tuple creation

```
In [4]: tup1= () #empty tuple
```

```
In [5]: tup2= (2,5,18,20) #tuple of integer numbers
```

```
In [6]: tup3= (3.4,5.2,10.4,6.7) #tuple of float numbers
```

```
In [7]: tup4= ('hi','welcome to','python') #tuple of strings
```

```
In [8]: tup5= ('python',10,(20,30),(50,40)) # nested tuples
```

```
In [9]: tup6= (10,6.4,'hello') #tuple of mixed data types
```

```
In [10]: tup7= (20,'hi',[50,40],{'hi','python'},(24,60))
```

```
In [11]: len(tup7)
```

```
Out[11]: 5
```

Tuple Indexing

```
In [12]: tup[2] #retreive third element of the tuple
```

```
Out[12]: 18
```

```
In [13]: tup4[1] #retreive second element of the tuple
```

```
Out[13]: 'welcome to'
```

```
In [14]: tup7[-1]  # Last item of the tuple
```

```
Out[14]: (24, 60)
```

Tuple Slicing

```
In [15]: mytuple = ('one','two','three','four','five','six','seven','eight','nine')
```

```
In [16]: mytuple[0:4] #return all element from 0 to 4 index
```

```
Out[16]: ('one', 'two', 'three', 'four')
```

```
In [17]: mytuple[:2] #return first two items
```

```
Out[17]: ('one', 'two')
```

```
In [18]: mytuple[-3:] #return last three items
```

```
Out[18]: ('seven', 'eight', 'nine')
```

```
In [19]: mytuple[-1] #return last item of the tuple
```

```
Out[19]: 'nine'
```

```
In [20]: mytuple[:] #return whole tuple
```

```
Out[20]: ('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight', 'nine')
```

Remove and chnge items

```
In [21]: mytuple
```

```
Out[21]: ('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight', 'nine')
```

```
In [22]: del mytuple[0] # Tuples are immutable which means we can't DELETE tuple items
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[22], line 1
----> 1 del mytuple[0] # Tuples are immutable which means we can't DELETE tuple i
tems

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

```
In [23]: mytuple[0] = 1 # Tuples are immutable which means we can't CHANGE tuple items
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[23], line 1
----> 1 mytuple[0] = 1 # Tuples are immutable which means we can't CHANGE tuple i
tems

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

```
In [24]: del mytuple # Deleting entire tuple object is possible
```

Loop through a tuple

```
In [2]: mytuple =('one' , 'two' , 'three' , 'four' , 'five' , 'six' , 'seven' , 'eight')
```

```
In [3]: for i in mytuple:
            print(i)
```

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

```
In [4]: for i in enumerate(mytuple):
            print(i)
```

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

count

In [9]:
```
mytuple = ('one', 'two', 'two','four','three', 'four', 'five', 'six', 'seven', '
```

In [10]:
```
mytuple.count('four') # Number of times item "four" occurred in the tuple.
```

Out[10]:
```
2
```

In [11]:
```
mytuple.count('one') # Number of times item "one" occurred in the tuple.
```

Out[11]:
```
1
```

tuple membership

In [13]:
```
mytuple
```

Out[13]:
```
('one', 'two', 'two', 'four', 'three', 'four', 'five', 'six', 'seven', 'eight')
```

In [14]:
```
'one' in mytuple  # Check if 'one' exist in the list
```

Out[14]:
```
True
```

In [15]:
```
'ten' in mytuple # Check if 'ten' exist in the list
```

Out[15]:
```
False
```

In [18]:
```
if 'two' in mytuple: # Check if 'two' exist in the list
    print('two is present in the tuple')
else:
    print('two is not present in the tuple')
```

```
two is present in the tuple
```

index position

In [19]:
```
mytuple
```

Out[19]:
```
('one', 'two', 'two', 'four', 'three', 'four', 'five', 'six', 'seven', 'eight')
```

