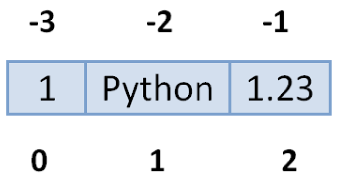
Tuple

- ☐ A tuple is created by placing all the elements inside a parenthesis ( ), separated by commas or just by separating items by comma.
- ☐ The elements of a tuple can be different data-type.

```python
tup = (1, "Python", 1.23)
```



tup[0]= 1= tup[-3]

tup[1]= "Python"= tup[-2]

tup[2]= 1.23= tup[-1]

Concatenation and Repetition of Tuples

Questions:
Do tuple support append, insert and extend methods?

- ☐ + operator is used for concatenation of tuples.
- ☐ * operator is used for the repetition of tuples.

```python
tup1 = (1, "Python", 1.23)
tup2 = ('a', 'b', 3)
tup3 = tup1 + tup2
tup4 = tup3*2
print(tup3)
(1, 'Python', 1.23, 'a', 'b', 3)
print(tup4)
(1, 'Python', 1.23, 'a', 'b', 3, 1, 'Python', 1.23, 'a',
    'b', 3)
```

Question

Questions:
Do all the methods we have seen for list, will work for tuple? Figure out which will work and why?

sorted() Function

- ☐ sorted() function sorts any sequence (tuple, list, string etc.) and returns a list with the elements in sorted manner, without modifying the original sequence.

Syntax

sorted(sequence, key = ..., reverse= ...)

sorted() Function

```
>>>NumList = [5, 15, 0, 20, 10]
>>>print(sorted(NumList))
[0, 5, 10, 15, 20]
>>>print(NumList)
[5, 15, 0, 20, 10]
>>>Courses = 'IC140', 'IC160', 'IC110', 'IC152','HS106',
    'IC152'
>>>print(Courses)
('IC140', 'IC160', 'IC110', 'IC152', 'HS106', 'IC152')
>>>NewCourses = sorted(Courses, reverse=True)
>>>print(NewCourses)
['IC160', 'IC152', 'IC152', 'IC140', 'IC110', 'HS106']
>>>Tup = [(1,5), (5, 2), (2, 3), (1, 3),  (2,1)]
>>>New_Tup = sorted(Tup, key = lambda x: x[1])
>>>print(New_Tup)
[(2, 1), (5, 2), (2, 3), (1, 3), (1, 5)]
```

reversed() Function

- ☐ reversed() function returns the reversed iterator of the given sequence.

Syntax

reversed(sequence)

```
>>>NumList = [5, 15, 0, 20, 10]
>>>print(reversed(NumList))
<list_reverseiterator object at 0x00000297E2CD30B8>
>>>print(tuple(reversed(NumList)))
(10, 20, 0, 15, 5)
>>>Courses = 'IC140', 'IC160', 'IC110', 'IC152','HS106',
    'IC152'
>>>print(reversed(Courses))
<reversed object at 0x00000297E2CD30B8>
>>>print(list(reversed(Courses)))
['IC152', 'HS106', 'IC152', 'IC110', 'IC160', 'IC140']
```

Difference Between Tuple and List

- ☐ The main difference between list and a tuple is the fact that lists are mutable (i.e. elements of list can be modified after its creation) whereas tuples are immutable(i.e., elements of tuple cannot be changed or modified after its creation).
- ☐ List has more functionality than tuple.
- ☐ Lists has variable length, tuple has fixed length.
- ☐ Tuples are more memory efficient as compare to lists
- ☐ Tuple operations have smaller size than that of list, which makes it a bit faster.

Set

- ☐ A set() is an unordered collection of objects.
- ☐ Every element of the set is unique (no duplicates) and must be immutable (i.e. cannot be changed).
- ☐ The set itself is mutable, we can add or remove items from it.
- ☐ A set is created by placing all the elements inside curly braces {}, separated by comma.
- ☐ The elements of a set can be different data-type.
- ☐ Set does not support indexing or slicing.

Set

- ☐ A set() is an unordered collection of objects.
- ☐ Every element of the set is unique (no duplicates) and must be immutable (i.e. cannot be changed).
- ☐ The set itself is mutable, we can add or remove items from it.
- ☐ A set is created by placing all the elements inside curly braces {}, separated by comma.
- ☐ The elements of a set can be different data-type.
- ☐ Set does not support indexing or slicing.

```
>>>emptySet = set()
>>>print("Empty set :", emptySet)
Empty set: set()
>>>print(type(emptySet))
<class 'set'>
>>>mixedSet = {1, 'a', 1.23, 1, 'python'}
>>>print("Mixed set:", mixedSet)
Mixed set: {1, 'python', 1.23, 'a'}
```

Set Methods

Method	Explanation
add()	**Add an element to the set**
update()	**Adds elements from a sequence (passed as an argument) to the set .**
discard()	**Removes an element from the set if it is a member**
remove()	**Removes an element from the set**
pop()	**Removes and returns an arbitrary element of set**

Set Methods

union()	**Returns the union of sets**
intersection()	**Returns the intersection of sets**
difference()	**Returns the difference of two or more sets**
symmetric_difference()	**Returns the symmetric difference of two sets**

add() Method

- ☐ add() method adds a given element to set (if element is not in the set).

