Tuples

1. Tuple is similar to List except that the objects in tuple are immutable which means we cannot change the elements of a tuple once assigned.
2. When we do not want to change the data over time, tuple is a preferred data type.
3. Iterating over the elements of a tuple is faster compared to iterating over a list

Tuple Creation

```
tup1 = () # Empty tuple

tup2 = (50,60,70) #Tuple of Integers numbers

tup3 = (2.5,18.95,45.23) #Tuple of float numbers

tup4 = ('kitkat','melody',"eclairs") #Tuple of Strings

tup5 = ("Rishabh",50,(80,120),(150,90)) #Nested tuples

tup6 = (70,'Rishabh',23.26) #Tuple of mixed data types

tup7 = ('Rishabh',15,[60,80],[200,300],[400,600],{'Rishabh',
'Rajbhar'} ,(12,13,14))

len(tup7) #Length of list

7
```

Tuple Indexing

```
tup2[0] #Retrieve first element of the tuple

50

tup4[0] #Retreive first element of the tuple

'kitkat'

tup4[0][0] #Nested indexing - Access the first character of the first
tuple

'k'

tup4[-1] #Last item of the tuple

'eclairs'

tup5[-1] #Last item of the tuple

(150, 90)
```

Tuple Slicing

```
mytuple = ('one', 'two','three','four','five','six','seven','eight')

mytuple[0:3] #Return all items from 0th to 3rd index location
excluding the item

('one', 'two', 'three')

mytuple[2:5] #List all items from 2nd to 5th index location excluding
the items

('three', 'four', 'five')

mytuple[:3] #Return first three itmes

('one', 'two', 'three')

mytuple[:2] #Return last three items

('one', 'two')

mytuple[-3:] #Return last three items

('six', 'seven', 'eight')

mytuple[-2:] #Return last two items

('seven', 'eight')

mytuple[-1] #Return last item of the tuple

'eight'

mytuple[:] #Return whole tuple

('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')
```

Remove & Change Items

```
mytuple

----------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
last)
Cell In[67], line 1
----> 1 mytuple

NameError: name 'mytuple' is not defined

del mytuple[0] #Tuples are immutable which means we can't DELETE tuple
items
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[56], line 1
----> 1 del mytuple[0]

TypeError: 'tuple' object doesn't support item deletion
```

```
mytuple[0] = 1#Tuples are immutable which means we cant change tuple items
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[58], line 1
----> 1 mytuple[0] = 1

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

```
del mytuple # Deleting entire tuple object is possible
```

Loop through a tuple

```
mytuple
```

```
('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')
```

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

```
one
two
three
four
five
six
seven
eight
```

```
for i in enumerate(mytuple):
    print(i)
```

```
(0, 'one')
(1, 'two')
(2, 'three')
(3, 'four')
(4, 'five')
(5, 'six')
```

```
(6, 'seven')
(7, 'eight')
```

Count

```
mytuple1 = ('one','two','three','four','one','one','two','three')

mytuple1.count('one') #Number of times item "one" occured in the
tuple.

3

mytuple1.count('two') #Occurence of item 'two' in the tuple

2

mytuple1.count('four') #occurence of item 'four' in the tuple

1
```

Tuple Membership

```
mytuple

('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')

'one' in mytuple #Check if 'one' exist in the list

True

'ten' in mytuple #Check if 'ten' exist in the list

False

if 'three' in mytuple: #Check if 'three' exist in the list
    print('Three is present in the tuple')
else:
    print('Three is not present in the tuple')

Three is present in the tuple

if 'eleven' in mytuple: #check if "Eleven" exist in the list
    print('eleven is present in the tuple')
else:
    print('eleven is not present in the tuple')

eleven is not present in the tuple
```

Index Positioning

```
mytuple
```

```
('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')

mytuple.index('one') #Index of first element equal to 'one'

0

mytuple.index('five') #Index of first element equal to 'five'

4

mytuple1

('one', 'two', 'three', 'four', 'one', 'one', 'two', 'three')

mytuple.index('one') #Index of first element equal to 'one'

0
```

Sorting

```
mytuple2 = (10,15,12,8,9,20,25,50,75)

sorted(mytuple2) #Returns a new sorted list and doesn't change
original tuple

[8, 9, 10, 12, 15, 20, 25, 50, 75]

sorted(mytuple2, reverse=True) #Sort in descending order

[75, 50, 25, 20, 15, 12, 10, 9, 8]
```

Sets

1) Unordered & Unindexed collection of items. 2) Set elements are unique. Duplicate elements are not allowed. 3) Set elements are immutable (cannot be changed). 4) Set itself is mutable. We can add or remove items from it.

Set Creation

