Tuple

Tuple creation

In [2]:
```
tup1=()                          # Empty tuple
```

In [3]:
```
tup2=(10,30,60)          # tuple of integers numbers
```

In [4]:
```
tup3=(10.77,30.66,89.22)     # tuple of float numbers
```

In [5]:
```
tup4=('one','two','three')    # tuple of strings
```

In [6]:
```
tup5=('san',25,[50,100],(150,90))    # Nested tuples
```

In [7]:
```
tup6=(100,'san',17.765)       # Tuple of mixed data types
```

In [8]:
```
tup7=('san',25,[50,100],[150,90],{'don','david'},(99,22,33))
```

In [11]:
```
len(tup7)     #Length of list
```

Out[11]:
```
6
```

Tuple indexing

In [16]:
```
tup2[0]      # Retreive first element of the tuple
```

Out[16]:
```
0
```

In [18]:
```
tup4[0]     # Retreive first element of the tuple
```

Out[18]:
```
'one'
```

In [20]:
```
tup4[0][0]    # Nested indexing - Access the first character of the first tuple ele
```

Out[20]:
```
'o'
```

In [21]:
```
tup4[-1]    # Last item of the tuple
```

Out[21]:
```
'three'
```

In [22]:
```
tup5[-1]  # Last item of the tuple
```

Out[22]:
```
(150, 90)
```

tuple slicing

In [74]:
```
mytuple = ('one','two','three','four','five','six','seven','eight')
```

In [75]:
```
mytuple[0:3]   # Return all items from 0th to 3rd index location excluding the item
```

Out[75]: ('one', 'two', 'three')

In [76]:
```
mytuple[2:5]    # List all items from 2nd to 5th index location excluding the item
```

Out[76]: ('three', 'four', 'five')

In [77]:
```
mytuple[:3]      # Return first three items
```

Out[77]: ('one', 'two', 'three')

In [78]:
```
mytuple[:2]    # Return first two items
```

Out[78]: ('one', 'two')

Remove and Change and Item

In [79]:
```
mytuple
```

Out[79]: ('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')

In [96]:
```
del mytuple[0]=1       # Tuples are immutable which means we can't CHANGE tuple ite
```

```
  Cell In[96], line 1
    del mytuple[0]=1
                  ^
SyntaxError: invalid syntax
```

In [97]:
```
del  mytuple     # Deleting entire tuple object is possible
```

Loop Through a Tuple

In [130…
```
mytuple=('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')
```

In [131…
```
for i in mytuple:
    print(i)
```

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

In [132…
```
for i in enumerate(mytuple):
    print(i)
```

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

Tuple Membereship

In [133…
```
mytuple
```

Out[133…
```
('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')
```

In [107…
```
'one' in mytuple     # Check if 'one' exist in the list
```

Out[107…
```
True
```

In [108…
```
'ten' in mytuple    # Check if 'ten' exist in the list
```

Out[108…
```
False
```

In [112…
```
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
```

In [113…
```
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 my tuple')
```

```
eleven is not present in my tuple
```

Index Position

In [114…
```
mytuple
```

Out[114…
```
('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')
```

In [115…
```
mytuple.index('one')     # Index of first element equal to 'one'
```

Out[115…
```
0
```

In [116…
```
mytuple.index('five')   # Index of first element equal to 'five'
```

Out[116…
```
4
```

In [117…
```
mytuple
```

Out[117…
```
('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')
```

In [134…
```
mytuple1=('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')
mytuple1.index('one')         # Index of first element equal to 'one'
```

Out[134…
```
0
```

Shorting

In [119…
```
mytuple2=(43,67,99,12,6,90,67)
```

In [120…
```
sorted(mytuple2)        # Returns a new sorted list and doesn't change original tup
```

Out[120…
```
[6, 12, 43, 67, 67, 90, 99]
```

In [121…
```
sorted(mytuple2,reverse=True)      # Sort in descending order
```

Out[121…
```
[99, 90, 67, 67, 43, 12, 6]
```

Sets

Set creation

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

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

In [3]:
```
len(myset)
```

Out[3]:
```
5
```

In [4]:
```
my_set={1,1,2,2,3,4,5,5}    # Duplicate elements are not allowed.
my_set
```

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

In [5]:
```
myset1={1.79,2.08,3.99,4.56,5.45}     # Set of float numbers
myset1
```

