

The background is a solid blue color with a subtle pattern of overlapping, semi-transparent geometric shapes, primarily triangles and polygons, in various shades of blue. In the top-left and bottom-right corners, there are white line-art diagrams representing networks or graphs, with small white dots at the vertices and thin white lines connecting them.

Dictionaryes & Sets

Dictionaries

Defining Dictionaries

Dic = { <key1> : <value1> , <key2> : <value2> , <key3> : <value3> }

```
dictionary = dict([  
    ('name', 'john'),  
    ('year', 1995),  
    ('score', [3.5, 80])  
])
```

```
dictionary = dict(  
    name = 'john',  
    year = 1995,  
    score = [3.5, 80])
```

Accessing Elements from Dictionary

List

Index	
0	'John'
1	1992
2	3.14
3	True
4	[steve, David]

Dictionary

Keys	Values
'name'	'John'
'year'	1992
'score'	(3.14, 80)
'married'	True
'child'	[steve, David]

```
>>> Dictionary['name']  
'John'
```

Accessing Elements, Keys & Values

```
dic = {'a': [0, 1], 1: 'b', True: 'c', (1, ('x', False)): 'd'}
```

`name_of_dictionary[key]`

Or

`name_of_dictionary.get(key)`



`dic['a']`

Or

`dic.get('a')`

`Name_of_dictionary.keys()`



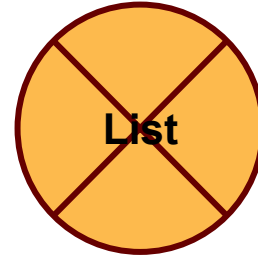
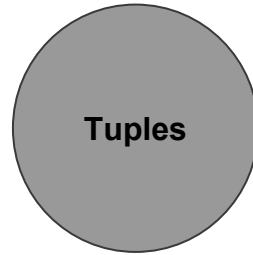
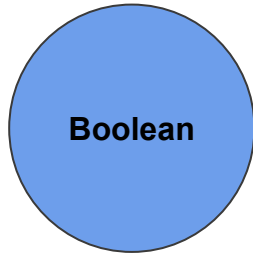
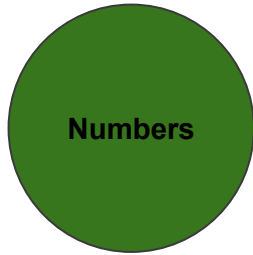
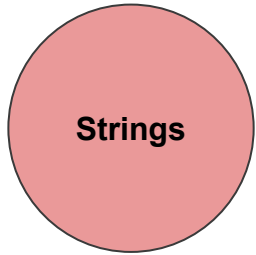
`dic.keys()`

`name_of_dictionary.values()`



`dic.values()`

Keys of dictionary are Immutable



```
dic = {'a': [0, 1], 1: 'b', True: 'c', (1, ('x', False)): 'd'}
```

