```
# SETS
a = set()
print(a)                        # Output => set()
print(type(a))                  # Output => <class 'set'>

a = {5, 2, 7, 1, 4}
print(a)                        # Output => set()
print(type(a))                  # Output => {1, 2, 4, 5, 7}

'''
=> This is based on a data structure known as a hash table.
=> Sets are unordered, we cannot access items using indexes like we do
in lists.
=> The major advantage of using a set, as opposed to a list,
   is that it has a highly optimized method for checking whether a
specific element is contained in the set.
'''
print(a)
```

```
set()
<class 'set'>
{1, 2, 4, 5, 7}
<class 'set'>
{1, 2, 4, 5, 7}
```

```
#adding an element in set.
print(a)
a.add(6)
print(a)                        # Output => {1, 2, 4, 5, 6, 7} => 6 is
added.
a.add(3)
print(a)                        # Output => {1, 2, 3, 4, 5, 6, 7} => 3 is
added.
a.add(0)
print(a)                        # Output => {0, 1, 2, 3, 4, 5, 6, 7} => 0
is added.
```

```
{1, 2, 4, 5, 7}
{1, 2, 4, 5, 6, 7}
{1, 2, 3, 4, 5, 6, 7}
{0, 1, 2, 3, 4, 5, 6, 7}
```

```
a.add(-10)
print(a)
a
```

```
{0, 1, 2, 3, 4, 5, 6, 7, -10}
```

```
{-10, 0, 1, 2, 3, 4, 5, 6, 7}
```

```
a.update([-1,9, 10, -2])
print(a)
a
```

```
{0, 1, 2, 3, 4, 5, 6, 7, 9, 10, -10, -2, -1}
```

```
{-10, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 9, 10}
```

```
b = {0, 1, 2, 3, 4, 5, 6, 7}
print(7 in b )                  # Output => True
print(-2 in b)                  # Output => False
```

```
True
False
```

```
#discard() => Removes the element from the set only if the element is
present in the set.
#            If the element is not present in the set, then no error
or exception is raised
#            and the original set is printed.
print(b)
b.discard(5)
print(b)                        # Output => {0, 1, 2, 3, 4, 6, 7}
b.discard(5)
print(b)                        # Output => {0, 1, 2, 3, 4, 6, 7}
''' There is no error after using discard() for the same element again
an again. '''
b.discard(10)
print(b)                        # Output => {0, 1, 2, 3, 4, 6, 7}

''' There is also no error after using discard()
    for the element which is not present in the given set. '''

# NOTE => While remove() function will show error.

print(b)
```

```
{0, 1, 2, 3, 4, 6, 7}
{0, 1, 2, 3, 4, 6, 7}
{0, 1, 2, 3, 4, 6, 7}
{0, 1, 2, 3, 4, 6, 7}
{0, 1, 2, 3, 4, 6, 7}
```

```
# remove() => Removes the element from the set only if the element is
present in the set,
#             just as the discard() method does but If the element is
not present in the set,
#             then an error or exception is raised.
print(b)
b.remove(3)
print(b)                       # Output => {0, 1, 2, 4, 6, 7} =>
```

```
removed 3 from the given set.

b.remove(3)                    # shows an error because the element 3
is already removed from the set.
print(b)

b.remove(9)
print(b)                       # Shows an error.


{0, 1, 2, 3, 4, 6, 7}
{0, 1, 2, 4, 6, 7}

---------------------------------------------------------------------------
-----
KeyError                                  Traceback (most recent call
last)
c:\Users\PKVidyarthi\Desktop\Data Science\Notes\Sets&Tuples.ipynb Cell
7 in <cell line: 8>()
      <a
href='vscode-notebook-cell:/c%3A/Users/PKVidyarthi/Desktop/Data
%20Science/Notes/Sets%26Tuples.ipynb#ch0000006?line=4'>5</a>
b.remove(3)
      <a
href='vscode-notebook-cell:/c%3A/Users/PKVidyarthi/Desktop/Data
%20Science/Notes/Sets%26Tuples.ipynb#ch0000006?line=5'>6</a> print(b)
# Output => {0, 1, 2, 4, 6, 7} => removed 3 from the given set.
----> <a
href='vscode-notebook-cell:/c%3A/Users/PKVidyarthi/Desktop/Data
%20Science/Notes/Sets%26Tuples.ipynb#ch0000006?line=7'>8</a>
b.remove(3)                    # shows an error because the element 3
is already removed from the set.
      <a
href='vscode-notebook-cell:/c%3A/Users/PKVidyarthi/Desktop/Data