In [20]:
```
mytuple.index('five') # Index of first element equal to 'five'
```

Out[20]:
```
6
```

In [24]:
```
mytuple.index('one') # Index of first element equal to 'one'
```

Out[24]:
```
0
```

sorting

In [26]:
```
mytuple2 = (50,60,70,80,90,10,30,20)
```

In [27]:
```
sorted(mytuple2) # Returns a new sorted list and doesn't change original tuple
```

Out[27]:
```
[10, 20, 30, 50, 60, 70, 80, 90]
```

In [28]:
```
sorted (mytuple2, reverse=True) # Sort in descending order
```

Out[28]:
```
[90, 80, 70, 60, 50, 30, 20, 10]
```

In [ ]:
```

```

Sets

set creation

In [1]:
```
myset= {1,2,3,4,5,6,7} #set of numbers
myset
```

Out[1]:
```
{1, 2, 3, 4, 5, 6, 7}
```

In [3]:
```
len(myset) #lengh of the set
```

Out[3]:
```
7
```

In [4]:
```
myset1= {1,1,5,2,2,4,6,3,4} #dublicate is not allowed in set
myset1
```

Out[4]:
```
{1, 2, 3, 4, 5, 6}
```

In [7]:
```
myset2= {10.2,20.4,30.6,} #set of float numbers
myset2
```

Out[7]:
```
{10.2, 20.4, 30.6}
```

In [8]:
```
myset3= {'one','two','three','four','five'} #set of strings
myset3
```

Out[8]:
```
{'five', 'four', 'one', 'three', 'two'}
```

In [9]:
```
myset4= {10,5.4,'hello',(50,60)} #set of mixed data types
myset4
```

Out[9]:
```
{(50, 60), 10, 5.4, 'hello'}
```

In [10]:
```
myset5 = {10,20, "Hello", [15, 32, 62]} # set doesn't allow mutable items like l
myset5
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[10], line 1
----> 1 myset5 = {10,20, "Hello", [15, 32, 62]} # set doesn't allow mutable items
like list
      2 myset5

TypeError: unhashable type: 'list'
```

In [11]:
```python
myset6= set() #creat an empty set
print(type(myset6))
```

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

In [13]:
```python
myset7= set(('one','two','three','four','five'))
myset7
```

Out[13]:
```
{'five', 'four', 'one', 'three', 'two'}
```

loop through a set

In [15]:
```python
myset = {'one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight'}
for i in myset:
    print(i)
```

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

In [16]:
```python
for i in enumerate(myset):
    print(i)
```

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

set membership

In [17]:
```python
myset
```

Out[17]:
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

In [18]:
```python
'two' in myset # checks if two in the set
```

Out[18]:
```
True
```

In [19]:
```python
'ten' in myset # checks if ten in the set
```

```
Out[19]:  False
```

```
In [20]: if 'one' in myset:
             print("one is present in the set")
         else:
             print('one is not present in the set')
```

```
one is present in the set
```

```
In [21]: if 'nine' in myset:
             print('nine is present in the set')
         else:
             print('nine is not present in the set')
```

```
nine is not present in the set
```

add and remove items

```
In [30]: myset
```

```
Out[30]:  {'eleven',
            'five',
            'four',
            'nine',
            'one',
            'seven',
            'six',
            'ten',
            'three',
            'twelve',
            'two'}
```

```
In [34]: myset.add('nine') #add item using the add() method
         myset
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[34], line 1
----> 1 myset.add('nine', 'eight') #add item using the add() method
      2 myset

TypeError: set.add() takes exactly one argument (2 given)
```

```
In [35]: myset.update(['ten','eight','eleven','twelve']) #add multiple items to a set
         myset
```

```
Out[35]:  {'eight',
            'eleven',
            'five',
            'four',
            'nine',
            'one',
            'seven',
            'six',
            'ten',
            'three',
            'twelve',
            'two'}
```

In [36]:
```
myset.remove('eight') #remove item in a set using remove() method
myset
```

Out[36]:
```
{'eleven',
 'five',
 'four',
 'nine',
 'one',
 'seven',
 'six',
 'ten',
 'three',
 'twelve',
 'two'}
```

In [37]:
```
myset.discard('nine') # remove item from a set using discard() method
myset
```

