









Cont. Data Structures-II

Session Objectives

-  Understand what tuples are.
-  Understand common methods and operations associated with tuples.
-  Understand what sets are.
-  Understand common methods and operations associated with sets.
-  Understand the comparison between lists, tuples and sets.
-  Understand what dictionaries are.
-  Understand common methods and operations associated with dictionaries.
-  Understand the comparison between lists, tuples, sets and dictionaries.

```
# Understanding Common Methods and operation associated with Sets
# Accessing an item [X No Indexing]
# membership operators # [Boolean Return]
car_set = {'Taigun', 'Creta', 'Thar', 'Kylaq', 'Camry', 'Virtus', 'Nexon', 'Rolls Royce', 'Verna'}
print('Slavia' in car_set) # False
print('Thar' in car_set) # True
print('Kylaq' not in car_set) # False
print('ScorpioN' not in car_set) # True
```

```
False
True
False
True
```

```
# Length of a set -> len() to count the number of elements.
car_set = {'Taigun', 'Creta', 'Thar', 'Kylaq', 'Camry', 'Virtus', 'Nexon', 'Rolls Royce', 'Verna'}
print(len(car_set))
```

```
9
```

```
# min() , max() and the sum()
quick_set = {0,1,2,3,4,5,11,22,33,44,55,False,True} # False[0] , True[1] {Duplicate}
print(min(quick_set)) # 0
print(max(quick_set)) # 55
print(sum(quick_set)) # sum
```

```
0
55
180
```

```
# min() , max() and the sum()
quick_set = {False,True,2,3,4,5,11,22,33,44,55,0,1} # False[0] , True[1] {Duplicate}
print(quick_set)
print(min(quick_set)) # False
print(max(quick_set)) # 55
print(sum(quick_set)) # sum
```

```
{False, True, 2, 3, 4, 5, 33, 11, 44, 22, 55}
False
55
180
```

```
# Adding Items to a set()
# .add() -> Adds a single elements
quick_set = {False,True,2,3,4,5,11,22,33,44,55,0,1}
quick_set.add(77)
print(quick_set)
```

```
{False, True, 2, 3, 4, 5, 33, 11, 44, 77, 22, 55}
```

```
quick_set.add(99)
print(quick_set)
```

```
{False, True, 2, 3, 4, 5, 33, 99, 11, 44, 77, 22, 55}
```

```
# .update(<iterables>) -> Adds a multiple elements -> iterables as argument
quick_set.update(['a','b','c'])
print(quick_set)
```

```
{False, True, 2, 3, 4, 5, 33, 99, 11, 44, 77, 22, 55, 'b', 'c', 'a'}
```

```
quick_set.update('Coding')
print(quick_set)
```

```
{False, True, 2, 3, 4, 5, 'C', 11, 'd', 22, 'n', 33, 44, 'o', 55, 'b', 'c', 77, 99, 'i', 'g', 'a'}
```

```
quick_set.update(('Coding',))
print(quick_set)
```

```
{False, True, 2, 3, 4, 5, 'C', 11, 'd', 22, 'n', 33, 'Coding', 44, 'o', 55, 'b', 'c', 77, 99, 'i', 'g', 'a'}
```

```
# Removing items from a set()
# remove() -> throws an errors if item doesn't exist
# discard() -> No error if item doesn't exist
quick_set.remove(False)
print(quick_set)
```

```
{True, 2, 3, 4, 5, 'C', 11, 'd', 22, 'n', 33, 'Coding', 44, 'o', 55, 'b', 'c', 77, 99, 'i', 'g', 'a'}
```

```
quick_set.remove(True)
print(quick_set)
```

```
{2, 3, 4, 5, 'C', 11, 'd', 22, 'n', 33, 'Coding', 44, 'o', 55, 'b', 'c', 77, 99, 'i', 'g', 'a'}
```

```
quick_set.remove('True') # doesn't exist , it throws error
print(quick_set)
```

```
quick_set.discard(['a','b','c']) # doesn't exist still won't throw an error
print(quick_set)
```

```
{2, 3, 4, 5, 'C', 11, 'd', 22, 'n', 33, 'Coding', 44, 'o', 55, 77, 99, 'i', 'g'}
```

```

quick_set.discard('a')
quick_set.discard('b')
quick_set.discard('c')
print(quick_set)

{2, 3, 4, 5, 'C', 11, 'd', 22, 'n', 33, 'Coding', 44, 'o', 55, 77, 99, 'i', 'g'}

quick_set.add(('x','y','z'))
print(quick_set)

{2, 3, 4, 5, 'C', ('x', 'y', 'z'), 11, 'd', 22, 'n', 33, 'Coding', 44, 'o', 55, 77, 99, 'i', 'g'}

quick_set.discard(('x','y','z')) # remove the element as it exist
print(quick_set)

{2, 3, 4, 5, 'C', 11, 'd', 22, 'n', 33, 'Coding', 44, 'o', 55, 77, 99, 'i', 'g'}