%20Science/Notes/Sets%26Tuples.ipynb#ch0000006?line=8'>9</a> print(b)
     <a
href='vscode-notebook-cell:/c%3A/Users/PKVidyarthi/Desktop/Data
%20Science/Notes/Sets%26Tuples.ipynb#ch0000006?line=10'>11</a>
b.remove(9)

KeyError: 3

#pop() => The pop() method removes a random item from the set.
        # This method returns the removed item.
print(b)
x = b.pop()           # Removed element will be store in x.
print(x)              # Output => 0  => Removed 0
print(b)              # Output => {1, 2, 4, 6, 7}
```

```
{0, 1, 2, 4, 6, 7}
0
{1, 2, 4, 6, 7}
```

```
# pop() deletes random items from set.
y = b.pop()
print(y)
print(b)
```

```
1
{2, 4, 6, 7}
```

```
set1 = {'apple', 'cherry', 'banana', 'orange', 'grapes'}
print(set1)
```

```
{'grapes', 'banana', 'apple', 'orange', 'cherry'}
```

```
set1.add('pineapple')
print(set1)
```

```
{'grapes', 'banana', 'apple', 'orange', 'pineapple', 'cherry'}
```

```
set1.update(['mango', 'berry'])
print(set1)
```

```
{'grapes', 'banana', 'berry', 'apple', 'mango', 'orange', 'pineapple',
'cherry'}
```

```
set1.discard('berry')
print(set1)
```

```
set1.discard('melon')
```

```
# This cell can be run again and again, doesn't show any error.
```

```
{'grapes', 'banana', 'apple', 'mango', 'orange', 'pineapple',
'cherry'}
```

```
set1.remove('cherry')
print('cherry');
```

```
cherry
```

```
set1.remove('cherry')
print(set1)
# shows error because 'cherry' is already removed.
```

```
---------------------------------------------------------------------
-----
KeyError                                  Traceback (most recent call
last)
c:\Users\PKVidyarthi\Desktop\Data Science\Notes\Sets&Tuples.ipynb Cell
```

```
15 in <cell line: 1>()
----> <a
href='vscode-notebook-cell:/c%3A/Users/PKVidyarthi/Desktop/Data
%20Science/Notes/Sets%26Tuples.ipynb#ch0000014?line=0'>1</a>
set1.remove('cherry')
      <a
href='vscode-notebook-cell:/c%3A/Users/PKVidyarthi/Desktop/Data
%20Science/Notes/Sets%26Tuples.ipynb#ch0000014?line=1'>2</a>
print(set1)

KeyError: 'cherry'
```

```
set1.remove('melon')
print(set1)
# shows error because 'melon' does not exist in the given set.
```

```
---------------------------------------------------------------------------
-----
KeyError                                  Traceback (most recent call
last)
c:\Users\PKVidyarthi\Desktop\Data Science\Notes\Sets&Tuples.ipynb Cell
16 in <cell line: 1>()
----> <a
href='vscode-notebook-cell:/c%3A/Users/PKVidyarthi/Desktop/Data
%20Science/Notes/Sets%26Tuples.ipynb#ch0000015?line=0'>1</a>
set1.remove('melon')
      <a
href='vscode-notebook-cell:/c%3A/Users/PKVidyarthi/Desktop/Data
%20Science/Notes/Sets%26Tuples.ipynb#ch0000015?line=1'>2</a>
print(set1)

KeyError: 'melon'
```

```
popped = set1.pop()
# It will remove random element from the set and will store in
variable 'popped'.
print(popped)
# popped => removed element.
print(set1)
# set1 => updated set.


grapes
{'banana', 'apple', 'mango', 'orange', 'pineapple'}
```

```
# clear()
b.clear()
print(b)
# can be run again and again.

set()
```

```
# del => delete the given set from root.
del b
print(b)              # Error => name 'b' is not defined
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call
last)
c:\Users\PKVidyarthi\Desktop\Data Science\Notes\Sets&Tuples.ipynb Cell
19 in <cell line: 3>()
      <a
href='vscode-notebook-cell:/c%3A/Users/PKVidyarthi/Desktop/Data
%20Science/Notes/Sets%26Tuples.ipynb#ch0000018?line=0'>1</a> # del =>
delete the given set from root.
      <a
href='vscode-notebook-cell:/c%3A/Users/PKVidyarthi/Desktop/Data
%20Science/Notes/Sets%26Tuples.ipynb#ch0000018?line=1'>2</a> del b
----> <a
href='vscode-notebook-cell:/c%3A/Users/PKVidyarthi/Desktop/Data
%20Science/Notes/Sets%26Tuples.ipynb#ch0000018?line=2'>3</a> print(b)

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

