Sets & Dictionary Practise

Sets

In [1]:
```
#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.
```

In [2]:
```
myset = {1,2,3,4,5}
myset
```

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

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

Out[3]:
```
set
```

In [4]:
```
len(myset)
```

Out[4]:
```
5
```

In [5]:
```
my_set = {1,1,2,2,3,4,5,5}
my_set
```

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

In [7]:
```
myset2 = {'arif','john','kennedy'}
myset2
```

Out[7]:
```
{'arif', 'john', 'kennedy'}
```

In [8]:
```
myset1 = {23.44,44.6,44.4,69,98,33.3}
myset1
```

Out[8]:
```
{23.44, 33.3, 44.4, 44.6, 69, 98}
```

In [9]:
```
myset3 = {"Nit",12,43.5,10+2j,(10,12,13),True}
myset3
```

Out[9]:
```
{(10+2j), (10, 12, 13), 12, 43.5, 'Nit', True}
```

In [11]:
```
myset3 = {"Nit",12,43.5,10+2j,[10,12,13],True} #it doesnt allow mutable items li
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[11], line 1
----> 1 myset3 = {"Nit",12,43.5,10+2j,[10,12,13],True}

TypeError: unhashable type: 'list'
```

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

Out[12]:
```
set()
```

In [14]:
```
print(type(myset4))
```

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

In [15]:
```
my_set1 = set(('one','two','three','four'))
my_set1
```

Out[15]:
```
{'four', 'one', 'three', 'two'}
```

looping through a set

In [19]:
```
myset = {'one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight'}
for i in myset:
    print(i)
```

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

In [20]:
```
myset[0]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[20], line 1
----> 1 myset[0]

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

In [21]:
```
for i in enumerate(myset):
    print(i)
```

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

set Membership

In [22]:
```
myset
```

Out[22]:
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

In [23]:
```
'one' in myset
```

Out[23]: True

In [24]:
```
'ten' in myset
```

Out[24]: False

In [25]:
```
'ten' not in myset
```

Out[25]: True

In [26]:
```
if 'three' in myset:
    print('three is in the set')
else:
    print('three is not in the set')
```

```
three is in the set
```

In [27]:
```
if 'ten' in myset:
    print('ten is in the set')
else:
    print('ten is not in the set')
```

```
ten is not in the set
```

Add & Remove elements

In [28]:
```
myset
```

Out[28]: {'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}

In [29]:
```
myset.add('NINE') # Add item to a set using add() method
myset
```

Out[29]: {'NINE', 'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}

In [33]:
```
myset.update({'TEN','ELEVEN','TWELVE'})
myset
```

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

In [34]:
```
myset.remove('NINE')
myset
```

Out[34]:
```
{'ELEVEN',
 'TEN',
 'TWELVE',
 'eight',
 'five',
 'four',
 'one',
 'seven',
 'six',
 'three',
 'two'}
```

In [35]:
```
myset.discard('TEN')
myset
```

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

In [36]:
```
myset.clear()
myset
```

Out[36]:
```
set()
```

In [37]:
```
del myset
```

In [38]:
```
myset
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[38], line 1
----> 1 myset

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

Copy Set

In [1]:
```
myset = {'one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight'}
myset
```

Out[1]:
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

In [2]:
```
myset1 = myset.copy()
myset1
```

Out[2]:
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

In [3]:
```
id(myset), id(myset1)
```

Out[3]: (2486780488928, 2486780488032)

In [4]:
```
my_set = myset.copy()
my_set
```

Out[4]: {'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}

In [5]:
```
id(my_set)
```

Out[5]: 2486780489824

In [6]:
```
myset.add('nine')
myset
```

Out[6]: {'eight', 'five', 'four', 'nine', 'one', 'seven', 'six', 'three', 'two'}

In [7]:
```
myset
```

Out[7]: {'eight', 'five', 'four', 'nine', 'one', 'seven', 'six', 'three', 'two'}

In [8]:
```
myset1
```

Out[8]: {'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}

set operation

union

In [18]:
```
A = {1,2,3,4,5}
B = {4,5,6,7,8}
C = {8,9,10}
print(A)
print(B)
print(C)
```

```
{1, 2, 3, 4, 5}
{4, 5, 6, 7, 8}
{8, 9, 10}
```

In [19]:
```
A | B
```

Out[19]: {1, 2, 3, 4, 5, 6, 7, 8}

In [20]:
```
B | C
```

Out[20]: {4, 5, 6, 7, 8, 9, 10}

In [21]:
```
C | A
```

Out[21]: {1, 2, 3, 4, 5, 8, 9, 10}

In [22]:
```
"""
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
```

Out[22]: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

intersection

In [24]:
```
A = {1,2,3,4,5}
B = {4,5,6,7,8}
C = {8,9,10}
print(A)
print(B)
print(C)
```

```
{1, 2, 3, 4, 5}
{4, 5, 6, 7, 8}
{8, 9, 10}
```

In [25]:
```
A & B
```

Out[25]: {4, 5}

In [26]:
```
B & C
```

Out[26]: {8}

In [27]:
```
C & A
```

Out[27]: set()

In [34]:
```
A = {1,2,3,4,5}
A
```

Out[34]: {1, 2, 3, 4, 5}

In [36]:
```
A.intersection_update(B)
A #The elements of A is replaced with only common elements b/w A and B set
```

Out[36]: {4, 5}

Difference

In [38]:
```
A = {1,2,3,4,5}
B = {4,5,6,7,8}
C = {8,9,10}
print(A)
print(B)
print(C)
```

```
{1, 2, 3, 4, 5}
{4, 5, 6, 7, 8}
{8, 9, 10}
```

In [39]:
```
A - B
```

Out[39]: {1, 2, 3}

In [40]:
```
B - C
```

Out[40]: {4, 5, 6, 7}

In [41]:
```
C - A
```

Out[41]: {8, 9, 10}

In [42]:
```
B.difference_update(A)
B
```

Out[42]: {6, 7, 8}

symmetric Diff

In [43]:
```
A = {1,2,3,4,5}
B = {4,5,6,7,8}
```

In [44]:
```
print(A)
print(B)
```

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

In [45]:
```
A ^ B
```

Out[45]: {1, 2, 3, 6, 7, 8}

In [46]:
```
B ^ A
```

Out[46]: {1, 2, 3, 6, 7, 8}

In [47]:
```
A.symmetric_difference(B)
```

Out[47]: {1, 2, 3, 6, 7, 8}

In [48]:
```
A.symmetric_difference_update(B)
A
```

Out[48]: {1, 2, 3, 6, 7, 8}

subset, superset & disjoint

In [49]:
```
A = {1,2,3,4,5,6,7,8,9}
B = {3,4,5,6,7,8}
C = {10,20,30,40}
print(A)
print(B)
print(C)
```

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

In [50]:
```
A.issuperset(B)
```

Out[50]: True

In [51]:
```
B.issubset(A)
```

Out[51]: True

In [52]:
```
C.isdisjoint(A)
```

Out[52]: True

In [53]:
```
B.isdisjoint(A)
```

Out[53]: False

Other Built-In Functions

In [54]:
```
A
```

Out[54]: {1, 2, 3, 4, 5, 6, 7, 8, 9}

In [55]:
```
max(A)
```

Out[55]: 9

In [56]:
```
min(A)
```

Out[56]: 1

In [57]:
```
sum(A)
```

Out[57]: 45

In [58]:
```
sorted(A,reverse=True)
```

Out[58]: [9, 8, 7, 6, 5, 4, 3, 2, 1]

In [59]:
```
list(enumerate(A))
```

Out[59]: [(0, 1), (1, 2), (2, 3), (3, 4), (4, 5), (5, 6), (6, 7), (7, 8), (8, 9)]

In [60]:
```
len(A)
```

Out[60]: 9

In [61]:
```
A
```

Out[61]: {1, 2, 3, 4, 5, 6, 7, 8, 9}

Dictionary

In [62]:
```
#Dictionary is a mutable data type in Python.
#A python dictionary is a collection of key and value pairs separated by a colon
#in curly braces {}.
#Keys must be unique in a dictionary, duplicate values are allowed.
```

create dict

In [67]:
```
mydict = dict()
mydict
```

Out[67]: {}

In [68]:
```
print(type(mydict))
```

```
<class 'dict'>
```

In [69]:
```
mydict = {}
mydict
```

Out[69]: {}

In [70]:
```
mydict = {1:'one', 2:'two',3:'three'}
mydict
```

Out[70]: {1: 'one', 2: 'two', 3: 'three'}

In [71]:
```
mydict.keys()
```

Out[71]: dict_keys([1, 2, 3])

In [72]:
```
mydict.items()
```

Out[72]: dict_items([(1, 'one'), (2, 'two'), (3, 'three')])

In [73]:
```
mydict = dict({1:'one', 2:'two',3:'three'})
mydict
```

Out[73]: {1: 'one', 2: 'two', 3: 'three'}

In [74]:
```
mydict.values()
```

Out[74]: dict_values(['one', 'two', 'three'])

In [76]:
```
mydict = {1:'one' , 2:'two' , 'A':['asif' , 'john' , 'Maria']}
mydict
```

Out[76]: {1: 'one', 2: 'two', 'A': ['asif', 'john', 'Maria']}

In [77]:
```
mydict = {1:'one' , 2:'two' , 'A':['asif' , 'john' , 'Maria'], 'B':('Bat' , 'car
mydict
```

Out[77]:
```
{1: 'one',
 2: 'two',
 'A': ['asif', 'john', 'Maria'],
 'B': ('Bat', 'carol', 'jake')}
```

In [79]:
```
mydict = {1:'one' , 2:'two' , 'A':{'Name':'asif' , 'Age' :20}, 'B':('Bat' , 'cat
mydict
```

Out[79]:
```
{1: 'one',
 2: 'two',
 'A': {'Name': 'asif', 'Age': 20},
 'B': ('Bat', 'cat', 'hat')}
```

In [80]:
```
keys = {'a' , 'b' , 'c' , 'd'}
mydict3 = dict.fromkeys(keys) # Create a dictionary from a sequence of keys
mydict3
```

Out[80]:
```
{'d': None, 'b': None, 'a': None, 'c': None}
```

In [82]:
```
keys = {'a' , 'b' , 'c' , 'd'}
value = 10
mydict3 = dict.fromkeys(keys , value) # Create a dictionary from a sequence of v
mydict3
```

Out[82]:
```
{'d': 10, 'b': 10, 'a': 10, 'c': 10}
```

In [83]:
```
keys = {'a' , 'b' , 'c' , 'd'}
value = [10,20,30]
mydict3 = dict.fromkeys(keys , value) # Create a dictionary from a sequence of
mydict3
```

Out[83]:
```
{'d': [10, 20, 30], 'b': [10, 20, 30], 'a': [10, 20, 30], 'c': [10, 20, 30]}
```

In [84]:
```
value.append(40)
mydict3
```

Out[84]:
```
{'d': [10, 20, 30, 40],
 'b': [10, 20, 30, 40],
 'a': [10, 20, 30, 40],
 'c': [10, 20, 30, 40]}
```

Accessing items

In [85]:
```
mydict = {1:'one' , 2:'two' , 3:'three' , 4:'four'}
mydict
```

Out[85]:
```
{1: 'one', 2: 'two', 3: 'three', 4: 'four'}
```

In [88]:
```
mydict.get(1) #access itmes using get() method or
```

Out[88]:
```
'one'
```

In [89]:
```
mydict[1] #access using index
```

Out[89]:
```
'one'
```

In [90]:
```
mydict1 = {' Name' : 'Aksay', 'ID':4567, 'Age': 69 , 'Salary':69000}
mydict1
```

Out[90]:
```
{' Name': 'Aksay', 'ID': 4567, 'Age': 69, 'Salary': 69000}
```

In [94]:
```
mydict.get(2) #using index
```

Out[94]:
```
'two'
```

In [95]:
```
mydict1['Salary'] #usings keys
```

Out[95]:
```
69000
```

In [98]:
```
mydict1.get(' Name')
```

Out[98]:
```
'Aksay'
```

add,remove & change items

In [1]:
```
mydict1 = {'Name':'Asif','Age':69,'Job':'Soft Dev','Salary':69000}
```

In [2]:
```
mydict1
```

Out[2]:
```
{'Name': 'Asif', 'Age': 69, 'Job': 'Soft Dev', 'Salary': 69000}
```

In [4]:
```
mydict1['Salary']=96000
mydict1['Age']=45
mydict1
```

Out[4]:
```
{'Name': 'Asif', 'Age': 45, 'Job': 'Soft Dev', 'Salary': 96000}
```

In [8]:
```
dict1={'Salary':69000}
mydict1.update(dict1)
```

In [9]:
```
mydict1
```

Out[9]:
```
{'Name': 'Asif', 'Age': 45, 'Job': 'Soft Dev', 'Salary': 69000}
```

In [10]:
```
mydict1.pop('Job')
```

Out[10]:
```
'Soft Dev'
```

In [11]:
```
mydict1
```

Out[11]:
```
{'Name': 'Asif', 'Age': 45, 'Salary': 69000}
```

In [12]:
```
mydict1['Job'] = 'Analyst'
mydict1
```

Out[12]:
```
{'Name': 'Asif', 'Age': 45, 'Salary': 69000, 'Job': 'Analyst'}
```

In [16]:
```
mydict1 = {'Name': 'Asif', 'Age': 45, 'Salary': 69000, 'Job': 'Analyst'}
```