```

```

# pop() -> Removes and returns an arbitrary elements. Since sets are unordered,
# you don't which item will be removed
popped_item = quick_set.pop()
print(popped_item)
print(quick_set)

2
{3, 4, 5, 'C', 11, 'd', 22, 'n', 33, 'Coding', 44, 'o', 55, 77, 99, 'i', 'g'}

popped_item = quick_set.pop()
print(popped_item)
print(quick_set)

3
{4, 5, 'C', 11, 'd', 22, 'n', 33, 'Coding', 44, 'o', 55, 77, 99, 'i', 'g'}

```

```

popped_item = quick_set.pop()
print(popped_item)
print(quick_set)

4
{5, 'C', 11, 'd', 22, 'n', 33, 'Coding', 44, 'o', 55, 77, 99, 'i', 'g'}

popped_item = quick_set.pop()
print(popped_item)
print(quick_set)

5
{'C', 11, 'd', 22, 'n', 33, 'Coding', 44, 'o', 55, 77, 99, 'i', 'g'}

popped_item = quick_set.pop()
print(popped_item)
print(quick_set)

C
{11, 'd', 22, 'n', 33, 'Coding', 44, 'o', 55, 77, 99, 'i', 'g'}

```



```
popped_item = quick_set.pop()
print(popped_item)
print(quick_set)

11
{'d', 22, 'n', 33, 'Coding', 44, 'o', 55, 77, 99, 'i', 'g'}

popped_item = quick_set.pop()
print(popped_item)
print(quick_set)

d
{22, 'n', 33, 'Coding', 44, 'o', 55, 77, 99, 'i', 'g'}

popped_item = quick_set.pop()
print(popped_item)
print(quick_set)

22
{'n', 33, 'Coding', 44, 'o', 55, 77, 99, 'i', 'g'}
```

```
popped_item = quick_set.pop()
print(popped_item)
print(quick_set)

n
{33, 'Coding', 44, 'o', 55, 77, 99, 'i', 'g'}

popped_item = quick_set.pop()
print(popped_item)
print(quick_set)

33
{'Coding', 44, 'o', 55, 77, 99, 'i', 'g'}

popped_item = quick_set.pop()
print(popped_item)
print(quick_set)

Coding
{44, 'o', 55, 77, 99, 'i', 'g'}
```

```
day_set = {'Mon', 'Wed', 'Thurs', 'Sat', 'Tues', 'Fri', 'Sun'}
pop_item = day_set.pop()
print(pop_item)

Wed

print(day_set)

{'Tues', 'Thurs', 'Sun', 'Mon', 'Sat', 'Fri'}

pop_item = day_set.pop()
print(pop_item)
print(day_set)

Tues
{'Thurs', 'Sun', 'Mon', 'Sat', 'Fri'}

day_set.add('Jan')
print(day_set)

{'Jan', 'Thurs', 'Sun', 'Mon', 'Sat', 'Fri'}
```

```

day_set.add('Feb')
print(day_set)

{'Thurs', 'Sun', 'Feb', 'Jan', 'Mon', 'Sat', 'Fri'}

# clear
day_set.clear()
print(day_set) # {}

set()

# del -> We can delete the entire set object
del day_set
print(day_set) # NameError: name 'day_set' is not defined

