arpana','mousami','afsan']
print('Viknesh' in a)
```

```
False
```

In [ ]:
```
print('mousami' in a)
```

```
True
```

In [27]:
```
# indentity
"""
return - Bool
op - is and is not

"""

a = [1,2,3,4,5]
b = [1,2,3,4,5]

print(a is b)
```

```
False
```

In [28]:
```
# mutable

a = [100,200,300,400]
print(id(a))
a[0] = 'afsan'
print(id(a))
```

```
132155478155392
132155478155392
```

In [29]:
```
1 a
```

Out[29]:
```
['afsan', 200, 300, 400]
```

In [30]:
```
1 a = 'afsan'
2 a[0] = 'p'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-30-02d7a9bf77b6> in <cell line: 2>()
      1 a = 'afsan'
----> 2 a[0] = 'p'

TypeError: 'str' object does not support item assignment
```

In [33]:
```
1 # nested lists - list inside a list
2
3 # data = [1,2,3,[1,2,3]] --> nested
```

In [32]:
```
1 data = [1,2,3,[1,2,3]]
2 data[-1][0]
```

Out[32]:
```
1
```

In [34]:
```
1 a = ['vikcy','viknesh','arpana',['mousami','afsan']]
2 print(a[3][-1])
```

```
afsan
```

In [35]:
```
1 data = [[1,2,3,4] , 5,6,7,8,[8,[9,10,['a','b']]]]
2 data[-1][-1][-1][0]
```

Out[35]:
```
'a'
```

In [36]:
```
1 data[5][1][2][0]
```

Out[36]:
```
'a'
```

In [37]:
```
1 data = [[1,2,3,4] , [5,6,7,['afsan','learnbay',['lionel messi', 'ronaldo']
2 data
```

Out[37]:
```
[[1, 2, 3, 4],
 [5, 6, 7, ['afsan', 'learnbay', ['lionel messi', 'ronaldo']]],
 8,
 [8, [9, 10, ['a', 'b']]]]
```

In [38]:
```
1 # messi
2 data[1][-1][-1][0][-5::]
```

Out[38]:
```
'messi'
```

In [39]:
```
a = 'lionel messi'
a[-5::1]
```

Out[39]:
```
'messi'
```

In [40]:
```
a = 'messi'
a
```

Out[40]:
```
'messi'
```

In [41]:
```
print(a)
```

```
messi
```

In [42]:
```
# advance list
print(dir(list))
```

```
['__add__', '__class__', '__class_getitem__', '__contains__', '__delattr__',
'__delitem__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__get
attribute__', '__getitem__', '__gt__', '__hash__', '__iadd__', '__imul__', '_
_init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__
mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__re
versed__', '__rmul__', '__setattr__', '__setitem__', '__sizeof__', '__str__',
'__subclasshook__', 'append', 'clear', 'copy', 'count', 'extend', 'index', 'i
nsert', 'pop', 'remove', 'reverse', 'sort']
```

In [44]:
```
# # methods to inserrt element in the list

# 1. append
# 2. extend
# 3. insert
```

In [45]:
```
# append
"""
1. it adds the elememt at the last
2. we can add only one element at a time
3. we can add both iterables and non-iterables object
"""

a = [100,65,77,'learnay']
a.append('afsan')
print(a)
```

```
[100, 65, 77, 'learnay', 'afsan']
```

In [46]:
```
a
```

Out[46]:
```
[100, 65, 77, 'learnay', 'afsan']
```

In [ ]:
```
a = [100,65,77,'learnay']
a.append('afsan','khan')
print(a)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [73], in <cell line: 2>()
      1 a = [100,65,77,'learnay']
----> 2 a.append('afsan','khan')
      3 print(a)

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

In [47]:
```
a = [100,65,77,'learnay']
a.append([100,200,300])
print(a)
```

```
[100, 65, 77, 'learnay', [100, 200, 300]]
```

In [48]:
```
# extend
"""
it is similar to concatination but in extent we are hchaing the original o
getting the new object

- you can use only iterable objects
"""
```

Out[48]:
```
'\nit is similar to concatination but in extent we are hchaing the original o
bject but in concatination we are\ngetting the new object\n\n- you can use on
ly iterable objects\n'
```