Out[5]:
```
{1.79, 2.08, 3.99, 4.56, 5.45}
```

In [7]:
```
myset2={'san','don','ram'}    # Set of Strings
myset2
```

Out[7]:
```
{'don', 'ram', 'san'}
```

In [9]:
```
myset3={10,20,'hola',(11,22,33)}    # Mixed datatypes
myset3
```

Out[9]:
```
{(11, 22, 33), 10, 20, 'hola'}
```

In [11]:
```
myset3={10,20,'hola',[11,22,33]}     # set doesn't allow mutable items like list
myset3
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[11], line 1
----> 1 myset3={10,20,'hola',[11,22,33]}
      2 myset3

TypeError: unhashable type: 'list'
```

In [12]:
```
myset4=set()
print(type(myset4))
```

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

In [13]:
```
my_set1= set(('one','two','three','four'))
my_set1
```

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

Loop through a set

In [17]:
```
myset={'one','two','three','four','five','six','seven','eight'}
for i in myset:
    print(i)
```

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

In [18]:
```
for i in enumerate(myset):
    print(i)
```

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

Set Membership

In [19]:
```
myset
```

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

In [21]:
```
'one' in myset
```

Out[21]: True

In [22]:
```
'ten' in myset
```

Out[22]: False

In [28]:
```
if 'three' in myset:
    print('three is present in the set')
else:
    print('three is not present in the set')
```

three is present in the set

In [29]:
```
if 'eleven' in myset:
    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

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

Out[30]: {'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}

In [32]:
```
myset.add('nine')    # Add item to a set using add() method
myset
```

Out[32]: {'eight', 'five', 'four', 'nine', 'one', 'seven', 'six', 'three', 'two'}

In [33]:
```
myset.update(['TEN','ELEVEN','TWELVE'])  # Add multiple item to a set using update(
myset
```

Out[33]:
```
{'ELEVEN',
 'TEN',
 'TWELVE',
 'eight',
 'five',
 'four',
 'nine',
 'one',
 'seven',
 'six',
 'three',
 'two'}
```

In [35]:
```
myset.remove('nine')   # remove item in a set using remove() method
myset
```

Out[35]:
```
{'ELEVEN',
 'TEN',
 'TWELVE',
 'eight',
 'five',
 'four',
 'one',
 'seven',
 'six',
 'three',
 'two'}
```

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

Out[37]:
```
{'ELEVEN',
 'TWELVE',
 'eight',
 'five',
 'four',
 'one',
 'seven',
 'six',
 'three',
 '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
----> 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 as
```

Out[42]:
```
(2202888728448, 2202888728448)
```

In [45]:
```
my_set= myset.copy()    # Create a copy of the list
my_set
```

Out[45]:
```
{'eight', 'five', 'four', 'nine', 'one', 'seven', 'six', 'three', 'two'}
```

In [46]:
```
id(my_set)    # The address of my_set will be different from myset because my_set i
```

Out[46]:
```
2202882239040
```

In [43]:
```
myset.add('nine')
myset
```

Out[43]:
```
{'eight', 'five', 'four', 'nine', 'one', 'seven', 'six', 'three', 'two'}
```

In [47]:
```
myset1    # myset1 will be also impacted as it is pointing to the same Set
```

Out[47]:
```
{'eight', 'five', 'four', 'nine', 'one', 'seven', 'six', 'three', 'two'}
```

In [48]:
```
my_set
```

Out[48]:
```
{'eight', 'five', 'four', 'nine', 'one', 'seven', 'six', 'three', 'two'}
```

Set Operation

Union

In [54]:
```
a = {1,2,3,4,5}
b = {4,5,6,7,8}
c = {8,9,10}
```

In [55]:
```
a|b    # Union of A and B (All elements from both sets. NO DUPLICATES)
```

Out[55]:
```
{1, 2, 3, 4, 5, 6, 7, 8}
```

In [56]:
```
a.union(b)    # Union of A and B
```

Out[56]:
```
{1, 2, 3, 4, 5, 6, 7, 8}
```

In [57]:
```
a.union(b,c)  # Union of A, B and C.
```

Out[57]:
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

In [60]:
```
"""
updates the set calling the update() method with union of a, b,c.
for the below example set a will be the updateed with union of a,b,c
"""
```