Modifying values

```
info = {'name': 'John', 'age': 24}
```

```
info[ ]
```

Update existing value

```
info['score'] = 80
```

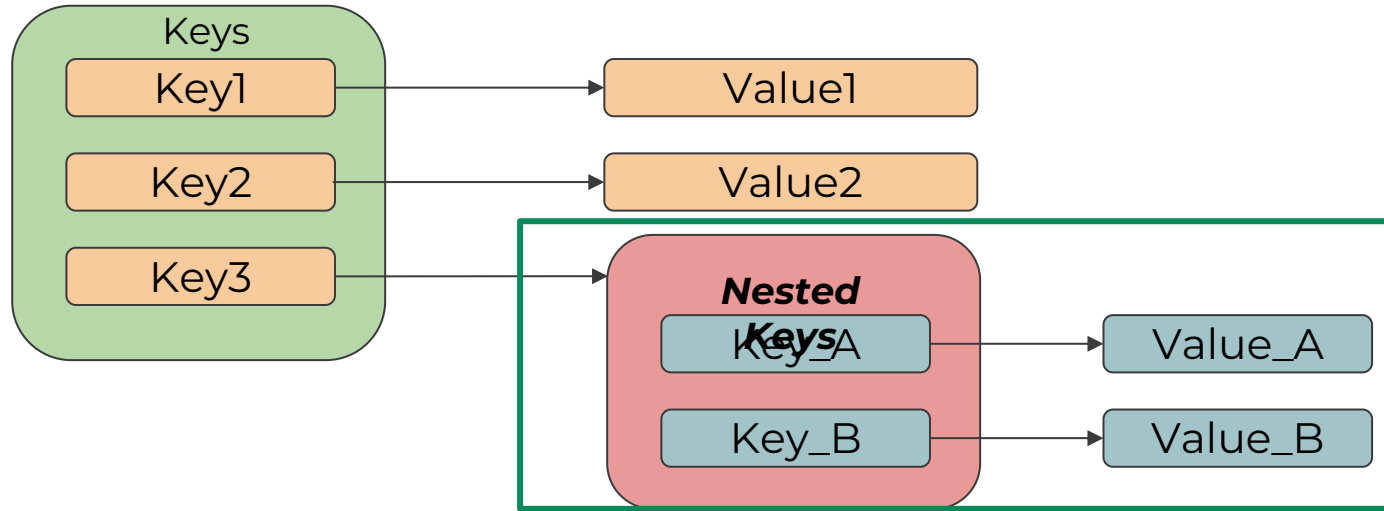
Add new key-value pair

```
info: {'name': 'John', 'age': 24, 'score': 80}
```

```
info.pop('age')
```

```
info.popitem()
```

Nested Dictionary



```
dic = {'a': 0, 1: 'b', True: {1: 'd', ('x', False): 'd'}}
```

```
dic[True][1]
```

or

```
dic.get(True).get(1)
```


Dictionaries: Code Demo

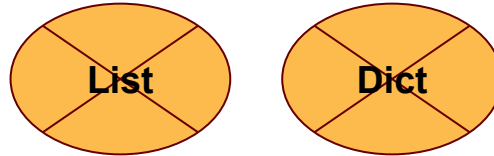
Sets

Python Sets

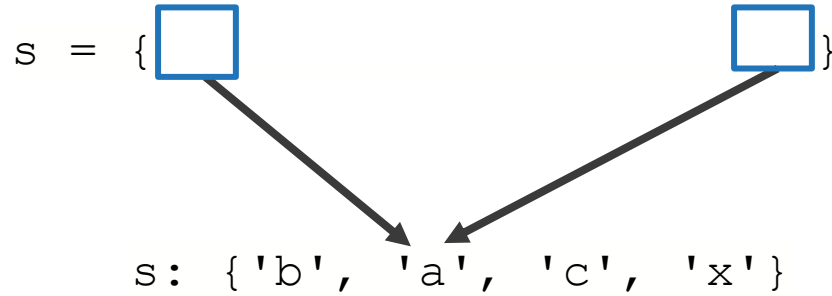
Collections of well-defined distinct items

Items are unordered.

Items are immutable.



Defining Sets



Modifying a set

`A = { "apple", "banana", "mango", "orange", "lemon" }`

`A.add()`

`A.remove()`

A

carrot

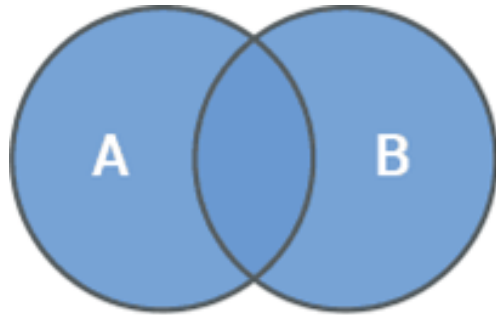
**apple, banana,
mango, orange,
lemon**

Some Sets Operations

Some Sets Operations: Union

$A = \{ \text{"apple"}, \text{"banana"}, \text{"mango"}, \text{"orange"}, \text{"lemon"} \}$

$B = \{ \text{"orange"}, \text{"pineapple"}, \text{"watermelon"}, \text{"grapes"}, \text{"lemon"} \}$



Union



$A \cup B$

OR

$A \mid B$

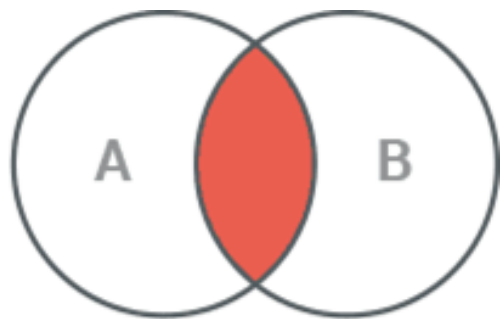


$\{ \text{'mango'}, \text{'banana'}, \text{'pineapple'}, \text{'apple'}, \text{'lemon'}, \text{'watermelon'}, \text{'grapes'}, \text{'orange'} \}$

Some Sets Operations: Intersection

$A = \{ \text{"apple"}, \text{"banana"}, \text{"mango"}, \text{"orange"}, \text{"lemon"} \}$

$B = \{ \text{"orange"}, \text{"pineapple"}, \text{"watermelon"}, \text{"grapes"}, \text{"lemon"} \}$



Intersection



`A.intersection(B)`

OR

`A & B`

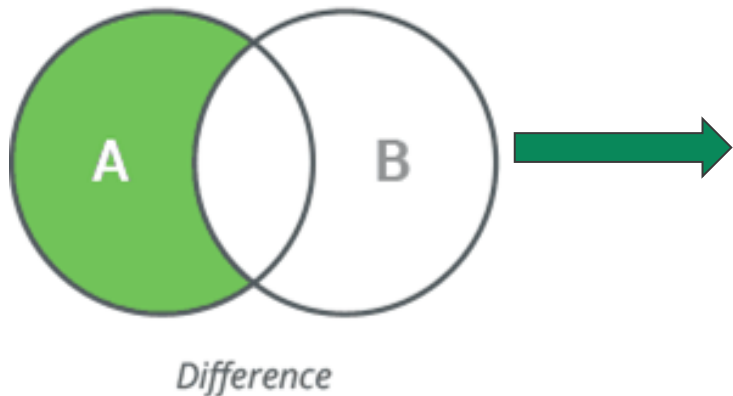


`{ 'orange', 'lemon' }`

Some Sets Operations: Difference

$A = \{ \text{"apple"}, \text{"banana"}, \text{"mango"}, \text{"orange"}, \text{"lemon"} \}$

$B = \{ \text{"orange"}, \text{"pineapple"}, \text{"watermelon"}, \text{"grapes"}, \text{"lemon"} \}$



$A.\text{difference}(B)$

OR

$A - B$

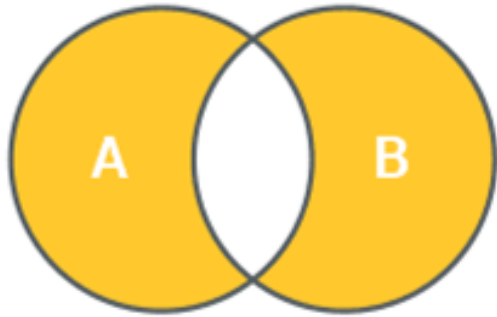


$\{ \text{'banana'}, \text{'apple'}, \text{'mango'} \}$

Some Sets Operations: Symmetric Difference

$A = \{ \text{"apple"}, \text{"banana"}, \text{"mango"}, \text{"orange"}, \text{"lemon"} \}$

$B = \{ \text{"orange"}, \text{"pineapple"}, \text{"watermelon"}, \text{"grapes"}, \text{"lemon"} \}$



Symmetric Difference



`A.symmetric_difference(B)`

OR

$A \triangle B$



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

Frozen sets

Frozen sets

→ `frozenset(iterable_object_name)`

Return Type: an equivalent **frozenset object**.

```
A = frozenset([1, 2, 3, 4])
```

~~A.add(8)~~
~~A.remove(1)~~

Modifying methods are not allowed

Non-modifying methods are allowed

**`copy()`, `difference()`, `intersection()`,
`isdisjoint()`, `issubset()`, `issuperset()`,
`symmetric_difference()` and `union()`**

Sets: Code Demo