In [49]:
```
a = [1,2,3,4]
b = [100,200,300]

a.extend(b)
a
```

Out[49]:
```
[1, 2, 3, 4, 100, 200, 300]
```

In [50]:
```
a = [1,2,3,4]
b = [100,200,300]

a+b
print(a)
```

```
[1, 2, 3, 4]
```

In [51]:
```
a = [1,2,3,4]
a.extend('abc')
a
```

Out[51]:
```
[1, 2, 3, 4, 'a', 'b', 'c']
```

In [52]:
```
# insert
"""
inserts the value at a index
- we can only insert 1 element a time

"""

a = [10,20,30,40,50]
b = 'afsan'
a.insert(2,b)
print(a)
```

```
[10, 20, 'afsan', 30, 40, 50]
```

In [54]:
```
# # removing the elements from list

# 1.pop
# 2.remove
```

In [55]:
```
# 1.pop
"""
removes the element by index
pop bt deafult removes the last index
it is the only methods that returns the value
"""

a = [100,200,90,78,44]
print(a.pop())
print(a)
```

```
44
[100, 200, 90, 78]
```

In [56]:
```
a = [100,200,90,78,44]
print(a.pop(1))
print(a)
```

```
200
[100, 90, 78, 44]
```

In [ ]:
```
#2. removes
"""
removes the element by providing the ekleme

"""

a = [100,200,90,78,44]
a.remove(600)
print(a)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [92], in <cell line: 8>()
      2 """
      3 removes the element by providing the ekleme
      4
      5 """
      7 a = [100,200,90,78,44]
----> 8 a.remove(600)
      9 print(a)

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

In [57]:
```
a = [100,200,90,200,200,78,44]
a.remove(200)
print(a)
```

```
[100, 90, 200, 200, 78, 44]
```

In [58]:
```
# count -

a = [1,2,3,1,1,2,3,2,1,1,2,3,4,5,4,3,2,1,2,3]
a.count(3)
```

Out[58]: 5

In [59]:
```
# index - returns the index of the elements
# if multiple elements are present then the index of the first occurance w

a = [1,2,3,1,1,2,3,2,1,1,2,3,4,5,4,3,2,1,2,3]
a.index(3,3)
```

Out[59]: 6

In [60]:
```
# reverse

a = [1,2,3,4,5]
a.reverse()
print(a)
```

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

In [61]:
```
# sort
# sort the list for you

a = [4,6,7,8,4,3,5,6,2,1]
a.sort()
print(a)
```

```
[1, 2, 3, 4, 4, 5, 6, 6, 7, 8]
```

In [62]:
```
a = [4,6,7,8,4,3,5,6,2,1]
a.sort(reverse=True)
print(a)
```

```
[8, 7, 6, 6, 5, 4, 4, 3, 2, 1]
```

In [63]:
```
# clear

a = [1,2,3,4]
a.clear()
print(a)
```

```
[]
```

In [64]:
```
a = ['afsan' , 'zen' , 'vikcy', 'viknesh']
a.sort()
```

In [65]:
```
# append

a = ['afsan' , 'zen' , 'vikcy', 'viknesh']

a.append('Nisha')
```

In [66]:
```
a.index('viknesh')
```

Out[66]: 3

In [67]:
```
a = [1,2,3,4,5]
b = [1,2,3,4,5]

print(a , id(a))
print(b , id(b))
```

```
[1, 2, 3, 4, 5] 132155205339008
[1, 2, 3, 4, 5] 132155482545536
```

In [68]:
```
a = [1,2,3,4,5]
b = a #alisaing

print(a , id(a))
print(b , id(b))

b[0] = 100

print(a , id(a))
print(b , id(b))
```

```
[1, 2, 3, 4, 5] 132155260399744
[1, 2, 3, 4, 5] 132155260399744
[100, 2, 3, 4, 5] 132155260399744
[100, 2, 3, 4, 5] 132155260399744
```

In [69]:
```
# shallow copy

a = [1,2,3,4,5]
b = a.copy()

print(a , id(a))
print(b , id(b))

b[0] = 100

print(a , id(a))
print(b , id(b))
```

```
[1, 2, 3, 4, 5] 132155260298496
[1, 2, 3, 4, 5] 132155260299200
[1, 2, 3, 4, 5] 132155260298496
[100, 2, 3, 4, 5] 132155260299200
```

In [70]:
```
a = [1,2,3,4,[5,10,122]]
b = a
c = a.copy()

