6.1) Tuples

- Built-in immutable sequence.
- If no argument is given, the constructor returns an empty tuple.
- If iterable is specified the tuple is initialized from iterable's items.
- If the argument is a tuple, the return value is the same object

In [40]:

```
t=()
print(type(t))

#tuple
t1 = (1,2,45,45.45 , 4+88j , "asit" , True)

print(t1[5])

#reversing tuple
t1 = (1,2,45,45.45 , 4+88j , "asit" , True)
print(t1[::-1])


#slicing tuple
print(t1[2:5])

#counting
print(t1.count(4))
print(t1.count('asit'))
print(t1.index(4+88j))
#imp
print(t1.count(True)) #here no of True values inside tuple is 2 becasue system reads 1 as
print(t1.count(1))

# max and min of tuple
t2=(1, 2, 3, 3, 4)
print(max(t2))
print(min(t2))

#nested tuple
print((t1,t2))
# nested tuple manipulation
t3=(t1,t2)
print(t3[1][2])

#length of a tuple
print(len(t3))
print(len(t1))

#checking elements in a tuple
print("asit" in t3[0])
print("asit"in t3) #imp-slicing does not work in tuple

#list inside a tuple
t4 = ((1,2,3,4,5) , [1,3,5,6,7,8]) #possible
print(t4)
```

```
<class 'tuple'>
asit
(True, 'asit', (4+88j), 45.45, 45, 2, 1)
(45, 45.45, (4+88j))
0
1
4
2
2
4
1
((1, 2, 45, 45.45, (4+88j), 'asit', True), (1, 2, 3, 3, 4))
3
2
7
True
False
((1, 2, 3, 4, 5), [1, 3, 5, 6, 7, 8])
```

In [41]:

```
#deleating a tuple
#as tuple can not be manipulated there for use del function to delete it from memory
t5 = ((1, 2, 32, 4), (4, 5, 6, 7, 8))
print(t5)
del t5
print(t5)
```

```
((1, 2, 32, 4), (4, 5, 6, 7, 8))

---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_3132\2433328342.py in <module>
      4 print(t5)
      5 del t5
----> 6 print(t5)

NameError: name 't5' is not defined
```

Tuple are basically follows Immutability concepts where it is not going to allow to change any element at a perticular index

- they are use for holding password generally cuz they are immutable

In [42]:

```
#immutability of tuples
t1[0]=67
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_3132\2838540288.py in <module>
      1 #immutability of tuples
----> 2 t1[0]=67

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

In [43]:

```
#iterrating over tuple
#same as a list
```

6.2) Sets

set is a collection of unique elements.

- How set is different from list and tuples?

> set always try to store unique elements by removing all the duplicate ele ments

In [59]:

```
#proof- that only contains unique elements
s2 = {1,1,12,12,3,3,3,4,5,5,5,55,523,34,3,45,6,67}
print("1 ",s2)
print("2 ",type(s2))

#empty curly bracket is a dectionary
s={}
print("3 ",type(s))

# List to Set conversion
print("4 ",list(s2))
```

```
1  {1, 34, 3, 4, 5, 6, 67, 523, 12, 45, 55}
2  <class 'set'>
3  <class 'dict'>
4  [1, 34, 3, 4, 5, 6, 67, 523, 12, 45, 55]
```

Note:- Set do does not allow to store lists cuz lists are mutable and only hashable type elements can be stored in a set

In [62]:

```
s4 = {1,2,3,4,[1,2,3,4]}
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_3132\846577679.py in <module>
----> 1 s4 = {1,2,3,4,[1,2,3,4]}

TypeError: unhashable type: 'list'
```

Note:- Set do does allow to store tuples cuz lists are hashable and immutable.

In [64]:

```
s4 = {1,2,3,4,(1,2,3,4)}
print(s4)
```

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

sets are not subscriptable i.e slicing or indexing operation can't be done in a set

In [66]:

```
s7  = {"sudh" , "sudh" , 2,3,4,5}
s7[2]
#only subsscriptable only if converted to list or a tuple
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_3132\309790300.py in <module>
      1 s7  = {"sudh" , "sudh" , 2,3,4,5}
----> 2 s7[2]
      3 #only subsscriptable only if converted to list or a tuple

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

In [67]:

```
#iterability of Sets
```

In [68]:

```
for i in s7 :
    print(i)
```

```
2
3
4
5
sudh
```

In [93]:

```
#Built in functions

# .add()
s7  = {"sudh" , "sudh" , 2,3,4,5}
s7.add(7)
print(s7)

# .pop()
s7  = {"sudh" , "sudh" , 2,8,4,23}
#removes an arbitrary element
print(s7.pop())
print(s7.pop())
print(s7.pop())
print(s7)
```

```
{2, 3, 4, 5, 7, 'sudh'}
2
4
23
{8, 'sudh'}
```