Out[37]:
```
{'eleven',
 'five',
 'four',
 'one',
 'seven',
 'six',
 'ten',
 'three',
 'twelve',
 'two'}
```

In [38]:
```
myset.clear() # Delete all items in a set
myset
```

Out[38]:
```
set()
```

In [39]:
```
del myset # Delete the set object
myset
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[39], line 2
      1 del myset # Delete the set object
----> 2 myset

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

copy set

In [40]:
```
myset = {'one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight'}
myset
```

Out[40]:
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

In [41]:
```
myset1 = myset # Create a new reference "myset1"
myset1
```

Out[41]:
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

In [42]:
```
id(myset) , id(myset1) # The address of both myset & myset1 will be the same
```
Out[42]:
```
(2377867169568, 2377867169568)
```
In [43]:
```
myset2 = myset.copy() # Create a copy of the set
myset2
```
Out[43]:
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```
In [44]:
```
id(myset2) # The address of myset2 will be different from myset
```
Out[44]:
```
2377867170464
```
In [45]:
```

myset.add('ten')
myset
```
Out[45]:
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'ten', 'three', 'two'}
```
In [46]:
```
myset1 # myset1 will be also impacted as it is pointing to the same Set
```
Out[46]:
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'ten', 'three', 'two'}
```
In [47]:
```
myset2 # Copy of the set won't be impacted due to changes made on the original S
```
Out[47]:
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

set operations

union

In [52]:
```
A = {1,2,3,4,5,6}
B = {4,5,6,7,8,9}
C = {8,9,10}
```
In [53]:
```
A | B # Union of A and B (All elements from both sets. NO DUPLICATES)
```
Out[53]:
```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```
In [54]:
```
A.union(B) # Union of A and B
```
Out[54]:
```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```
In [55]:
```
A.union(B, C) # Union of A, B and C.
```
Out[55]:
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

intersection

In [ ]:
```
A = {1,2,3,4,5,6}
B = {4,5,6,7,8,9}
C = {8,9,10}
```

```
In [56]: A & B # Intersection of A and B (Common items in both sets)
```

```
Out[56]: {4, 5, 6}
```

```
In [58]: A.intersection(B) #Intersection of A and B
```

```
Out[58]: {4, 5, 6}
```

Difference

```
In [59]: A = {1,2,3,4,5,6}
         B = {4,5,6,7,8,9}
```

```
In [60]: A-B # set of elements that are only in A but not in B
```

```
Out[60]: {1, 2, 3}
```

```
In [61]: A.difference(B) #difference of set
```

```
Out[61]: {1, 2, 3}
```

```
In [62]: B- A # set of elements that are only in B but not in A
```

```
Out[62]: {7, 8, 9}
```

```
In [63]: B.difference(A)  # difference of set
```

```
Out[63]: {7, 8, 9}
```

symmetric difference

```
In [1]: A = {1,2,3,4,5,6}
        B = {4,5,6,7,8,9}
```

```
In [2]: A ^ B # Symmetric difference (Set of elements in A and B but not in both
```

```
Out[2]: {1, 2, 3, 7, 8, 9}
```

```
In [3]: A.symmetric_difference(B) # Symmetric difference of sets
```

```
Out[3]: {1, 2, 3, 7, 8, 9}
```

subset, superset & disjoint

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

```
In [7]: B.issubset(A) # Set B is said to be the subset of set A if all elements of B are
```