b[0] = 100
c[0] = 100
```

In [71]:
```
# shallow - it will fail if we have a nestes list and we are changing the
```

In [72]:
```
# deep copy

from copy import deepcopy

a = [1,2,3,[1,2,3]]
b = deepcopy(a)

print(id(a))
print(id(b))

b[-1][0] = 200

print(a)
print(b)
```

```
132155260302912
132155260390080
[1, 2, 3, [1, 2, 3]]
[1, 2, 3, [200, 2, 3]]
```

In [73]:
```
print(id(a[-1]))
print(id(b[-1]))
```

```
132155478162880
132155205340032
```

In [74]:
```
# list - list comphrension
```

In [75]:
```
# -> never repeat the code
# -> write to write a code in mininum line possible
```

In [76]:
```
for i in 'afsan':
    print('hey')
```

```
hey
hey
hey
hey
hey
```

In [77]:
```
a = ['abc' , 'xyz' , 100]

for data in a:
    print('hi')
```

```
hi
hi
hi
```

In [78]:
```
for i in ['afsan','megana','sarthak','mohan']:
    print(i[::-1])
```

```
nasfa
anagem
kahtras
nahom
```

In [79]:
```
a = 'mohan'
a[-1:-len(a)-1:-1]
```

Out[79]: 'nahom'

In [80]:
```
for i in ['a','b','c']:
    print(i+'k')
```

```
ak
bk
ck
```

In [81]:
```
a = [1,2,3,4,[5,6,7,8],[1,2,3,4,[8,['mousumi','messi']]]]

a[5][4][1][0]
```

Out[81]: 'mousumi'

In [82]:
```
for i in ['afsan','megana','sarthak','mohan'][0]:
    print(i[::-1])
```

```
a
f
s
a
n
```

In [83]:
```
a = [1,2,3,4,5]

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

```
1
2
3
4
5
```

In [84]:
```
a = [1,2,3,4,5]
b = []
for i in a:
    b.append(i**2)
```

In [85]:
```
a = ['afsan' , 'sarthak', 'yoges', 'jhilam']

# construct a list where it will contain the reverse of all the element
b = []
for i in a:
    b.append(i[::-1])
print(b)
```

```
['nasfa', 'kahtras', 'segoy', 'malihj']
```

In [87]:
```
# list comprehension

# 1. if we want to create a new list from the existing list
# 2. you logic should only have 1 statment

# syntax
# [statment for var in iterable if cond]
```

In [88]:
```
a = [1,2,3,4,5]
b = []
for i in a:
    b.append(i**2)
```

In [89]:
```
a = [1,2,3,4,5]
b = [i**2 for i in a]
```

In [90]:
```
a = ['afsan' , 'sarthak', 'yoges', 'jhilam']

# construct a list where it will contain the reverse of all the element
b = [i[::-1] for i in a]
print(b)
```

```
['nasfa', 'kahtras', 'segoy', 'malihj']
```

In [91]:
```
a = [1,2,3,4,5,6,7,8,9]
[(i%2 ==0) for i in a ]
```

Out[91]: [False, True, False, True, False, True, False, True, False]

In [92]:
```
a = ['afsaan@gmail.com' , 'akash@email.com' , 'gulab@gmail.com' , 'krishna
# contruct a list which will only contain the username of all the emails
b = []
for i in a:
    b.append(i.split('@')[0])
```

In [93]:
```
[i.split('@')[0] for i in a]
```

Out[93]: ['afsaan', 'akash', 'gulab', 'krishna', 'pranali']

In [95]:
```
# tuple

# collection of elements which is written inside a round bracket(optional)
```

In [96]:
```
a = (1,2,3,4,5,6)
print(a , type(a))
```

```
(1, 2, 3, 4, 5, 6) <class 'tuple'>
```

In [97]:
```
a = 1,2,3,4,5
print(a, type(a))
```

```
(1, 2, 3, 4, 5) <class 'tuple'>
```

In [99]:
```
# 2. taking user input

tup = eval(input('enter the tuple'))
print(type(tup))
```

```
enter the tuple1,2,3
<class 'tuple'>
```