Syntax

set.add(element)

```
>>>NumSet = {5, 15, 0, 20, 10}
>>>NumSet.add(25)
>>>print(NumSet)
{0, 5, 10, 15, 20, 25}
>>>Courses = {'IC140', 'IC160', 'IC110', 'IC152','HS106',
    'IC152'}
>>>Courses.add('IC110')
>>>print(Courses)
{'HS106', 'IC152', 'IC140', 'IC110', 'IC160'}
```

update() Method

- ☐ update() method adds elements from a sequence (passed as an argument) to the set (calling the update() method).

Syntax

set.update(sequence)

```
>>>NumSet = {5, 15, 0, 20, 10}
>>>string = "hello"
>>>NumSet.update(string)
>>>print(NumSet)
{0, 5, 'e', 10, 'l', 15, 20, 'h', 'o'}
>>>List = [1, 'a', (5,'a')]
>>>NumSet.update(List)
>>>print(NumSet)
{0, 1, 5, 'e', 10, 'l', 15, 20, 'h', 'a', (5, 'a'), 'o'}
>>>Courses = {'IC140', 'IC160', 'IC110', 'IC152','HS106',
    'IC152'}
>>>Courses.update(NumSet)
>>>print(Courses)
{0, 1, 'IC140', 5, 'IC110', 'e', 10, 'l', 15, 'IC152',
    'IC160', 20, 'h', 'a', (5, 'a'), 'o', 'HS106'}
```

discard() Method

- ☐ discard() method removes an element from the set, if present.

Syntax

set.discard(element)

```
>>>NumSet = {5, 15, 0, 20, 10}
>>>NumSet.discard(0)
>>>print(NumSet)
{5, 10, 15, 20}
>>>NumSet.discard(25)
>>>print(NumSet)
{5, 10, 15, 20}
```

remove() Method

- ☐ remove() method removes the given element from the set if present, otherwise throughs an error.

Syntax

set.remove(element)

```
>>>NumSet = {5, 15, 0, 20, 10}
>>>NumSet.remove(0)
>>>print(NumSet)
{5, 10, 15, 20}
>>>NumSet.remove(25)
>>>print(NumSet)
KeyError: 25
```

pop() Method

- ☐ pop() method removes an arbitrary element from the set and return it.

Syntax

set.pop()

```
>>>NumSet = {5, 15, 0, 20, 10}
>>>print(NumSet.pop())
0
>>>print(NumSet)
{5, 10, 15, 20}
>>>Courses = {'IC140', 'IC160', 'IC110', 'IC152','HS106',
    'IC152'}
>>>print(Courses.pop())
IC140
>>>print(Courses)
{'IC110', 'IC152', 'IC160', 'HS106'}
```

union() Method

- union() method returns the union of the sets.

Syntax

SetA.union(SetB)

```
>>>NumSet1 = {5, 15, 0, 20, 10}
>>>NumSet2 = {10, 20, 30, 40, 50}
>>>NumSet3 = NumSet1.union(NumSet2)
>>>print(NumSet3)
{0, 5, 40, 10, 15, 50, 20, 30}
>>>Courses = {'IC140', 'IC160', 'IC110', 'IC152','HS106',
    'IC152'}
>>>Set = Courses.union(NumSet1, NumSet2)
>>>print(Set)
{0, 'IC140', 5, 'IC110', 40, 10, 'HS106', 15, 50, 'IC152',
    20, 'IC160', 30}
```

intersection() Method

- intersection() method returns the intersection of sets.

Syntax

SetA.intersection(SetB)

```
>>>NumSet1 = {5, 15, 0, 20, 10}
>>>NumSet2 = {10, 20, 30, 40, 50}
>>>NumSet3 = NumSet1.intersection(NumSet2)
>>>print(NumSet3)
{10, 20}
>>>Courses = {'IC140', 'IC160', 'IC110', 'IC152','HS106',
    'IC152'}
>>>Set = Courses.intersection(NumSet1, NumSet2)
>>>print(Set)
set()
```

difference() Method

- difference() method returns the difference of the sets.

Syntax

SetA.difference(SetB)

```
>>>NumSet1 = {5, 15, 0, 20, 10}
>>>NumSet2 = {10, 20, 30, 40, 50}
>>>NumSet3 = NumSet1.difference(NumSet2)
>>>print(NumSet3)
{0, 5, 15}
>>>Set4 = NumSet2 - NumSet1
>>>print(Set4)
{40, 50, 30}
```

symmetric_difference() Method

- symmetric_difference() method returns the symmetric difference of two sets. .

Syntax

SetA.symmetric_difference(SetB)

```
>>>NumSet1 = {5, 15, 0, 20, 10}
>>>NumSet2 = {10, 20, 30, 40, 50}
>>>NumSet3 = NumSet1.symmetric_difference(NumSet2)
>>>print(NumSet3)
{0, 50, 5, 40, 30, 15}
>>>NumSet4 = NumSet2.symmetric_difference(NumSet1)
>>>print(NumSet4)
{0, 50, 5, 40, 30, 15}
>>>Set = {5, 'a', 10}
>>>print(NumSet1.symmetric_difference(Set))
{0, 20, 'a', 15}
```