```
myset = {'a','b','c','d','e','f','g'} #Set of alphabets
myset

{'a', 'b', 'c', 'd', 'e', 'f', 'g'}

len(myset) #length of set

7

my_set = {'a','a','a','b','b','c','d','e','e','f','g'} #Duplicates
elements are not allowed
my_set
```

```
{'a', 'b', 'c', 'd', 'e', 'f', 'g'}
```

```
myset1 = {5.2,3.5,6.6,8.9,3.56,4.2} #Set of float numbers
myset1
```

```
{3.5, 3.56, 4.2, 5.2, 6.6, 8.9}
```

```
myset2 = {'Rahul','Narendra','Arvind','Shashi'} #Set of strings
myset2
```

```
{'Arvind', 'Narendra', 'Rahul', 'Shashi'}
```

```
myset3 = {4.5,'Rishabh',(20,30,40)} #Mixed Datatypes
myset3
```

```
{(20, 30, 40), 4.5, 'Rishabh'}
```

```
myset3 = {4.5,'Krishna',[20,30,40]} #Set does not allow mutable items
like list
myset3
```

```
----------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
Cell In[143], line 1
----> 1 myset3 = {4.5,'Krishna',[20,30,40]} #Set does not allow
mutable items like list
      2 myset3

TypeError: unhashable type: 'list'
```

```
myset4 = set() #Create an empty set
print(type(myset4))
```

```
<class 'set'>
```

```
my_set1 = set(('one','two','three','four'))
my_set1
```

```
{'four', 'one', 'three', 'two'}
```

Loop through a Set

```
myset = {'one','two','three','four','five','six','seven','eight'}
for i in myset:
    print(i)
```

```
seven
five
two
eight
```

```
six
four
three
one
```

```
for i in enumerate(myset):
    print(i)
```

```
(0, 'seven')
(1, 'five')
(2, 'two')
(3, 'eight')
(4, 'six')
(5, 'four')
(6, 'three')
(7, 'one')
```

SET MEMBERSHIP

```
myset
```

```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

```
'one' in myset #Check if 'one' exist in the set
```

```
True
```

```
'ten' in myset #Check if 'ten' exist in the set
```

```
False
```

```
if 'three' in myset: #Check if 'three' exist in the set
    print('Three is present in the set')
else:
    print('Three is not present in the set')
```

```
Three is present in the set
```

```
if 'eleven' in myset: #check if 'eleven' exist in the list
    print('eleven is present in the set')
else:
    print('eleven is not present in the set')
```

```
eleven is not present in the set
```

Add & Remove Items

```
myset
```

```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

```
myset.add('NINE') #Add item to a set using add() method
```

```
myset
```

```
{'NINE', 'eight', 'five', 'four', 'one', 'seven', 'six', 'three',
'two'}
```

```
myset.update(['TEN','ELEVEN','TWELVE']) #Add multiple itemusing
myset
```

```
{'ELEVEN',
 'NINE',
 'TEN',
 'TWELVE',
 'eight',
 'five',
 'four',
 'one',
 'seven',
 'six',
 'three',
 'two'}
```

```
myset.remove('NINE') #remove item in a set using remove() method
myset
```

```
{'ELEVEN',
 'TEN',
 'TWELVE',
 'eight',
 'five',
 'four',
 'one',
 'seven',
 'six',
 'three',
 'two'}
```

```
myset.discard('TEN') #remove item from a set using discard() method

myset
```

```
{'ELEVEN',
 'TWELVE',
 'eight',
 'five',
 'four',
 'one',
 'seven',
 'six',
 'three',
 'two'}
```

```
myset.clear() #Delete all items in a set
myset

set()

del myset #Delete the set object
myset

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[195], line 2
      1 del myset #Delete the set object
----> 2 myset

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

Copy Set

```
myset =  {'one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight'}
myset

{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}

myset1 = myset #creating a new reference 'myset1'
myset1

{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}

id(myset) , id(myset1) # The address of both myset & myset1 will be the same as

(2172788543488, 2172788543488)

my_set = myset.copy() #Create a copy of the list
my_set

{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}

id(my_set) #The address of my_set will be different

2172788546624

myset.add('nine')

myset

{'eight', 'five', 'four', 'nine', 'one', 'seven', 'six', 'three', 'two'}
```

```
myset1 # myset will also be impacted as at it is pointing to the same
set
```

```
{'eight', 'five', 'four', 'nine', 'one', 'seven', 'six', 'three',
'two'}
```

```
my_set #copy of the set wont be impacted due to changes made on the
original
```

```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

Set Operation

#Union