In [101]:
```
# # # properties of tuple

# 1. Sequence data type
# 2. +ve and -ve indexing
# 3. allows duplicate element
# 4. allows to store any data type -
# 5. basic operation - indexing, slicing , concatination, repition , ident
# 6. immutbale object
```

In [102]:
```
# 1. Sequence data type
# 2. +ve and -ve indexing

a = (1,2,3.5,'afsan',[1,2,3])
a[0]
a[-1]
```

Out[102]:
```
[1, 2, 3]
```

In [103]:
```
# 3. allows duplicate element

a = (1,2,3,4,1,1,1,1,1,1)
print(a)
```

```
(1, 2, 3, 4, 1, 1, 1, 1, 1, 1)
```

In [104]:
```
# 4. allows to store any data type -

a = (1,1.2,True,'afsan',(1,2,3))

print(a)
```

```
(1, 1.2, True, 'afsan', (1, 2, 3))
```

In [105]:
```
# indexing

a = (1,1.2,True,'afsan',(1,2,3))

print(a[2])
```

```
True
```

In [106]:
```
# concatination
"""
adding tuples
op - +
operands = both should be tuple

"""

a  = (1,2,3)
b = (True,False)

print(a+b)
```

```
(1, 2, 3, True, False)
```

In [107]:
```
# repetition
"""
op - *
operand - int and tuple

"""
a = (1,2,3)
b = 50

print(a*b)
```

```
(1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1,
2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3,
1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2,
3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1,
2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3,
1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3)
```

In [108]:
```
# identity operator
"""
op - is and is not
return - bool
"""
a = (1,2,3,4,5)
b = (1,2,3,4,5)

print(a is not b)
```

```
True
```

In [109]:
```
# membership op
"""
op - in and not in
return - bool
"""

a = (1,1.2,True,'afsan',(1,2,3))
print('1.2' in a)
```

```
False
```

In [110]:
```
a = (1,1.2,True,'afsan',(1,2,3))
print(1.2 in a)
```

```
True
```

In [111]:
```
a = (1,2,3)
b = (5,6,7)

print(id(a))
print(id(b))

print(a is b)
```

```
132155260215936
132154648800000
False
```

In [112]:
```
# packing - assign multiple data type to a single variable

a = 1,2,True,100.2
print(a)
```

```
(1, 2, True, 100.2)
```

In [113]:
```
a = (1,2,3) # 1,2,3
b = (1,2) # 1,2
c = (1) # 1
print(c , type(c))
```

```
1 <class 'int'>
```

In [114]:
```
# can tuple contain a single element

c = (1,)
print(c)
```

```
(1,)
```

In [115]:
```
a = [1]
tuple(a)
```

Out[115]:
```
(1,)
```

In [117]:
```
a = eval(input())
print(a)
```

```
a
[1]
```

In [118]:
```
a
```

Out[118]:
```
[1]
```

In [119]:
```
# tuple unpacking

a,b,c,d = 1,2,3,4

print(d)
```

```
4
```

In [120]:
```
1 dir(tuple)
```

Out[120]:
```
['__add__',
 '__class__',
 '__class_getitem__',
 '__contains__',
 '__delattr__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__getnewargs__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__mul__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__rmul__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'count',
 'index']
```

In [121]:
```
1 # 2. index
2
3 a = 10,20,30,40,50,60,10,20,10
4
5 print(a.index(20))
```

```
1
```

In [122]:
```
1 # format string
2
3 # wap to take 2 user input and add the two number and output should be
4
5 "addition of two number is : ans"
```

Out[122]:
```
'addition of two number is : ans'
```

In [123]:
```
a=int(input())
b=int(input())
print("addition of two numbers is :", a+b)
```

```
10
20
addition of two numbers is : 30
```

In [124]:
```
# 1 way
a=int(input())
b=int(input())
print("addition of",a,"and",b,"is:",a+b)
```

```
10
20
addition of 10 and 20 is: 30
```

In [125]:
```
# 2way
a=int(input())
b=int(input())
print(f"addition of {a} and {b} is {a+b}")
```

```
10
20
addition of 10 and 20 is 30
```

In [126]:
```
# 3way
a=int(input())
b=int(input())
print("addition of {} and {} is {}".format(a,b,a+b))
```

```
30
10
addition of 30 and 10 is 40
```