In [95]:

```
#set difference
s8 = {1,2,3,4}
s9 = {1,2,3,5}
print(s8.difference(s9))
print(s9.difference(s8))
```

```
{4}
{5}
```

In [96]:

```
#creating an empty set
s8.clear()
s8
```

Out[96]:

```
set()
```

6.3) Dictionary

In [132]:

```
# empty dictionary
d={}
print("1 ", type(d))

#dictionary contains keys and values such as dict = {"key":"value"}
d1 = {"name" : "sudh" , "emiil_id" : "ss@gmail.com" , "number" :234324}
print("2 ",d1)

#Multiple Keys are not allowed
d2 = {"name" : 1 ,"id":45, "name" :2}
print("3 ",d2) #dictionary takes the highest index key and removes the rest
#************************************************************#



#************************************************************#
#key can be string,integer,float or bool but i can't be a special character
#integer key
d3 = {234234 : "abc"}
print("4 ",d3)

#floating pt key
d4 = {234.45 : "abc"}
print("5 ",d4)

#boolian key
d5 = {True : "abc"}
print("6 ",d5)

#special character key (not possible)
print("7 ","{@:24}- special char key is not possible")

#list as a key (not possible)
print("8 ","{[1,2,3]:24}- list as a key is not possible because it is unhashable")

#tupleas a key(possible)
d9 = {(1,2,3) : "abc"}
print("9 ",d9)

#set as a key (not possible)
print("10 ","{{1,2,3}:24}- set as a key is not possible")

#dictionary as a key (not possible)
print("11 ","-{{}:24} - Dictionary as a key is not possible")
#************************************************************#

#************************************************************#
#Values can be be a list,dictionary,string,integer,float
d16 = {"batch_name" :["data science masters" , "web dev" , "JDS"]
       , "start_date": (28,14,21),
       "mentor_name" : {"krish naik", "sudhanshu" , "hitesh",
                        "anurag" , "navin","hayder"}}
print()
print("12 ",d16)
#************************************************************#



#************************************************************#
print()
```

```
#dictionary Manipulation

#adding ney key value pair to the dictionary
d16 = {"batch_name" :["data science masters" , "web dev" , "JDS"]
       , "start_date": (28,14,21),
       "mentor_name" : {"krish naik", "sudhanshu" , "hitesh",
                        "anurag" , "navin","hayder"}}
d16["timing"] = (8 , 8 ,8)
print("13 ", d16)
print()

#to know keys inside the dictionatry
print("14 ",d16.keys())
```

```
1  <class 'dict'>
2  {'name': 'sudh', 'emiil_id': 'ss@gmail.com', 'number': 234324}
3  {'name': 2, 'id': 45}
4  {234234: 'abc'}
5  {234.45: 'abc'}
6  {True: 'abc'}
7  {@:24}- special char key is not possible
8  {[1,2,3]:24}- list as a key is not possible because it is unhashable
9  {(1, 2, 3): 'abc'}
10  {{1,2,3}:24}- set as a key is not possible
11  -{{}:24} - Dictionary as a key is not possible

12  {'batch_name': ['data science masters', 'web dev', 'JDS'], 'start_dat
e': (28, 14, 21), 'mentor_name': {'anurag', 'hitesh', 'hayder', 'krish nai
k', 'sudhanshu', 'navin'}}

13  {'batch_name': ['data science masters', 'web dev', 'JDS'], 'start_dat
e': (28, 14, 21), 'mentor_name': {'anurag', 'hitesh', 'hayder', 'krish nai
k', 'sudhanshu', 'navin'}, 'timing': (8, 8, 8)}

14  dict_keys(['batch_name', 'start_date', 'mentor_name', 'timing'])
```

slicing

In [136]:

```
d15 = {"key" : {"name" : "sudhanshu" , "class" : "DSM"}}
#extracting class
d15["key"]["class"]
```

Out[136]:

```
'DSM'
```

In [137]:

```
#adding and deleting a key
d15["key1"]=[15]
d15
```

Out[137]:

```
{'key': {'name': 'sudhanshu', 'class': 'DSM'}, 'key1': [15]}
```

In [138]:

```
del d15["key1"]
d15
```

Out[138]:

```
{'key': {'name': 'sudhanshu', 'class': 'DSM'}}
```

In [139]:

```
#length of dictionary- gives no of key value pairs
len(d16)
```

Out[139]:

```
4
```

In [145]:

```
#getting all values in a dict.
print("1 ",d16.values())
print()

#converting above into a list
print("2 ",list(d16.values()))
```

```
1  dict_values([['data science masters', 'web dev', 'JDS'], (28, 14, 21),
{'anurag', 'hitesh', 'hayder', 'krish naik', 'sudhanshu', 'navin'}, (8, 8,
8)])

2  [['data science masters', 'web dev', 'JDS'], (28, 14, 21), {'anurag',
'hitesh', 'hayder', 'krish naik', 'sudhanshu', 'navin'}, (8, 8, 8)]
```