```
Out[7]: True
```

In [8]:
```
A.issuperset(B) # Set A is said to be the superset of set B if all elements of B
```
Out[8]:
```
True
```
In [9]:
```
C.isdisjoint(A) # Two sets are said to be disjoint sets if they have no common e
```
Out[9]:
```
True
```
In [10]:
```
B.isdisjoint(A) # Two sets are said to be disjoint sets if they have no common e
```
Out[10]:
```
False
```
In [ ]:
```

```

Dictionary

create dictionary

In [12]:
```
mydict = dict() # empty dictionary
mydict
```
Out[12]:
```
{}
```
In [13]:
```
mydict = dict() # empty dictionary
mydict
```
Out[13]:
```
{}
```
In [14]:
```
mydict = {1:'one' , 2:'two' , 3:'three'} # dictionary with integer keys
mydict
```
Out[14]:
```
{1: 'one', 2: 'two', 3: 'three'}
```
In [15]:
```
mydict = dict({1:'one' , 2:'two' , 3:'three'}) # Create dictionary using dict()
mydict
```
Out[15]:
```
{1: 'one', 2: 'two', 3: 'three'}
```
In [16]:
```
mydict = {'A':'one' , 'B':'two' , 'C':'three'} # dictionary with character keys
mydict
```
Out[16]:
```
{'A': 'one', 'B': 'two', 'C': 'three'}
```
In [17]:
```
mydict = {1:'one' , 'A':'two' , 3:'three'} # dictionary with mixed keys
mydict
```
Out[17]:
```
{1: 'one', 'A': 'two', 3: 'three'}
```
In [18]:
```
mydict.keys() # Return Dictionary Keys using keys() method
```
Out[18]:
```
dict_keys([1, 'A', 3])
```
In [19]:
```
mydict.values() # Return Dictionary Values using values() method
```

Out[19]:
```
dict_values(['one', 'two', 'three'])
```

In [20]:
```
mydict.items() # Access each key-value pair within a dictionary
```

Out[20]:
```
dict_items([(1, 'one'), ('A', 'two'), (3, 'three')])
```

In [17]:
```
mydict = {1:'one' , 2:'two' , 'A':['anas' , 'john' , 'Maria'], 'B':('Bat' , 'cat
mydict
```

Out[17]:
```
{1: 'one',
 2: 'two',
 'A': ['anas', 'john', 'Maria'],
 'B': ('Bat', 'cat', 'mat')}
```

In [23]:
```
keys = {'a' , 'b' , 'c' , 'd'}
mydict3 = dict.fromkeys(keys) # Create a dictionary from a sequence of keys
mydict3
```

Out[23]:
```
{'d': None, 'a': None, 'b': None, 'c': None}
```

In [24]:
```
keys = {'a' , 'b' , 'c' , 'd'}
value = 10
mydict3 = dict.fromkeys(keys , value) # Create a dictionary from a sequence of
mydict3
```

Out[24]:
```
{'d': 10, 'a': 10, 'b': 10, 'c': 10}
```

Accessing items

In [25]:
```
mydict = {1:'one' , 2:'two' , 3:'three' , 4:'four'}
mydict
```

Out[25]:
```
{1: 'one', 2: 'two', 3: 'three', 4: 'four'}
```

In [28]:
```
mydict[2] # Access item using key
```

Out[28]:
```
'two'
```

In [29]:
```
mydict.get(1) # Access item using get() method
```

Out[29]:
```
'one'
```

Add, Remove & change items

In [31]:
```
mydict1 = {'Name':'anas' , 'ID': 12345 , 'DOB': 2001 , 'Address': 'latur'}
mydict1
```

Out[31]:
```
{'Name': 'anas', 'ID': 12345, 'DOB': 2001, 'Address': 'latur'}
```

In [32]:
```
mydict1['DOB'] = 2000 # Changing Dictionary Items
mydict1['Address'] = 'hyderabad'
mydict1
```

Out[32]:
```
{'Name': 'anas', 'ID': 12345, 'DOB': 2000, 'Address': 'hyderabad'}
```

In [33]:
```
dict1 = {'DOB':2004}
mydict1.update(dict1)
mydict1
```

Out[33]:
```
{'Name': 'anas', 'ID': 12345, 'DOB': 2004, 'Address': 'hyderabad'}
```

In [34]:
```
mydict1['Job'] = 'data scientist' # Adding items in the dictionary
mydict1
```

Out[34]:
```
{'Name': 'anas',
 'ID': 12345,
 'DOB': 2004,
 'Address': 'hyderabad',
 'Job': 'data scientist'}
```

In [35]:
```
mydict1.pop('Job') # Removing items in the dictionary using Pop method
mydict1
```

Out[35]:
```
{'Name': 'anas', 'ID': 12345, 'DOB': 2004, 'Address': 'hyderabad'}
```

In [36]:
```
mydict1.popitem() # A random item is removed
```

Out[36]:
```
('Address', 'hyderabad')
```

In [37]:
```
mydict1
```

Out[37]:
```
{'Name': 'anas', 'ID': 12345, 'DOB': 2004}
```

In [38]:
```
del[mydict1['ID']] # Removing item using del method
mydict1
```

Out[38]:
```
{'Name': 'anas', 'DOB': 2004}
```

In [39]:
```
mydict1.clear() # Delete all items of the dictionary using clear method
mydict1
```

Out[39]:
```
{}
```

Copy dictionary

In [50]:
```
mydict = {'Name':'anas' , 'ID': 12345 , 'DOB': 2001 , 'Address': 'latur'}
mydict
```

Out[50]:
```
{'Name': 'anas', 'ID': 12345, 'DOB': 2001, 'Address': 'latur'}
```

In [51]:
```
mydict1 = mydict # Create a new reference "mydict1"
```

In [52]:
```
id(mydict) , id(mydict1) # The address of both mydict & mydict1 will be the same
```

Out[52]:
```
(2798971212032, 2798971212032)
```

In [53]:
```
mydict2 = mydict.copy() # Create a copy of the dictionary
```

In [54]:
```
id(mydict2) # The address of mydict2 will be different from mydict
```