```

Joining Sets

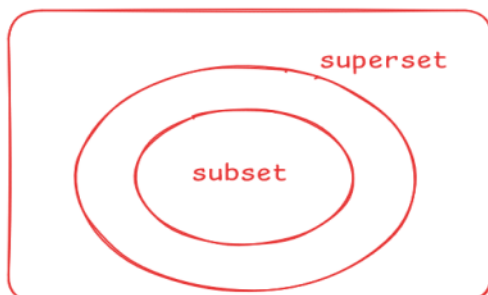
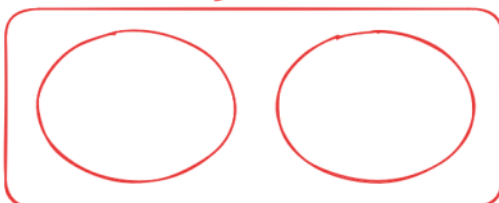


```

# difference_update()
# shortcut : a -= b|c
setA = {'a','b','c','d','e'}
setB = {'p','q','r','s','a','b','c'}
setC = {'x','y','r','s','a','c','z'}
# setB | setC => {'a','b','c','p','q','r','s','x','y','z'}
# setA - (setB | setC) => {'d','e'}
# result = setA.difference_update(setB,setC)
result = setA = setA - (setB | setC)
setA.difference_update(setB,setC)
print(result)
print(setA)

```

disjoint Set



```
# Union (/ or .union())
setA = {1,2,3,4,5,5,6,6,8,8,8,8} # {1,2,3,4,5,6,8}
setB = {3,5,7,7,7,7,7,9,9} # {3,5,7,9}
union_set = setA | setB # (AUB)
print(union_set) # {1...9}

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

result = setA.union(setB)
print(result)

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

```
# Advance union concepts
setA = {'a','b','c','d','e'}
setB = {'p','q','r','s','a','b','c'}
setC = {'x','y','r','s','a','c','z'}
print(setA | setB | setC)
print(setA.union(setB).union(setC))

{'q', 'y', 'p', 'e', 'd', 'z', 'r', 'x', 'b', 's', 'c', 'a'}
{'q', 'y', 'p', 'e', 'd', 'z', 'r', 'x', 'b', 's', 'c', 'a'}
```

```
# Intersection (& or intersection())
setA = {'a','b','c','d','e'}
setB = {'p','q','r','s','a','b','c'}
setC = {'x','y','r','s','a','c','z'}
print(setA & setB) # {a,b,c}
print(setA.intersection(setB).intersection(setC)) # {'a','c'}
print(setA & setB & setC) # {'a','c'}

{'b', 'c', 'a'}
{'c', 'a'}
{'c', 'a'}
```

```
# difference (- or .difference()) # Elements in A but not in B
setA = {'a','b','c','d','e'}
setB = {'p','q','r','s','a','b','c'}
setC = {'x','y','r','s','a','c','z'}
print(setA - setB) # {'d','e'}
print(setA.difference(setB)) # {'d','e'}

{'d', 'e'}
{'d', 'e'}

print(setB - setA) # {'p','q','r','s'}
print(setB.difference(setA)) # {'p','q','r','s'}

{'p', 's', 'r', 'q'}
{'p', 's', 'r', 'q'}
```

```

setA = {'a','b','c','d','e'}
setB = {'p','q','r','s','a','b','c'}
setC = {'x','y','r','s','a','c','z'}
print(setB - setA - setC) # {'p','q'}
print(setB.difference(setA).difference(setC))

```

```

{'p', 'q'}
{'p', 'q'}

```

```

# Symmetric Difference (^ or .symmetric_difference())
# Elements in either SetA or SetB , but not in both
setA = {'a','b','c','d','e'}
setB = {'p','q','r','s','a','b','c'}
setC = {'x','y','r','s','a','c','z'}
print(setA ^ setB) # {'d','e','p','q','r','s'}
print(setA.symmetric_difference(setB)) # {'d','e','p','q','r','s'}

```

```

{'q', 'p', 'e', 'd', 'r', 's'}
{'q', 'p', 'e', 'd', 'r', 's'}

```

```

setA = {'a','b','c','d','e'}
setB = {'p','q','r','s','a','b','c'}
setC = {'x','y','r','s','a','c','z'}
print(setA ^ setB ^ setC) # {'d','e','p','q','r','s'} ^ {'x','y','r','s','a','c','z'}
# {'d','e','p','q','x','y','a','c','z'}
print(setA.symmetric_difference(setB).symmetric_difference(setC)) # {'d','e','p','q','x','y','a','c','z'}

```