```
print(set1)           # Output => {'orange', 'pineapple', 'banana',
'mango', 'apple'}
del set1
print(set1)           # Error => name 'set1' is not defined
```

```
{'banana', 'apple', 'mango', 'orange', 'pineapple'}
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call
last)
c:\Users\PKVidyarthi\Desktop\Data Science\Notes\Sets&Tuples.ipynb Cell
20 in <cell line: 3>()
      <a
href='vscode-notebook-cell:/c%3A/Users/PKVidyarthi/Desktop/Data
%20Science/Notes/Sets%26Tuples.ipynb#ch0000019?line=0'>1</a>
print(set1)           # Output => {'orange', 'pineapple', 'banana',
'mango', 'apple'}
      <a
href='vscode-notebook-cell:/c%3A/Users/PKVidyarthi/Desktop/Data
%20Science/Notes/Sets%26Tuples.ipynb#ch0000019?line=1'>2</a> del set1
----> <a
href='vscode-notebook-cell:/c%3A/Users/PKVidyarthi/Desktop/Data
%20Science/Notes/Sets%26Tuples.ipynb#ch0000019?line=2'>3</a>
print(set1)

NameError: name 'set1' is not defined
```

```
b = set(['a','e','i','u','o'])
print(b)              # Output => {'o', 'a', 'i', 'u', 'e'}
print(type(b))        # Output => <class 'set'>
print(len(b))         # Output => 5
b                     # Output => {'a', 'e', 'i', 'o', 'u'}
```

```
{'a', 'i', 'e', 'u', 'o'}
<class 'set'>
5

{'a', 'e', 'i', 'o', 'u'}
```

```
A = set([1, 2, 3, 4, 5])
B = {0, 4, 5, 6, 7, 8}

print(A)
print(type(A))
print(B)
print(type(B))
```

```
{1, 2, 3, 4, 5}
<class 'set'>
{0, 4, 5, 6, 7, 8}
<class 'set'>
```

```
# Union Operator
print(A.union(B))
```

```
{0, 1, 2, 3, 4, 5, 6, 7, 8}
```

```
print(A | B)
```

```
{0, 1, 2, 3, 4, 5, 6, 7, 8}
```

```
print(B | A)
```

```
{0, 1, 2, 3, 4, 5, 6, 7, 8}
```

```
# Intersection Operator
C = A.intersection(B)
print(C)
```

```
{4, 5}
```

```
print(A & B)         # Intersection => '&'
```

```
{4, 5}
```

```
print(B & A)
```

```
{4, 5}
```

```
print(A)
print(B)
```

```
{1, 2, 3, 4, 5}
{0, 4, 5, 6, 7, 8}
```

```
# Difference
D = A.difference(B)
print(D)
```

```
{1, 2, 3}
```

```
D = B.difference(A)
print(D)
```

```
{0, 8, 6, 7}
```

```
# Symmetric Difference
SD1 = A.symmetric_difference(B)
print(SD1)
SD2 = B.symmetric_difference(A)
print(SD2)
```

```
{0, 1, 2, 3, 6, 7, 8}
{0, 1, 2, 3, 6, 7, 8}
```

```
''' Symmetric Difference => ^  '''
print(A ^ B)
print(B ^ A)
```

```
{0, 1, 2, 3, 6, 7, 8}
{0, 1, 2, 3, 6, 7, 8}
```

```
print(a)
```

```
{0, 1, 2, 3, 4, 5, 6, 7, 9, 10, -10, -2, -1}
```

```
# copy()
c = a.copy()
print(c)
```

```
{0, 1, 2, 3, 4, 5, 6, 7, 9, 10, -10, -1, -2}
```

```
# clear()
c.clear()
print(c)       # It can be run again and again
```

```
set()
```

```
# del => It deletes set from root.
del c
# after deletion of set 'c' we can't run again or can't print elements
of set 'c'.
# 1st time this cell will run but 2nd time this cell can't run
otherwise it will show an error.
```