```
a.update(b,c)
a
```

Out[60]: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

Intersection

In [61]:
```
a={1,2,3,4,5}
b={4,5,6,7,8}
```

In [62]:
```
a&b    # Intersection of A and B (Common items in both sets)
```

Out[62]: {4, 5}

In [64]:
```
a.intersection(b)  intersection of a and b
```

```
  Cell In[64], line 1
    a.intersection(b)  intersection of a and b
                       ^
SyntaxError: invalid syntax
```

In [66]:
```
"""
update the set calling the intersection_update()  method with intersection of a,b.
for the below example set a will be the updated with union of a,b
"""
a.intersection_update(b)
a
```

Out[66]: {4, 5}

Difference

In [68]:
```
a={1,2,3,4,5}
b={4,5,6,7,8}
```

In [69]:
```
a-b  # set of elements that are only in A but not in B
```

Out[69]: {1, 2, 3}

In [70]:
```
b-a   # set of elements that are only in B but not in A
```

Out[70]: {6, 7, 8}

In [71]:
```
b.difference(a)
```

Out[71]: {6, 7, 8}

In [86]:
```
"""
update the set calling the difference_update() method with the difference of set
foe below example set b will be  updated with the difference of b & a
"""
b.difference_update(a)
b
```

Out[86]:
```
{4, 5, 6, 7, 8}
```

Symmetric Difference

In [92]:
```
a = {1,2,3,4,5}
b = {4,5,6,7,8}
```

In [93]:
```
a^b    # Symmetric difference (Set of elements in A and B but not in both.
```

Out[93]:
```
{1, 2, 3, 6, 7, 8}
```

In [94]:
```
a.symmetric_difference(b)
```

Out[94]:
```
{1, 2, 3, 6, 7, 8}
```

In [95]:
```
"""
update the set calling the symmetric_difference_update() method with the difference
foe below example set a will be  updated with the difference of a & b
"""
a.symmetric_difference_update(b)
a
```

Out[95]:
```
{1, 2, 3, 6, 7, 8}
```

Subset, Superset & Disjoint

In [96]:
```
a = {1,2,3,4,5,6,7,8,9}
b = {3,4,5,6,7,8}
c = {10,20,30,40}
```

In [97]:
```
b.issubset(a)      # Set B is said to be the subset of set A if all elements of B a
```

Out[97]:
```
True
```

In [101…
```
a.issuperset(b)     # Set A is said to be the superset of set B if all elements of
```

Out[101…
```
True
```

In [103…
```
c.isdisjoint(a)      # Two sets are said to be disjoint sets if they have no common
```

Out[103…
```
True
```

In [104…
```
b.isdisjoint(a)      # Two sets are said to be disjoint sets if they have no common
```

Out[104…
```
False
```

Other Builtin functions

In [105…
```
a
```

Out[105…
```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

In [106…
```
sum(a)
```

Out[106…
```
45
```

In [107…
```
max(a)
```

Out[107…
```
9
```

In [108…
```
min(a)
```

Out[108…
```
1
```

In [109…
```
len(a)
```

Out[109…
```
9
```

In [110…
```
list(enumerate(a))
```

Out[110…
```
[(0, 1), (1, 2), (2, 3), (3, 4), (4, 5), (5, 6), (6, 7), (7, 8), (8, 9)]
```

In [111…
```
d=sorted(a,reverse=True)
d
```

Out[111…
```
[9, 8, 7, 6, 5, 4, 3, 2, 1]
```

In [112…
```
sorted(d)
```

Out[112…
```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [113…
```
sorted(d,reverse=False)
d
```

Out[113…
```
[9, 8, 7, 6, 5, 4, 3, 2, 1]
```

Dictionary

Create Dictionary

In [64]:
```
mydict= dict()   # empty dictionary
mydict
```

Out[64]:
```
{}
```

In [65]:
```
mydict={1:'one',2:'two',3:'three'}    # dictionary with integer keys
mydict
```

Out[65]:
```
{1: 'one', 2: 'two', 3: 'three'}
```

In [66]:
```
mydict = dict({1:'one' , 2:'two' , 3:'three'}) # Create dictionary using dict()
mydict
```

Out[66]: {1: 'one', 2: 'two', 3: 'three'}

In [67]:
```
mydict = {'a':'one' , 'b':'two' , 'c':'three'}    # dictionary with character keys
mydict
```

Out[67]: {'a': 'one', 'b': 'two', 'c': 'three'}

In [68]:
```
mydict = {1:'one' , 'A':'two' , 3:'three'}   # dictionary with mixed keys
mydict
```

Out[68]: {1: 'one', 'A': 'two', 3: 'three'}

In [69]:
```
mydict.keys()      # Return Dictionary Keys using keys() method
```

Out[69]: dict_keys([1, 'A', 3])

In [70]:
```
mydict.values()       # Return Dictionary Values using values() method
```

Out[70]: dict_values(['one', 'two', 'three'])

In [71]:
```
mydict.items()   # Access each key-value pair within a dictionary
```

Out[71]: dict_items([(1, 'one'), ('A', 'two'), (3, 'three')])

In [72]:
```
mydict = {1:'one' , 2:'two' , 'A':['asif' , 'john' , 'Maria']}
mydict
```

Out[72]: {1: 'one', 2: 'two', 'A': ['asif', 'john', 'Maria']}

In [73]:
```
mydict = {1:'one',2:'two','a':{'name':'san','age':22},'b':{'bat','cat','hat'}}
mydict
```

Out[73]:
```
{1: 'one',
 2: 'two',
 'a': {'name': 'san', 'age': 22},
 'b': {'bat', 'cat', 'hat'}}
```

In [74]:
```
keys = {'a','b','c','d'}            # Create a dictionary from a sequence of keys
mydict3 = dict.fromkeys(keys)
mydict3
```

Out[74]: {'d': None, 'c': None, 'b': None, 'a': None}

In [75]:
```
keys = {'a' , 'b' , 'c' , 'd'}       # Create a dictionary from a sequence of keys
value = 10
mydict3 = dict.fromkeys(keys , value)
mydict3
```

Out[75]: {'d': 10, 'c': 10, 'b': 10, 'a': 10}

In [76]:
```
keys = {'a','b','c','d'}
value = {10,20,30}
```

```
mydict3 = dict.fromkeys(keys,value)
mydict3
```

Out[76]:
```
{'d': {10, 20, 30}, 'c': {10, 20, 30}, 'b': {10, 20, 30}, 'a': {10, 20, 30}}
```

In [77]:
```
value.append(40)
mydict3
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[77], line 1
----> 1 value.append(40)
      2 mydict3

AttributeError: 'set' object has no attribute 'append'
```

Accessing Items

In [78]:
```
mydict = {1:'one',2:'two',3:'three',4:'four'}
mydict
```

Out[78]:
```
{1: 'one', 2: 'two', 3: 'three', 4: 'four'}
```

In [79]:
```
mydict[3]      # Access item using key
```

Out[79]:
```
'three'
```

In [80]:
```
mydict.get(1)     # Access item using get() method
```

Out[80]:
```
'one'
```

In [81]:
```
mydict1 = {'name':'san','id':'15420','DOB':'2001','job':'data science'}
mydict1
```

Out[81]:
```
{'name': 'san', 'id': '15420', 'DOB': '2001', 'job': 'data science'}
```

In [82]:
```
mydict1['name']        # Access item using key
```

Out[82]:
```
'san'
```

In [83]:
```
mydict1.get('job')      # Access item using get() method
```

Out[83]:
```
'data science'
```

Add, Remove & Change Item

In [84]:
```
mydict1 = {'name':'san','id':'15420','DOB':'2001','Address':'Kolhapur'}
mydict1
```

Out[84]:
```
{'name': 'san', 'id': '15420', 'DOB': '2001', 'Address': 'Kolhapur'}
```

In [85]:
```
mydict1['DOB'] = 1999             # Changing Dictionary Items
mydict1['Address'] = 'Goa'
```