```
Out[54]: 2798971216256
```

```
In [55]: mydict['Address'] = 'pune'
```

```
In [56]: mydict
```

```
Out[56]: {'Name': 'anas', 'ID': 12345, 'DOB': 2001, 'Address': 'pune'}
```

```
In [57]: mydict1 # mydict1 will be also impacted as it is pointing to the same dictionary
```

```
Out[57]: {'Name': 'anas', 'ID': 12345, 'DOB': 2001, 'Address': 'pune'}
```

```
In [58]: mydict2 # Copy of list won't be impacted due to the changes made in the original
```

```
Out[58]: {'Name': 'anas', 'ID': 12345, 'DOB': 2001, 'Address': 'latur'}
```

Loop through a dictionary

```
In [59]: mydict1
```

```
Out[59]: {'Name': 'anas', 'ID': 12345, 'DOB': 2001, 'Address': 'pune'}
```

```
In [61]: for i in mydict1:
             print(i , ':' , mydict1[i]) # Key & value pair
```

```
Name : anas
ID : 12345
DOB : 2001
Address : pune
```

```
In [62]: for i in mydict1:
             print(mydict1[i]) # Dictionary items
```

```
anas
12345
2001
pune
```

Dictionary Membership

```
In [63]: mydict1
```

```
Out[63]: {'Name': 'anas', 'ID': 12345, 'DOB': 2001, 'Address': 'pune'}
```

```
In [64]: 'Name' in mydict1 # Test if a key is in a dictionary or not.
```

```
Out[64]: True
```

```
In [66]: 'anas' in mydict1 # Membership test can be only done for keys.
```

```
Out[66]: False
```

```
In [67]: 'Address' in mydict1
```

```
Out[67]: True
```

```
In [68]: 'age' in mydict
```

```
Out[68]: False
```

All / Any

```
In [69]: mydict1
```

```
Out[69]: {'Name': 'anas', 'ID': 12345, 'DOB': 2001, 'Address': 'pune'}
```

```
In [70]: all(mydict1) #will return true as there is no false value
```

```
Out[70]: True
```

```
In [71]: any(mydict1) #will returns true if at least one key is truthy.
```

```
Out[71]: True
```

Range

```
In [2]: range(10)
```

```
Out[2]: range(0, 10)
```

```
In [3]: range(10,15)
```

```
Out[3]: range(10, 15)
```

```
In [4]: list(range(10,20))
```

```
Out[4]: [10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
```

```
In [5]: list(range(10))
```

```
Out[5]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
In [15]: r = range (10,20,5)
         r
```

```
Out[15]: range(10, 20, 5)
```

```
In [16]: for i in r:
             print(i)
```

```
10
15
```

```
In [ ]:
```