In [127]:
```
a = f"addition of {a} and {b} is {a+b}"
```

In [128]:
```
a = ['afsan' , 18 , 70000]
print(a[0])
print(a[1])
print(a[2])
```

```
afsan
18
70000
```

In [129]:
```
# problem 1
a = [18 , 70000, 'afsan']
print(a[0])
print(a[1])
print(a[2])
```

```
18
70000
afsan
```

In [130]:
```
# problem number 2
a = [18 , 18, 'afsan']
print(a[2])
```

```
afsan
```

In [131]:
```
# dictionary data type - is used to represent thet structure data

employee_data = {"age":18 , 'salary':18 , "name":'afsan'}
print(employee_data , type(employee_data))
```

```
{'age': 18, 'salary': 18, 'name': 'afsan'} <class 'dict'>
```

In [132]:
```
employee_data["name"]
employee_data["age"]
employee_data['salary']
```

Out[132]:
```
18
```

In [134]:
```
# # properties of dict

# 1. it is a collection of key value pair
# 2. a key,value is called as item
# 3. items are sep by commas
# 4. key and value are sep by :
# 5. key cannot be duplicated whereas value can be duplicated
# 6. key cannot be mutable
# 7. value can be of any datatype
# 8. all the items are enclosed inside a {}
# 9. dict is not a seq data type - indexing , slicing , concatination, rep
# 10. membership op is applicable but only on keys
# 11. identity is applicable
# # 12. mutable datatype
```

In [136]:
```
# how to create dict

# 1. we already know the leme

employee_data = {"age":18 , 'salary':18 , "name":'afsan'}
```

In [139]:
```
# 3 using dict function

"""
dict function takes seq of inner seq where inner seq will have a pair of v
will be the key and scond value will be the value
"""
a = [(1,100),(2,200),(3,300),(4,400)]
dict(a)
```

Out[139]:
```
{1: 100, 2: 200, 3: 300, 4: 400}
```

In [140]:
```
# 4 zip

# zip(iter1 , iter2)

a = ["name","age","salary"]
b = [18 , 18 , 24 , "afsan"]
```

In [141]:
```
# 5. key cannot be duplicated whereas value can be duplicated

# scenario 1 - key is duplicated
employee_data = {"name":'afsan',"age":"18","salary":20 ,"age":20}
print(employee_data)
```

```
{'name': 'afsan', 'age': 20, 'salary': 20}
```

In [142]:
```
# scenario 2 - value is duplicated
employee_data = {"name":'afsan',"age":"afsan","salary":"afsan"}
print(employee_data)
```

```
{'name': 'afsan', 'age': 'afsan', 'salary': 'afsan'}
```

In [144]:
```
# 6. key cannot be mutable

# mutable
# 1. list
# 2. dict
# 3. set


# not mutable datatype
employee_data = {1:'afsan',1.2:"18","salary":20 ,10+7j:20 , False:100}
print(employee_data)
```

```
{1: 'afsan', 1.2: '18', 'salary': 20, (10+7j): 20, False: 100}
```

In [145]:
```
# 7. value can be of any datatype
employee_data = {1:'afsan',1.2:18,"salary":[20] ,10+7j:(20) , False:100}
print(employee_data)
```

```
{1: 'afsan', 1.2: 18, 'salary': [20], (10+7j): 20, False: 100}
```

In [146]:
```
# 10. membership op is applicable but only on keys

employee_data = {1:'afsan',1.2:18,"salary":[20] ,10+7j:(20) , False:100}

print('1.2' in employee_data)
```

```
False
```

In [147]:
```
employee_data = {1:'afsan',1.2:18,"salary":[20] ,10+7j:(20) , False:100}

print((20) in employee_data)
```

```
False
```

In [148]:
```
employee_data = {1:'afsan',1.2:18,"salary":[20] ,10+7j:(20) , False:100}

print((1) in employee_data)
```

```
True
```

In [149]:
```
# 11. identity is applicable

employee_data = {1:'afsan',1.2:18,"salary":[20] ,10+7j:(20) , False:100}
employee_data_1 = {1:'afsan',1.2:18,"salary":[20] ,10+7j:(20) , False:100}

print(employee_data is employee_data_1)
```

```
False
```

In [150]:
```
# 12. mutable datatype - original object can be changed

employee_data = {1:'afsan',1.2:18,"salary":[20] ,10+7j:(20) , False:100}
employee_data[1] = 'rahul'
```

In [151]:
```
employee_data
```

Out[151]:
```
{1: 'rahul', 1.2: 18, 'salary': [20], (10+7j): 20, False: 100}
```

In [152]:
```
name = "rohit"

name.replace
```

Out[152]:
```
<function str.replace(old, new, count=-1, /)>
```

In [153]:
```
# methods on dict data type
print(dir(dict))
```

```
['__class__', '__class_getitem__', '__contains__', '__delattr__', '__delitem_
_', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute_
_', '__getitem__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__
ior__', '__iter__', '__le__', '__len__', '__lt__', '__ne__', '__new__', '__or
__', '__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__ror__', '_
_setattr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'cle
ar', 'copy', 'fromkeys', 'get', 'items', 'keys', 'pop', 'popitem', 'setdefaul
t', 'update', 'values']
```

In [156]:
```
# # methods to add and access the element from the dict

# 1. get
# 2. setdeafault
# 3. update
```

In [155]:
```
# get - if key is present it returns the value, else deafult value will be

employee_details = { 'emp1' : {'name':'Afsan' , 'age': 23 , 'salary':70000
                    {'name':'rahul' , 'age': 23 , 'salary':6000}}
```

In [157]:
```
# if key is present

employee_details.get('emp1')
```

Out[157]:
```
{'name': 'Afsan', 'age': 23, 'salary': 70000}
```

In [158]:
```
# if key is not present
print(employee_details.get('emp3'))
```

```
None
```

In [159]:
```
# we can give customised message also

print(employee_details.get('emp3' , 'key not present'))
```

```
key not present
```

In [160]:
```
# 2. setdeafault - insert the key with the value of the deafult if the key
# value of the key

employee_details = { 'emp1' : {'name':'Afsan' , 'age': 23 , 'salary':70000
                    {'name':'rahul' , 'age': 23 , 'salary':6000}}


# if key is present

employee_details.setdefault('emp1')
```

Out[160]:
```
{'name': 'Afsan', 'age': 23, 'salary': 70000}
```

In [161]:
```
# if key is not present

employee_details.setdefault('emp3')
```

In [162]:
```
# cusotmised message
employee_details = { 'emp1' : {'name':'Afsan' , 'age': 23 , 'salary':70000
                    {'name':'rahul' , 'age': 23 , 'salary':6000}}

employee_details.setdefault('emp3' , 'somethingelse')
```

Out[162]:
```
'somethingelse'
```

In [163]:
```
# 3. update
# - updaye method is simialr to extend in list

employee_details = { 'emp1' : {'name':'Afsan' , 'age': 23 , 'salary':70000
                    {'name':'rahul' , 'age': 23 , 'salary':6000}}

new_data = { 'emp3' : {'name':'vikas' , 'age': 23 , 'salary':70000} , 'emp
                    {'name':'anshul' , 'age': 23 , 'salary':6000}}


employee_details.update(new_data)
print(employee_details)
```

```
{'emp1': {'name': 'Afsan', 'age': 23, 'salary': 70000}, 'emp2': {'name': 'rah
ul', 'age': 23, 'salary': 6000}, 'emp3': {'name': 'vikas', 'age': 23, 'salar
y': 70000}, 'emp4': {'name': 'anshul', 'age': 23, 'salary': 6000}}
```

In [164]:
```
employee_details = {'name':'sweta' , 'gender':'female'}

age = {'age':24 , 'gender':'male'}

employee_details.update(age)
```

In [165]:
```
# # methods to delete the items from the dict

# 1. pop
# 2. popitem
```

```
  File "<ipython-input-165-3ad40a481c53>", line 3
    1. pop
       ^
SyntaxError: invalid syntax
```

In [166]:
```
# pop
"""
removes specified key and return the corresponding value, if key is not fo
otherwise key error will be raised

"""

# if key is presnet
employee_details = { 'emp1' : {'name':'Afsan' , 'age': 23 , 'salary':70000
                    {'name':'rahul' , 'age': 23 , 'salary':6000}}

employee_details.pop('emp1')

employee_details
```

Out[166]:
```
{'emp2': {'name': 'rahul', 'age': 23, 'salary': 6000}}
```

In [167]:
```
# if key is not presnet
employee_details = { 'emp1' : {'name':'Afsan' , 'age': 23 , 'salary':70000
                    {'name':'rahul' , 'age': 23 , 'salary':6000}}

employee_details.pop('emp3' , 'key is not presnet')
```

Out[167]:
```
'key is not presnet'
```

In [168]:
```
# pop item

# removes and return the value as tuple (key,value) from the last

employee_details = { 'emp1' : {'name':'Afsan' , 'age': 23 , 'salary':70000
                    {'name':'rahul' , 'age': 23 , 'salary':6000}}

employee_details.popitem()
```

Out[168]:
```
('emp2', {'name': 'rahul', 'age': 23, 'salary': 6000})
```

In [169]:
```
# methods to check all the keys in a dict

employee_details = { 'emp1' : {'name':'Afsan' , 'age': 23 , 'salary':70000
                    {'name':'rahul' , 'age': 23 , 'salary':6000}}


employee_details.keys()
```

Out[169]:
```
dict_keys(['emp1', 'emp2'])
```

In [170]:
```
# methods to check all the values in a dict

employee_details = { 'emp1' : {'name':'Afsan' , 'age': 23 , 'salary':70000
                    {'name':'rahul' , 'age': 23 , 'salary':6000}}


employee_details.values()
```

Out[170]:
```
dict_values([{'name': 'Afsan', 'age': 23, 'salary': 70000}, {'name': 'rahul',
'age': 23, 'salary': 6000}])
```

In [172]:
```
# methods to check all the items in a dict

employee_details = { 'emp1' : {'name':'Afsan' , 'age': 23 , 'salary':70000
                    {'name':'rahul' , 'age': 23 , 'salary':6000}}


employee_details.items()
```

Out[172]:
```
dict_items([('emp1', {'name': 'Afsan', 'age': 23, 'salary': 70000}), ('emp2',
{'name': 'rahul', 'age': 23, 'salary': 6000})])
```

In [173]:

```
# fromkeys

a = [i for i in range(0,100)]
data = dict.fromkeys(a , 'some data')
data
```

Out[173]:
```
{0: 'some data',
 1: 'some data',
 2: 'some data',
 3: 'some data',
 4: 'some data',
 5: 'some data',
 6: 'some data',
 7: 'some data',
 8: 'some data',
 9: 'some data',
 10: 'some data',
 11: 'some data',
 12: 'some data',
 13: 'some data',
 14: 'some data',
 15: 'some data',
 16: 'some data',
 17: 'some data',
 18: 'some data',
 19: 'some data',
 20: 'some data',
 21: 'some data',
 22: 'some data',
 23: 'some data',
 24: 'some data',
 25: 'some data',
 26: 'some data',
 27: 'some data',
 28: 'some data',
 29: 'some data',
 30: 'some data',
 31: 'some data',
 32: 'some data',
 33: 'some data',
 34: 'some data',
 35: 'some data',
 36: 'some data',
 37: 'some data',
 38: 'some data',
 39: 'some data',
 40: 'some data',
 41: 'some data',
 42: 'some data',
 43: 'some data',
 44: 'some data',
 45: 'some data',
 46: 'some data',
 47: 'some data',
 48: 'some data',
 49: 'some data',
 50: 'some data',
 51: 'some data',
 52: 'some data',
 53: 'some data',
 54: 'some data',
 55: 'some data',
 56: 'some data',
```

```
57: 'some data',
58: 'some data',
59: 'some data',
60: 'some data',
61: 'some data',
62: 'some data',
63: 'some data',
64: 'some data',
65: 'some data',
66: 'some data',
67: 'some data',
68: 'some data',
69: 'some data',
70: 'some data',
71: 'some data',
72: 'some data',
73: 'some data',
74: 'some data',
75: 'some data',
76: 'some data',
77: 'some data',
78: 'some data',
79: 'some data',
80: 'some data',
81: 'some data',
82: 'some data',
83: 'some data',
84: 'some data',
85: 'some data',
86: 'some data',
87: 'some data',
88: 'some data',
89: 'some data',
90: 'some data',
91: 'some data',
92: 'some data',
93: 'some data',
94: 'some data',
95: 'some data',
96: 'some data',
97: 'some data',
98: 'some data',
99: 'some data'}
```