```
mydict1
```

Out[85]:
```
{'name': 'san', 'id': '15420', 'DOB': 1999, 'Address': 'Goa'}
```

In [86]:
```
dict1={'DOB':1889}
mydict1. update(dict1)
mydict1
```

Out[86]:
```
{'name': 'san', 'id': '15420', 'DOB': 1889, 'Address': 'Goa'}
```

In [87]:
```
mydict1['Job'] = 'analyst'          # Adding items in the dictionary
mydict1
```

Out[87]:
```
{'name': 'san', 'id': '15420', 'DOB': 1889, 'Address': 'Goa', 'Job': 'analyst'}
```

In [88]:
```
mydict1.pop('Job')             # Removing items in the dictionary using Pop method
```

Out[88]:
```
'analyst'
```

In [89]:
```
mydict1.popitem()             # A random item is removed
```

Out[89]:
```
('Address', 'Goa')
```

In [90]:
```
mydict1
```

Out[90]:
```
{'name': 'san', 'id': '15420', 'DOB': 1889}
```

In [91]:
```
del[mydict1['id']]                # Removing item using del method
mydict1
```

Out[91]:
```
{'name': 'san', 'DOB': 1889}
```

In [92]:
```
mydict1.clear()            # Delete all items of the dictionary using clear method
mydict1.clear()
mydict1
```

Out[92]:
```
{}
```

In [93]:
```
del mydict1         # Delete the dictionary object
mydict1
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[93], line 2
      1 del mydict1         # Delete the dictionary object
----> 2 mydict1

NameError: name 'mydict1' is not defined
```

Copy Dictionary

In [94]:
```
mydict = {'Name':'san' , 'ID': 12345 , 'DOB': 2002 , 'Address' : 'kolhapur'}
mydict
```

Out[94]:
```
{'Name': 'san', 'ID': 12345, 'DOB': 2002, 'Address': 'kolhapur'}
```

In [95]:
```
mydict1 = mydict        # Create a new reference "mydict1"
```

In [96]:
```
id(mydict), id(mydict1)     # The address of both mydict & mydict1 will be the same
```

Out[96]:
```
(2006133159680, 2006133159680)
```

In [97]:
```
mydict2 = mydict.copy()      # Create a copy of the dictionary
```

In [98]:
```
id(mydict2)       # The address of mydict2 will be different from mydict
```

Out[98]:
```
2006133126912
```

In [102…
```
mydict['address'] = 'mumbai'
```

In [104…
```
mydict
```

Out[104…
```
{'Name': 'san',
 'ID': 12345,
 'DOB': 2002,
 'Address': 'kolhapur',
 'address': 'mumbai'}
```

In [105…
```
mydict1     # mydict1 will be also impacted as it is pointing to the same dictionar
```

Out[105…
```
{'Name': 'san',
 'ID': 12345,
 'DOB': 2002,
 'Address': 'kolhapur',
 'address': 'mumbai'}
```

In [106…
```
mydict2       # Copy of list won't be impacted due to the changes made in the origi
```

Out[106…
```
{'Name': 'san', 'ID': 12345, 'DOB': 2002, 'Address': 'kolhapur'}
```

Loop Through a Dictionary

In [114…
```
mydict1 = {'Name':'san' , 'ID': 12345 , 'DOB': 2002 , 'Address' :'kolhapur'}
mydict1
```

Out[114…
```
{'Name': 'san', 'ID': 12345, 'DOB': 2002, 'Address': 'kolhapur'}
```

In [115…
```
for i in mydict1:                                  # Key & value pair
    print(i,':',mydict1[i])
```

```
Name : san
ID : 12345
DOB : 2002
Address : kolhapur
```

In [117…
```
for i in mydict1:            # Dictionary items
    print(mydict1[i])
```

```
san
12345
2002
kolhapur
```

Dictionary Membership

In [118…
```
mydict1 = {'Name':'san' , 'ID': 12345 , 'DOB': 2002 , 'Address' :'kolhapur'}
mydict1
```

Out[118…
```
{'Name': 'san', 'ID': 12345, 'DOB': 2002, 'Address': 'kolhapur'}
```

In [120…
```
'Name' in mydict1      # Test if a key is in a dictionary or not.
```

Out[120…
```
True
```

In [121…
```
'san' in mydict1       # Membership test can be only done for keys.
```

Out[121…
```
False
```

In [124…
```
'ID' in mydict1
```

Out[124…
```
True
```

In [126…
```
'Address' in mydict1
```

Out[126…
```
True
```

All / Any

In [127…
```
mydict1 = {'Name':'san' , 'ID': 12345 , 'DOB': 2002 , 'Address' :'kolhapur'}
mydict1
```

Out[127…
```
{'Name': 'san', 'ID': 12345, 'DOB': 2002, 'Address': 'kolhapur'}
```

In [128…
```
all(mydict1)
```

Out[128…
```
True
```