```
A = {'violet','indigo','blue'}
B = {'green','yellow','orange'}
C = {'red'}
```

```
A | B #Union of A and B (All elements from both sets. No duplicates
```

```
{'blue', 'green', 'indigo', 'orange', 'violet', 'yellow'}
```

```
A.union(B) #union of A and B
```

```
{'blue', 'green', 'indigo', 'orange', 'violet', 'yellow'}
```

```
A.union(B, C)
```

```
{'blue', 'green', 'indigo', 'orange', 'red', 'violet', 'yellow'}
```

```
"""
Updates the set calling the update() method with union of A , B & C.
For below example Set A will be updated with union of A,B & C.
"""
A.update(B,C)
A
```

```
{'blue', 'green', 'indigo', 'orange', 'red', 'violet', 'yellow'}
```

Intersection

```
A = {'violet','indigo','blue','green','yellow','red'}
B = {'orange','red','blue'}
```

```
A & B #Intersection of A and B (common items in both sets)
```

```
{'blue', 'red'}
```

```
A.intersection(B) Intersection of A and B
```

```
  Cell In[268], line 1
    A.intersection(B) Intersection of A and B
                      ^
SyntaxError: invalid syntax
```

```
"""
Updates the set calling the intersection_update() method with the
intersection o
For below example Set A will be updated with the intersection of A &
B.
"""
A.intersection_update(B)
A

{'blue', 'red'}
```

#Difference

```
a = {1,2,3,4,5}
b = {4,5,6,7,8}

a - b #set of elements that are only in A but not in B

{1, 2, 3}

a.difference(b) #Difference of sets

{1, 2, 3}

b - a #Set of elements that are only in b but not in a

{6, 7, 8}

"""
Updates the set calling the difference_update() method with the
difference of se
For below example Set B will be updated with the difference of B & A.
"""
b.difference_update(a)
b

{6, 7, 8}
```

#Symmetric Difference

```
A = {1,2,3,4,5}
B = {4,5,6,7,8}

A ^ B #Symmetric difference (Set of elements in A and B but not in
both)
```

```
{1, 2, 3, 6, 7, 8}

A.symmetric_difference(B) #Symmetric difference of sets

{1, 2, 3, 6, 7, 8}

"""
Updates the set calling the symmetric_difference_update() method with
the symmet
For below example Set A will be updated with the symmetric difference
of A & B.
"""
A.symmetric_difference_update(B)
A

{1, 2, 3, 6, 7, 8}
```

Subset , Superset & Disjoint

```
A = {1,2,3,4,5,6,7,8,9}
B = {3,4,5,6,7,8}
C = {10,20,30,40}

B.issubset(A)

True

A.issuperset(B)

True

C.isdisjoint(A)

True

B.isdisjoint(A)

False

# Other Builtin functions

A

{1, 2, 3, 4, 5, 6, 7, 8, 9}

sum(A)

45

max(A)