```
print(c)            # It will show an error
```

```
---------------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
last)
c:\Users\PKVidyarthi\Desktop\Data Science\Notes\Sets&Tuples.ipynb Cell
38 in <cell line: 1>()
----> <a
href='vscode-notebook-cell:/c%3A/Users/PKVidyarthi/Desktop/Data
%20Science/Notes/Sets%26Tuples.ipynb#ch0000041?line=0'>1</a> print(c)

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

```python
# Converting a set into a list.
c = {-1, 1, 2, 3, 4, 7, 9, 10}
print(c)
print(type(c))            # Output => <class 'set'>

c = list(c)               # set to list.
print(c)                  # Output => [1, 2, 3, 4, 7, 9, 10, -1]
print(type(c))            # Output => <class 'list'>
```

```
{1, 2, 3, 4, 7, 9, 10, -1}
<class 'set'>
[1, 2, 3, 4, 7, 9, 10, -1]
<class 'list'>
```

```python
# TUPLES
'''
=> Tuples are used to store multiple items in a single variable.
=> A tuple is a collection which is ordered and unchangeable.
=> Tuples are written with round brackets.
'''
t = ()                  # Output => ()
print(t)
print(type(t))          # Output => <class 'tuple'>
t = (3, 2, 5, 7, 6, 1)
print(t)                # Output => (3, 2, 5, 7, 6, 1)
print(type(t))          # Output => <class 'tuple'>
```

```
()
<class 'tuple'>
(3, 2, 5, 7, 6, 1)
<class 'tuple'>
```

```python
player = 'MS DHONI', 'VIRAT KOHLI'
print(player)           # Output => ('MS DHONI', 'VIRAT KOHLI')
print(type(player))     # Output => <class 'tuple'>
```

```
('MS DHONI', 'VIRAT KOHLI')
<class 'tuple'>
```

```python
#! Tuples are :
# Ordered => It means that the items have a defined order, and that
order will not change.
# Unchangeable => We cannot change, add or remove items after the
tuple has been created.
# Allow Duplicates => Since tuples are indexed, they can have items
with the same value.

fruits = ('banana', 'apple', 'cherry', 'apple','berry')
print(fruits)           # Output => ('banana', 'apple', 'cherry',
'apple', 'berry')
print(type(fruits))     # Output => <class 'tuple'>
# There is a duplicate item as 'apple'.
```

```
('banana', 'apple', 'cherry', 'apple', 'berry')
<class 'tuple'>
```

```python
a = 6,7,10,1
print(a)               # Output => (6, 7, 10, 1)
print(type(a))         # Output => <class 'tuple'>

# It means tuples can be written without round brackets.
```

```
(6, 7, 10, 1)
<class 'tuple'>
```

```python
# Question : How to print the tuple without enclosing parentheses?
tuple => t = (3, 2, 5, 7, 6, 1)
# Solution:
t = (3, 2, 5, 7, 6, 1)
print(*t)            # Output => 3 2 5 7 6 1

'''
Explanation:
The asterisk operator * is used to unpack an iterable into the
argument list of a given function.
The expression print(*t) will print the elements in my_tuple, empty-
space separated, without the enclosing parentheses!
'''
```

```
3 2 5 7 6 1

' \nExplanation:\nThe asterisk operator * is used to unpack an
iterable into the argument list of a given function. \nThe expression
print(*t) will print the elements in my_tuple, empty-space separated,
without the enclosing parentheses!\n'
```

```python
# How to unpack tuple with separator ?
''' Solution: '''
print(*t, sep = ', ')    # Output => 3, 2, 5, 7, 6, 1
```

```
# can be use any symbol to seperate as place of comma.
print(*t, sep = ' | ')   # Output => 3 | 2 | 5 | 7 | 6 | 1

''' Explanation :
To print a comma-separated tuple without enclosing parentheses,
the most Pythonic way is to unpack all tuple values into the print()
function and
use the sep=', ' argument to separate the tuple elements with a comma
and a space.
Specifically, the expression print(*t, sep=', ') will print the tuple
elements without parentheses and
with a comma between subsequent tuple elements.
'''
```

```
3, 2, 5, 7, 6, 1
3 | 2 | 5 | 7 | 6 | 1
```

```
" Explanation :\nTo print a comma-separated tuple without enclosing
parentheses, \nthe most Pythonic way is to unpack all tuple values
into the print() function and \nuse the sep=', ' argument to separate
the tuple elements with a comma and a space. \nSpecifically, the
expression print(*t, sep=', ') will print the tuple elements without
parentheses and \nwith a comma between subsequent tuple elements.\n"
```

```
a = (5,2,7,4,1,8,11,15,17,21,25)
print(len(a))
```

```
11
```

```
# We can perform slicing, dicing and indexing on tuples.
print(a)
b = a[2:8]
print(b)
```

```
(5, 2, 7, 4, 1, 8, 11, 15, 17, 21, 25)
(7, 4, 1, 8, 11, 15)
```

```
c = a[2:8:2]
print(c)
```

```
(7, 1, 11)
```

```
del a
```

```
print(a)
```

```
(5, 2, 7, 4, 1, 8, 11, 15, 17, 21, 25)
```

```
# Concatenation or merging of a tuple.
t1 = (1, 2, 3, 4)
t2 = (5, 6, 7, 8)
```

```
t3 = t1 + t2
print(t3)             # Output => (1, 2, 3, 4, 5, 6, 7, 8)
```

```
(1, 2, 3, 4, 5, 6, 7, 8)
```

```
a = 'a','p','p','l','e'
print(a)
print(type(a))
```

```
('a', 'p', 'p', 'l', 'e')
<class 'tuple'>
```

```
# print count
# 1
print(a.count('p'))
```

```
2
```

```
# 2
b = a.count('p')
print(b)
```

```
2
```

```
# print index
# 1.
print(a.index('e'))
# 2
i = a.index('e')
print(i)
```

```
4
4
```

```
t1 = (4, 2, 3, [5, 6])
print(t1)                 # Output => (4, 2, 3, [5, 6])
print(type(t1))           # Output => <class 'tuple'>
```

```
(4, 2, 3, [5, 6])
<class 'tuple'>
```

```
i = t1.index(3)
print(i)
```

```
2
```

```
print(t1[2])
# prints 3 bcz 3is at the index position 2.
```

```
3
```

```
print(t1[3])
```

```
[5, 6]
```

```
print(t1[5])
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
c:\Users\PKVidyarthi\Desktop\Data Science\Notes\Sets&Tuples.ipynb Cell 61 in <cell line: 1>()
----> <a href='vscode-notebook-cell:/c%3A/Users/PKVidyarthi/Desktop/Data%20Science/Notes/Sets%26Tuples.ipynb#ch0000062?line=0'>1</a> print(t1[5])

IndexError: tuple index out of range
```

```
print(t1)
print(t1[3][0])
print(t1[3][1])
```

```
(4, 2, 3, [5, 6])
5
6
```

```
t1[3][0] = 400
t1[3][1] = 600
print(t1)
```

```
(4, 2, 3, [400, 600])
```

```
t1[0] = 200
# TypeError: 'tuple' object does not support item assignment.
# Can not be change or assign values in tuples.
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
c:\Users\PKVidyarthi\Desktop\Data Science\Notes\Sets&Tuples.ipynb Cell 64 in <cell line: 1>()
----> <a href='vscode-notebook-cell:/c%3A/Users/PKVidyarthi/Desktop/Data%20Science/Notes/Sets%26Tuples.ipynb#ch0000065?line=0'>1</a> t1[0] = 200

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