In [17]:
```
mydict1.popitem()
mydict1
```

Out[17]:
```
{'Name': 'Asif', 'Age': 45, 'Salary': 69000}
```

In [21]:
```
del[mydict1['Salary']]
```

In [22]:
```
mydict1
```

Out[22]:
```
{'Name': 'Asif', 'Age': 45}
```

In [23]:
```
mydict1.clear()
```

In [24]:
```
mydict1
```

Out[24]:
```
{}
```

In [25]:
```
del mydict1
```

In [26]:
```
mydict1
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[26], line 1
----> 1 mydict1

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

copy dictionary

In [52]:
```
mydict = {'Name':'Asif' , 'ID': 12345 , 'DOB': 1991 , 'Address' : 'Hilsinki'}
mydict
```

Out[52]:
```
{'Name': 'Asif', 'ID': 12345, 'DOB': 1991, 'Address': 'Hilsinki'}
```

In [53]:
```
mydict1=mydict
mydict
```

Out[53]:
```
{'Name': 'Asif', 'ID': 12345, 'DOB': 1991, 'Address': 'Hilsinki'}
```

In [61]:
```
id(mydict1), id(mydict)
```

Out[61]:
```
(2392083502528, 2392083502528)
```

In [62]:
```
mydict2=mydict.copy()
mydict2
```

Out[62]:
```
{'Name': 'Asif', 'ID': 12345, 'DOB': 1991, 'Address': ['Mumbai']}
```

In [67]:
```
mydict['Address'] =['HYD']
mydict
```

Out[67]:
```
{'Name': 'Asif', 'ID': 12345, 'DOB': 1991, 'Address': ['HYD']}
```

In [68]:
```
mydict1
```

Out[68]:
```
{'Name': 'Asif', 'ID': 12345, 'DOB': 1991, 'Address': ['HYD']}
```

In [69]:
```
mydict2
```

Out[69]:
```
{'Name': 'Asif', 'ID': 12345, 'DOB': 1991, 'Address': ['Mumbai']}
```

loop through a dict

In [70]:
```
mydict = {'Name':'Asif' , 'ID': 12345 , 'DOB': 1991 , 'Address' : 'Hilsinki'}
mydict
```

Out[70]:
```
{'Name': 'Asif', 'ID': 12345, 'DOB': 1991, 'Address': 'Hilsinki'}
```

In [73]:
```
for i in mydict: #its only giving me keys but not values
    print(i)
```

```
Name
ID
DOB
Address
```

In [74]:
```
for i in enumerate(mydict):
    print(i)
```

```
(0, 'Name')
(1, 'ID')
(2, 'DOB')
(3, 'Address')
```

In [76]:
```
for i in mydict:
    print(i,':',mydict[i]) #by using this we can print both keys and values
```

```
Name : Asif
ID : 12345
DOB : 1991
Address : Hilsinki
```

In [80]:
```
for i in mydict:
    print(mydict[i]) #it will print only values not their keys
```

```
Asif
12345
1991
Hilsinki
```

dictionary membership

In [82]:
```
mydict = {'Name':'Asif' , 'ID': 12345 , 'DOB': 1991 , 'Address' : 'Hilsinki'}
mydict
```

Out[82]:
```
{'Name': 'Asif', 'ID': 12345, 'DOB': 1991, 'Address': 'Hilsinki'}
```

In [83]:
```
'Name' in mydict
```

Out[83]: True

In [85]:
```
'Asif' in mydict  # Membership test can be only done for keys.
```

Out[85]: False

In [86]:
```
'ID' not in mydict
```

Out[86]: False

All / Any

In [87]:
```
#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, a
```

In [88]:
```
mydict = {'Name':'Asif' , 'ID': 12345 , 'DOB': 1991 , 'Address' : 'Hilsinki'}
mydict
```

Out[88]: {'Name': 'Asif', 'ID': 12345, 'DOB': 1991, 'Address': 'Hilsinki'}

In [91]:
```
all(mydict)  # Will Return false as one value is false (Value 0)
```

Out[91]: True

In [92]:
```
any(mydict)
```

Out[92]: True

practise completed