9
```

```
min(A)
```

```
1
```

```
len(A)
```

```
9
```

```
list(enumerate(A))
```

```
[(0, 1), (1, 2), (2, 3), (3, 4), (4, 5), (5, 6), (6, 7), (7, 8), (8,
9)]
```

```
D = sorted(A,reverse=True)
D
```

```
[9, 8, 7, 6, 5, 4, 3, 2, 1]
```

```
sorted(D)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Dictionary

Dictionary is a mutable data type in Python. A python dictionary is a collection of key and value pairs separated by a colon (:) & enclosed in curly braces {}. Keys must be unique in a dictionary, duplicate values are allowed.

```
# Create Dictionary
```

```
mydict = dict()
mydict          #empty dictionary
```

```
{}
```

```
mydict = {}
mydict
```

```
{}
```

```
mydict = {1:'ek', 2:'do',3:'teen'}  # dictionary with integer keys
```

```
mydict
```

```
{1: 'ek', 2: 'do', 3: 'teen'}
```

```
mydict = dict({1:'ek',2:'do',3:'teen'}) #create dicitonary using
dict()
```

```
mydict
```

```
{1: 'ek', 2: 'do', 3: 'teen'}
```

```
mydict = {'A':'one','B':'two','C':'three'} #dictionary with character
keys
mydict
```

```
{'A': 'one', 'B': 'two', 'C': 'three'}
```

```
mydict = {1:'one','A':'two' , 3:'three'} #dictionary with mixed keys
mydict
```

```
{1: 'one', 'A': 'two', 3: 'three'}
```

```
mydict.keys() # Return Dictionary Keys using keys() method
```

```
dict_keys([1, 'A', 3])
```

```
mydict.values() # Return Dictionary Values using values() method
```

```
dict_values(['one', 'two', 'three'])
```

```
mydict.items() # Access each key-value pair within a dictionary
```

```
dict_items([(1, 'one'), ('A', 'two'), (3, 'three')])
```

```
mydict = {1:'one' , 2:'two' , 'A':['Jenifer','Katty','Maria']}
mydict
```

```
{1: 'one', 2: 'two', 'A': ['Jenifer', 'Katty', 'Maria']}
```

```
mydict = {1:'one' , 2:'two' , 'A':['Jenifer','Katty','Maria'], 'B':
('Bat','Cat','Hat')}
mydict
```

```
{1: 'one',
 2: 'two',
 'A': ['Jenifer', 'Katty', 'Maria'],
 'B': ('Bat', 'Cat', 'Hat')}
```

```
keys = {'a' , 'b' , 'c' , 'd'}
mydict3 = dict.fromkeys(keys) # Create a dictionary from a sequence of
keys
mydict3
```

```
{'a': None, 'd': None, 'c': None, 'b': None}
```

```
keys = {'a' , 'b' , 'c' , 'd'}
value = 10
mydict3 = dict.fromkeys(keys , value) # Create a dictionary from a
sequence of
mydict3
```

```
{'a': 10, 'd': 10, 'c': 10, 'b': 10}
```

```
keys = {'a' , 'b' , 'c' , 'd'}
value = [10,20,30]
mydict3 = dict.fromkeys(keys , value) # Create a dictionary from a 
sequence of
mydict3
```

```
{'a': [10, 20, 30], 'd': [10, 20, 30], 'c': [10, 20, 30], 'b': [10, 
20, 30]}
```

```
value.append(40)
mydict3
```

```
{'a': [10, 20, 30, 40],
 'd': [10, 20, 30, 40],
 'c': [10, 20, 30, 40],
 'b': [10, 20, 30, 40]}
```

```
# Accessing items
```

```
mydict = {1:'one',2:'two',3:'three',4:'four'}
mydict
```

```
{1: 'one', 2: 'two', 3: 'three', 4: 'four'}
```

```
mydict[1] #access item using key
```

```
'one'
```

```
mydict.get(1) #Access item using get() method
```

```
'one'
```

```
mydict = {'Name':'Asif','ID':74123, 'DOB':1990,'Job':'Analyst'}
mydict
```

```
{'Name': 'Asif', 'ID': 74123, 'DOB': 1990, 'Job': 'Analyst'}
```

```
mydict['Name'] #acces item using key
```

```
'Asif'
```

```
mydict.get('Job') #access item using get() method
```

```
'Analyst'
```

Loop through a Dicitonary

```
mydict1 = 
{'Name':'Rishabh','Id':7898,'DOB':1998,'Address':'Varanasi','Job':'Ana
lyst'}
mydict1
```

```
{'Name': 'Rishabh',
 'Id': 7898,
 'DOB': 1998,
 'Address': 'Varanasi',
 'Job': 'Analyst'}
```

```
for i in mydict1:
    print(mydict1[i]) #Dictionary items
```

```
Rishabh
7898
1998
Varanasi
Analyst
```

Dictionary Membership

```
mydict1 =
{'Name':'Rishabh','Id':7898,'DOB':1998,'Address':'Varanasi','Job':'Ana
lyst'}
mydict1
```

```
{'Name': 'Rishabh',
 'Id': 7898,
 'DOB': 1998,
 'Address': 'Varanasi',
 'Job': 'Analyst'}
```

```
'Name' in mydict1 #test if a key is in a dictionary or not.
```

```
True
```

```
'Rishabh' in mydict1 #Membership test can only be done for keys
```

```
False
```

```
'Id' in mydict1
```

```
True
```

```
'Address' in mydict1
```

```
True
```

```
'contact' in mydict1
```

```
False
```

All / Any

The all() method returns:

.True - If all all keys of the dictionary are true

.False - If any key of the dictionary is false

.The any() function returns True if any key of the dictionary is True. If not, any() returns False.

```
mydict1 =
{'Name':'Rishabh','Id':7898,'DOB':1998,'Address':'Varanasi','Job':'Ana
lyst'}
mydict1
```

```
{'Name': 'Rishabh',
 'Id': 7898,
 'DOB': 1998,
 'Address': 'Varanasi',
 'Job': 'Analyst'}
```

```
all(mydict1) #will return false as one value is false (value 0)
```

```
True
```