In [174]:

```
# clear
employee_details = { 'emp1' : {'name':'Afsan' , 'age': 23 , 'salary':70000
                    {'name':'rahul' , 'age': 23 , 'salary':6000}}

employee_details.clear()
```

In [175]:
```
# nested dict

employee_data = {'emp1':{"name":"afsan" , "age":25, "gender":"male"},
                 'emp2':{"name":"rahul" , "age":32, "gender":"male"}}
```

In [176]:
```
# get the age of emp2

employee_data['emp2']['age']
```

Out[176]: 32

In [177]:
```
# get - gender of employee 1

employee_data["emp1"].get("gender")

employee_data.get("emp1")["gender"]

employee_data.get("emp1").get("gender")

# without get
employee_data['emp1']['gender']
```

Out[177]: 'male'

In [178]:
```
employee_data = {'emp0':{"name":"afsan" , "age":25},
                 'emp2':{"name":"rahul" , "age":32, "gender":"male"}}

print(employee_data.get("emp1",{}).get("gender"))
```

```
None
```

In [179]:
```
print(employee_data.get("emp1"))
```

```
None
```

In [180]:
```
employee_data = {'emp1':{"name":"afsan" , "age":25, "gender":"male"},
                 'emp2':{"name":"rahul" , "age":32, "gender":"male"}}

employee_data['emp1']['age']=28
```

In [181]:
```
# dict comp

data = [1,2,3,4,5,6,7]
dict_data = {}
for i in data:
    dict_data[i] = i**2
```

In [182]:
```
data = [1,2,3,4,5,6,7]
dict_data = {}
for i in data:
    dict_data[i] = i**2
```

In [183]:
```
# set
```

In [184]:
```
{i:i**2 for i in data if i%2==0}
```

Out[184]:
```
{2: 4, 4: 16, 6: 36}
```

In [185]:
```
data = {}
data[1] = 4
```

In [186]:
```
# set

# - collection of values
# - it a not a seq data type
# - indexing and slicing is noy suported
# - collection of a unique values
# - mutable
# - cannot conatine mutable data type
```

In [187]:
```
data = {(1,),"1",2,3}
data
```

Out[187]:
```
{(1,), '1', 2, 3}
```

In [188]:
```
print(dir(set))
```

```
['__and__', '__class__', '__class_getitem__', '__contains__', '__delattr__',
'__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__',
'__gt__', '__hash__', '__iand__', '__init__', '__init_subclass__', '__ior__',
'__isub__', '__iter__', '__ixor__', '__le__', '__len__', '__lt__', '__ne__',
'__new__', '__or__', '__rand__', '__reduce__', '__reduce_ex__', '__repr__',
'__ror__', '__rsub__', '__rxor__', '__setattr__', '__sizeof__', '__str__', '_
_sub__', '__subclasshook__', '__xor__', 'add', 'clear', 'copy', 'difference',
'difference_update', 'discard', 'intersection', 'intersection_update', 'isdis
joint', 'issubset', 'issuperset', 'pop', 'remove', 'symmetric_difference', 's
ymmetric_difference_update', 'union', 'update']
```

In [189]:
```
data = {1,2.0,True,100+7j,False,0,"afsan"}
```

In [190]:
```
# # range data type

# -it is used with for loop

# syntax = range(start, stop, step) - used to generate the seq if number i

# start = 0 --> inclusive
# stop = user --> exlusive
# step = 1
```

In [191]:
```
list(range(22,2))
```

Out[191]:
```
[]
```

In [192]:
```
list(range(22,2,-1))
```

Out[192]:
```
[22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3]
```

In [193]:

```
# write a even number from 0 to 100 using loop

for i in range(101):
    if i%2==0:
        print(i)
```

```
0
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
34
36
38
40
42
44
46
48
50
52
54
56
58
60
62
64
66
68
70
72
74
76
78
80
82
84
86
88
90
92
94
96
98
100
```

In [194]:
```
for i in range(0,101,2):
    print(i)
```

```
0
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
34
36
38
40
42
44
46
48
50
52
54
56
58
60
62
64
66
68
70
72
74
76
78
80
82
84
86
88
90
92
94
96
98
100
```

In [ ]:
```
1
```