```

{'q', 'd', 'x', 'y', 'e', 'z', 'p', 'c', 'a'}
{'q', 'd', 'x', 'y', 'e', 'z', 'p', 'c', 'a'}

```

```

# difference_update()
# shortcut : a -= b/c
setA = {'a','b','c','d','e'}
setB = {'p','q','r','s','a','b','c'}
setC = {'x','y','r','s','a','c','z'}
# setB | setC => {'a','b','c','p','q','r','s','x','y','z'}
# setA -= (setB | setC) => {'d','e'}
# setA.difference_update(setB,setC)
setA -= setB | setC
print(setA)

```

```

{'d', 'e'}

```

```

setA.difference_update(setB,setC)
print(setA)

```

```

{'d', 'e'}

```

difference_update() → changes original set, returns None.


```
# intersection_update()
# shortcut : a &= b -> a = a&b
setA = {'a','b','c','d','e'}
setB = {'p','q','r','s','a','b','c'}
setC = {'x','y','r','s','a','c','z'}
setA.intersection_update(setB)
print(setA)

{'b', 'c', 'a'}
```

```
setA = setA & setB
print(setA)

{'b', 'c', 'a'}
```

```
setA = {'a','b','c','d','e'}
setB = {'p','q','r','s','a','b','c'}
setC = {'x','y','r','s','a','c','z'}
# setA.intersection_update(setB,setC) {'c', 'a'}
setA = setA & (setB & setC)
# {'a','b','c','d','e'} & ({'a','c'})
print(setA)

{'c', 'a'}
```

```
# isdisjoint
setA = {'a','b','c','d','e'}
setB = {'p','q','r','s','a','b','c'}
setC = {'x','y','r','s','a','c','z'}
setD = {'k','m'}
print(setA.isdisjoint(setB)) # False
print(setA.isdisjoint(setD)) # True

False
True
```

```
# issubset() & issuperset()
set1 = {1,2,3,4,5}
set2 = {1,2,3,4,5,6,7,8,9}
print(set1.issubset(set2)) # True as set1 is subset of set2
print(set2.issuperset(set1)) # True as set2 is superset of set1

True
True
```

```
# symmetric_difference_update()
# shortcut a ^= b
setA = {'a','b','c','d','e'}
setB = {'p','q','r','s','a','b','c'}
setC = {'x','y','r','s','a','c','z'}
setA.symmetric_difference_update(setB)
print(setA)

{'d', 'r', 'q', 's', 'p', 'e'}
```



```

setA = {'a','b','c','d','e'}
setB = {'p','q','r','s','a','b','c'}
setC = {'x','y','r','s','a','c','z'}
setA = setA ^ setB
print(setA)

```

```

{'q', 'p', 'e', 'd', 'r', 's'}

```

```

# (setB,setC) -> {'p','q','r','s','a','b','c'} ^ {'x','y','r','s','a','c','z'}
# -> {'p','q','b','x','y','z'}
# setA ^ {'p','q','b','x','y','z'} -> {'a','c','d','e','p','q','x','y','z'}
setA = {'a','b','c','d','e'}
setB = {'p','q','r','s','a','b','c'}
setC = {'x','y','r','s','a','c','z'}
setA = setA ^ (setB ^ setC)
print(setA)

```

```

{'d', 'x', 'y', 'p', 'e', 'z', 'q', 'c', 'a'}

```

```

# update() |
# shortcut a |= b/c
setA = {'a','b','c','d','e'}
setB = {'p','q','r','s','a','b','c'}
setC = {'x','y','r','s','a','c','z'}
setA.update(setB,setC)
print(setA)

```

```

{'q', 'y', 'p', 'e', 'd', 'z', 'r', 'x', 'b', 's', 'c', 'a'}

```

```

setA = {'a','b','c','d','e'}
setB = {'p','q','r','s','a','b','c'}
setC = {'x','y','r','s','a','c','z'}
setA = setA | setB | setC
print(setA)

```

```

{'q', 'y', 'p', 'e', 'd', 'z', 'r', 'x', 'b', 's', 'c', 'a'}

```