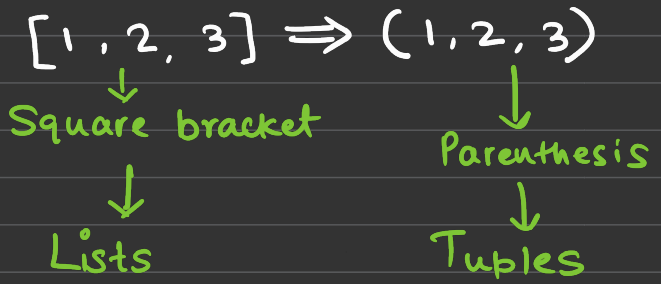

DECODE
AI

Tuples

→ Examples



↳ my_tuple = (1, 2, 3) ← homogenous data

my_tuple2 = (10, "Python", True) ← heterogenous data

→ Tuples are similar to lists but they are Immutable.

Properties of Tuples

- Ordered
- Immutable
  - Can't be changed after creation. No item assignment, addition or removal is allowed.
- Allow duplicates
- Can contain heterogenous data
- Support indexing and slicing
- Faster than lists

- While you are passing around an object and if you need to make sure that it doesnot get changed then tuple become your solution.

Sets

→ Examples

① `my_set = {1, 2, 3, 4}` ← homogenous

② `set_from_list = set([1, 2, 2, 3, 4])`

③
```
my_set = set()
my_set.add(10)
my_set.add(15)
```

④ `my_set = {10, 6.5, "India"}` ← heterogenous

~~⑤~~ `my_set = {10, 6.5, [2, 3]}`
↑ List: unhashable type

~~⑥~~ `my_set = {10, 6.5, {2, 3}}`
↑ Set: unhashable type

Properties of a set

→ not ordered

→ Duplicates not allowed

→ Indexing not supported

→ Mutable

→ Elements must be hashable

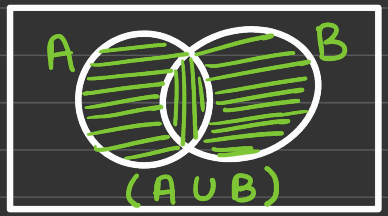
☑ Common use Cases

→ Removing duplicates from list

→ Membership test (in , not in)

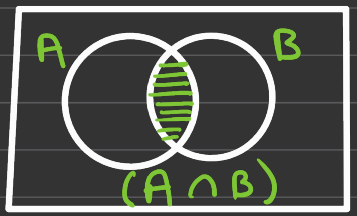
→ Mathematical Set operations (∪ , ∩ , −)

☑ operations on set



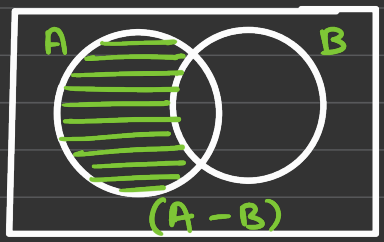
A.union(B)

(A | B)



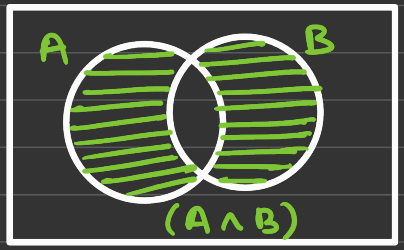
A.intersection(B)

(A & B)



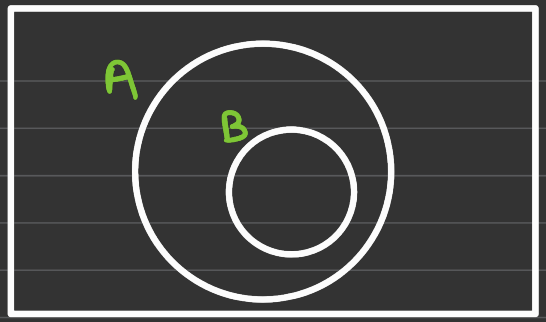
A.difference(B)

(A-B)



A.symmetric_difference(B)

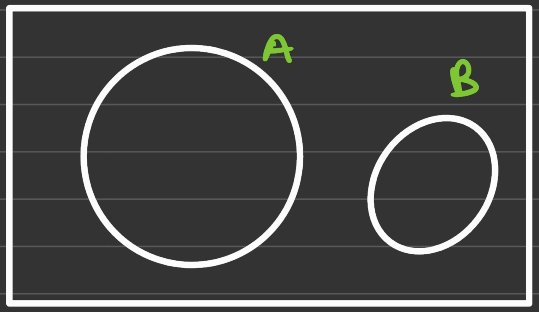
(A ^ B)



→ B is Subset of A

→ A is superset of B

→ A and B is disjoint



Dictionary

```
my_dict = { 'key1' : 'value1',
            'key2' : 'value2',
            'key3' : 'value3'
          }
```

→ Keys = ['key1', 'key2', 'key3']

→ values = ['value1', 'value2', 'value3']

→ Items = [('key1', 'value1'), ('key2', 'value2'), ('key3', 'value3')]

→ Examples

① `marks = {'Ramesh' : 60 , 'Mahesh' : 80 , 'Bupesh' : 60}`

② `grades = {'Ramesh' : 'B' , 'Mahesh' : 'A' , 'Bupesh' : 'B'}`

③ `Roll_to_marks = { 2 : 60 , 6 : 80 , 9 : 20 }`

④
```
info = { "name" : "Ramesh"
         "Score" : 60
         "Skills" : ["Python", "Django", "SQL"]
       }
```

Properties of a Dict

Property	Supported	Notes
Key-Value Storage	✅	Fast and flexible
Mutable	✅	Can modify after creation
Insertion Ordered	✅	From Python 3.7+
Duplicate Keys	❌	Last occurrence is stored
Indexed by Position	❌	Access by key only
Nested Structures	✅	Values can be lists, dicts, etc.
Iterable	✅	Can iterate over keys, values, or items
Hash-Based Lookup	✅	Fast retrieval using keys (O(1) average case)

NOTE

① In a dict, **values** can be of any types, hashable or not.

② But, a **keys** must be immutable and hashable

- int, float, str, tuple
- ~~list~~, ~~dict~~, ~~set~~

Demo Time