In [146]:

```
#fetching list of key and values both
list(d16.items())
#here key and value pairs are inside tuples
```

Out[146]:

```
[('batch_name', ['data science masters', 'web dev', 'JDS']),
 ('start_date', (28, 14, 21)),
 ('mentor_name',
  {'anurag', 'hayder', 'hitesh', 'krish naik', 'navin', 'sudhanshu'}),
 ('timing', (8, 8, 8))]
```

In [148]:

```
#building replica of d16
d17=d16.copy()
print(d17)
```

```
{'batch_name': ['data science masters', 'web dev', 'JDS'], 'start_date':
(28, 14, 21), 'mentor_name': {'anurag', 'hitesh', 'hayder', 'krish naik',
'sudhanshu', 'navin'}, 'timing': (8, 8, 8)}
```

note in .copy method the value is stored in another address so any change in d17 does not affect d16 and visa versa. it is also called **deep copying**

note this method is different than d17=d16 cu is assigns same address to the value c/a **swollow copy**

In [152]:

```
d18 = d16
print(id(d16))
print(id(d18))
print(id(d17))
```

```
2086896165376
2086896165376
2086896166976
```

In [161]:

```
#removing value using POP()
#.pop() method also returns removed key-value
d19 = {"batch_name" :["data science masters" , "web dev" , "JDS"]
       , "start_date": (28,14,21),
       "mentor_name" : {"krish naik", "sudhanshu" , "hitesh",
                        "anurag" , "navin","hayder"},"timing":(8,8,8)}
a=d19.pop("timing")
print(d19)
print()
print(a)
```

```
{'batch_name': ['data science masters', 'web dev', 'JDS'], 'start_date':
(28, 14, 21), 'mentor_name': {'anurag', 'hitesh', 'hayder', 'krish naik',
'sudhanshu', 'navin'}}

(8, 8, 8)
```

In [162]:

```
#.fromkeys
d.fromkeys((1,2,3) , ('a','b','c'))
```

Out[162]:

```
{1: ('a', 'b', 'c'), 2: ('a', 'b', 'c'), 3: ('a', 'b', 'c')}
```

In [164]:

```
#dictionary inside a tuple
d19 = {"key1" : "value" , "key2" : "value2"}
d20 = {"key3" : "value3" , "key4" :"value4"}
(d19,d20)
```

Out[164]:

```
({'key1': 'value', 'key2': 'value2'}, {'key3': 'value3', 'key4': 'value
4'})
```

In [167]:

```
#.update()- use it to update any unique key-value pair from another dictionary
d19 = {"key1" : "value" , "key2" : "value2"}
d20 = {"key3" : "value3" , "key4" :"value4", "key2" : "value3"}
d19.update(d20)
d19
#if same key is available in d20 they d19 will be updated with the new value
```

Out[167]:

```
{'key1': 'value', 'key2': 'value3', 'key3': 'value3', 'key4': 'value4'}
```

In [170]:

```
#.get()- function

d20.get("asit")
```

In [171]:

```
d20.get("key3")
```

Out[171]:

```
'value3'
```

```
.get() has con as it does not give an keyerror
```

In [173]:

```
d20["asit"]
```

```
---------------------------------------------------------------------------
-
KeyError                                  Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_3132\4289294648.py in <module>
----> 1 d20["asit"]

KeyError: 'asit'
```

Dictionary Comprehenion

In [174]:

```
{i : i**2 for i in range(1,11)}
```

Out[174]:

```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100}
```

In [190]:

```
#Q- Create a dictionary containing log values b/w 1 and 10
import math as m
d22= {i :m.log10(i) for i in range(1,11)}
print(d22)
```

```
{1: 0.0, 2: 0.3010299956639812, 3: 0.47712125471966244, 4: 0.6020599913279
624, 5: 0.6989700043360189, 6: 0.7781512503836436, 7: 0.8450980400142568,
8: 0.9030899869919435, 9: 0.9542425094393249, 10: 1.0}
```

In [191]:

```
d22.keys()
```

Out[191]:

```
dict_keys([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
```

In [192]:

```
d22
```

Out[192]:

```
{1: 0.0,
 2: 0.3010299956639812,
 3: 0.47712125471966244,
 4: 0.6020599913279624,
 5: 0.6989700043360189,
 6: 0.7781512503836436,
 7: 0.8450980400142568,
 8: 0.9030899869919435,
 9: 0.9542425094393249,
 10: 1.0}
```

In [195]:

```
for i in d22.keys():
    if i%2==0:
        print(d22[i])
```

```
0.3010299956639812
0.6020599913279624
0.7781512503836436
0.9030899869919435
1.0
```

In [ ]: