# Python Set

# Python Sets

- A set is an unordered collection of items. Every element is unique and must be immutable.
- However, the set itself is mutable. We can add or remove items from it.
- Sets can be used to perform mathematical set operations like union, intersection, symmetric difference etc.

# Creating a Set

- A set is created by placing all the elements inside curly braces {}, separated by comma or by using the built-in function set().
- The elements can be of different types (integer, float, tuple, string etc.).
- But a set cannot have a mutable element, like list, set or dictionary, as its element.

```
#creating a set
numberSet = {1,2,3,4,3,2}
print(numberSet)

#creating an empty set
emptySet = {}   #This creates a dictionary
print(type(emptySet))

emptySet = set()   #This creates a empty set
print(type(emptySet))
```

```
{1, 2, 3, 4}

<class 'dict'>

<class 'set'>
```

A set can contain elements of different type

```
# set of mixed datatypes
my_set = {1.0, "Hello", (1, 2, 3)}
print(my_set)
```

```
{1.0, 'Hello', (1, 2, 3)}
```

A set cannot contain lists.

```
set_with_lists = {[1,2,3]}
```

```
----------------------------
TypeError
```

We can convert a list to set using set function

```
set_with_lists = set([1,2,3])
print(type(set_with_lists))
print(set_with_lists)
```

```
<class 'set'>
{1, 2, 3}
```

## LIST

- append
- extend
- Insert

- Pop
- remove

## SET

- add
- Update
- ----------

- ----------
- remove
- discard

# Adding elements to set

- Sets are mutable. But since they are unordered, indexing have no meaning.
- We cannot access or change an element of set using indexing or slicing.
- We can add single element using the **add()** method and multiple elements using the **update()** method.
- The update() method can take tuples, lists, strings or other sets as its argument.
- In all cases, duplicates are avoided.

```
my_set = set()   #empty set
my_set.update([9 , 12])
my_set.update((3,5))
my_set.update("SIKANDER")

print(my_set)
```

```
{3, 5, 9, 'A', 12, 'N', 'E', 'S', 'D', 'R', 'I', 'K'}
```

```
my_set.update(("INDIA" , "BHARAT"))
print(my_set)
```

```
my_set.update(4,5)
```

```
----------------------
TypeError
```

# Remove elements from a set

- A particular item can be removed from set using methods, discard() and remove().
- using discard() if the item does not exist in the set, it remains unchanged.
- But remove() will raise an error in such condition.

```
print (my_set)
```

```
{3, 5, 9, 12, 'INDIA', 'BHARAT'}
```

```
my_set.discard(12)
```

```
print(my_set)
```

```
{3, 5, 9, 'INDIA', 'BHARAT'}
```

```
my_set.discard(15)
print(my_set)
```

```
{3, 5, 9, 'INDIA', 'BHARAT'}
```

```
print(my_set)
my_set.remove(15)
```

```
{3, 5, 9, 'INDIA', 'BHARAT'}

---------------------------------
KeyError
```

# Set Membership Test

- We can test if an item exists in a set or not, using the **in** operator.

```
# initialize my_set
my_set = set("apple")

# check if 'a' is present
# Output: True
print('a' in my_set)

# check if 'p' is present
# Output: False
print('p' not in my_set)
```

```
True
False
```

## Exercise 1. DIY

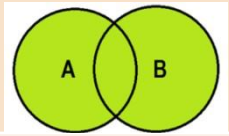
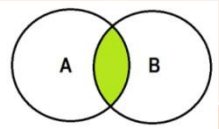
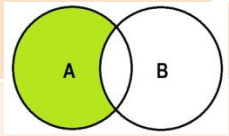
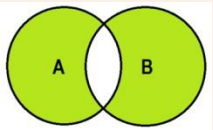
1. Given an list of elements, remove the duplicate elements?
2. Read a string and find the number of unique characters in it.

2 Read a string and find the number of unique characters in it.

```
data = "cranes varsity bangalore"

unique = set(data)

print("Number of unique characters =",len(unique))
print("List of unique characters = ",unique)

data = list(data)
for ele in unique:
    data.remove(ele)

print("Repeated Characters " , set(data))
```

# Python Set Operations

- Sets can be used to carry out mathematical set operations like union, intersection, difference and symmetric difference.
- We can do this with operators or methods.

| Method | Operator | |
|---|---|---|
| union | \| |   |
| intersection | & |   |
| difference | - |   |
| symmetric_difference | ^ |   |

```python
A = {1, 2, 3, 4, 5}
B = {4, 5, 6, 7, 8}

print('Union        = ' , A | B)
print('Intersection = ' , A & B)
print('Difference   = ' , A - B)
print('Symmetric Diff =' , A ^ B)
```

```
Union        =  {1, 2, 3, 4, 5, 6, 7, 8}
Intersection =  {4, 5}
Difference   =  {1, 2, 3}
Symmetric Diff = {1, 2, 3, 6, 7, 8}
```

```
studentList = {'danish', 'jaison', 'sangeetha', 'uma',
               'amrutha', 'lohita', 'prasad', 'ashwathi'}

placedStudList = ["uma","danish","amrutha"]

notPlacedStudList = set(studentList) - set(placedStudList)
print("Students yet to get job \n", notPlacedStudList)
```

```
batsmen = ["virat","rohit","dhawan","dhoni","pandya","jadeja"]

bowlers = ["bhuvenashwar","shami","pandya","jadeja","kuldeep"]

allrounders = set(batsmen) & set(bowlers)

print("Batsmen : " , batsmen)
print("Bowlers : " , bowlers)
print("All rounders : " , allrounders)
```

```
tcs = ["uma","danish","amrutha"]
infosys = ["lohita","danish","ashwathi"]
wipro = ["sangeetha","jaison","prasad","amrutha"]

placed = set(tcs) | set(infosys) | set(wipro)

print(placed)
print("Number of people placed = " , len(placed))
```

# Set Mutation Operations

- We have seen the applications of *union, intersection, difference* and *symmetric difference* operations, but these operations do not make any changes or mutations to the set.
- **We can use the following operations to create mutations to a set:**

| Method | Operator | |
|---|---|---|
| update | \|= | Update the set by adding elements from an iterable/another set. |
| **intersection_update** | &= | Update the set by keeping only the elements found in it and an iterable/another set. |
| Difference_update | -= | Update the set by removing elements found in an iterable/another set. |
| symmetric_difference_update | ^= | Update the set by only keeping the elements found in either set, but not in both. |

- Isdisjoint– This method will return True if two set have a null intersection
- Issubset – This method reports whether another set contains this set
- Issuperset – This method will report whether this set contains another set

```
numbers = {1,2,3,4,5,6,7,8,9,10}
odd = {1,3,5,7,9}
even = {2,4,6,8,10}

print(odd.isdisjoint(even))          #True
print(numbers.issuperset(odd))       #True
print(odd.issuperset(numbers))       #False
print(odd.issubset(numbers))         #True
```