```
# See the difference:
print(t1)
print(type(t1))
print(t1[3])
```

```
c = t1[3]
print(c)
print(type(c))
```

```
(4, 2, 3, [400, 600])
<class 'tuple'>
[400, 600]
[400, 600]
<class 'list'>
```

```
# Converting a tuple into a list.
t1 = (4, 2, 3, [400, 600])
print(t1)                           # Output => (4, 2, 3, [400, 600])
print(type(t1))                     # Output => <class 'tuple'>
lst = list(t1)
print(lst)                          # Output => [4, 2, 3, [400, 600]]
print(type(lst))                    # Output => <class 'list'>
```

```
(4, 2, 3, [400, 600])
<class 'tuple'>
[4, 2, 3, [400, 600]]
<class 'list'>
```

```
# Now we can change, update or assign the value of lst[0]

lst[0] = 200
print(lst)                          # Output => [200, 2, 3, [400, 600]]

print(lst[3])                       # Output => [400, 600]
print(lst[3][1])                    # Output => 600

lst[3][1] = 100
print(lst)                          # Output => [200, 2, 3, [400, 100]]

lst[3] = 1000                       # It will change [400, 100] to 1000.
print(lst)                          # Output => [200, 2, 3, 1000]
```

```
[200, 2, 3, [400, 100]]
[400, 100]
100
[200, 2, 3, [400, 100]]
[200, 2, 3, 1000]
```