Python Sets and its operations

- Sets are used to store multiple items in a single variable.
- Set is one of 4 built-in data types in Python used to store collections of data, the other 3 are List, Tuple, and Dictionary, all with different qualities and usage.
- A set is both *unordered* and *unindexed*.
- Unordered means that the items in a set do not have a defined order.
- Set items can appear in a different order every time you use them, and cannot be referred to by index or key.
- Set items are unchangeable, meaning that we cannot change the items after the set has been created.
- But you can add new items.
- Sets do not allow duplicate values.
- Sets are written with curly brackets.

Sets are unordered, so you cannot be sure in which order the items will appear.

Example:

```
print("\n*********************************************\n")
print("\n")

my_set = {"apple", "banana", "cherry"}

print(my_set)

print("\n")
```

```
(base) F:\CSE1001\Python-Programs>python prgm22-Sets.py

******************************************
{'cherry', 'banana', 'apple'}


(base) F:\CSE1001\Python-Programs>python prgm22-Sets.py

******************************************
{'apple', 'banana', 'cherry'}


(base) F:\CSE1001\Python-Programs>python prgm22-Sets.py

******************************************
{'apple', 'cherry', 'banana'}
```

Sets do not allow duplicate values

Example:

```
my_set_subjects = {"Physics", "Mathematics", "Chemistry", "Chemistry", "Physics"}
print(my_set_subjects)
```

```
{'Physics', 'Mathematics', 'Chemistry'}
```

Example:

```
my_set_subjects = {"Physics", "Mathematics", "Chemistry", "chemistry", "Physics"}
print(my_set_subjects)
```

```
{'Mathematics', 'chemistry', 'Chemistry', 'Physics'}
```

Example:

```
my_set_subjects = {"Physics", "Mathematics", "Chemistry", "chemistry", "physics"}
print(my_set_subjects)
```

```
{'physics', 'Chemistry', 'chemistry', 'Mathematics', 'Physics'}
```

Get the Length of a Set

To determine how many items a set has, use the len() method.

Example

```
my_set_subjects = {"Physics", "Mathematics", "Chemistry", "chemistry", "physics"}
print(my_set_subjects)
x= len(my_set_subjects)
print("Length of the set is :", x)
```

```
{'chemistry', 'physics', 'Physics', 'Mathematics', 'Chemistry'}
Length of the set is : 5
```

Set Items - Data Types: Set items can be of any data type:

Example

```
set1_fruits = {"apple", "banana", "cherry"}
print(set1_fruits)
print(type(set1_fruits))

print("\n")
set2_numbers = {1, 5, 7, 9, 3}
print(set2_numbers)
print(type(set2_numbers))

print("\n")
set3_boolean = {True, False, False}
print(set3_boolean)
print(type(set3_boolean))

print("\n")
set4_mix= {"Ajith", 18, True, "CSE_BSE", 9.95, "male"}
print(set4_mix)
print(type(set4_mix))
```

```
{'banana', 'apple', 'cherry'}
<class 'set'>


{1, 3, 5, 7, 9}
<class 'set'>


{False, True}
<class 'set'>


{True, 9.95, 18, 'male', 'CSE_BSE', 'Ajith'}
<class 'set'>
```

Access Items

- You cannot access items in a set by referring to an index or a key.
- But you can loop through the set items
  - using a for loop,
  - or ask if a specified value is present in a set, by using the in keyword.

Example: Loop through the following sets, and print the values:

```
set1_fruits = {"apple", "banana", "cherry"}
for x in set1_fruits:
  print(x)
set2_numbers = {1, 5, 7, 9, 3}
for x in set2_numbers:
  print(x)



set3_boolean = {True, False, False}
for x in set3_boolean:
  print(x)
set4_mix= {"Ajith", 18, True, "CSE_BSE", 9.95, "male"}
for x in set4_mix:
  print(x)
```

```
apple
cherry
banana
1
3
5
7
9
False
True
True
CSE_BSE
9.95
male
18
Ajith
```

Example: Checking items in sets using ‘in’ keyword.

```
set_mix= {"Ajith", 18, True, "CSE_BSE", 9.95, "male"}
print("Ajith" in set_mix)
print("Antony" in set_mix)
print(18 in set_mix)
```

```
True
False
True
```

Add Items:

Once a set is created,

- we cannot change its items,
- but you can add new items.

Example: Add an item to a set, using the add() method:

```
set_mix= {"Ajith", 18, True, "CSE_BSE", 9.95, "male"}
print(set_mix)
set_mix.add("Email: sp@gmail.com")
print(set_mix)
```

```
{True, 9.95, 18, 'Ajith', 'male', 'CSE_BSE'}
{True, 9.95, 18, 'Ajith', 'Email: sp@gmail.com', 'male', 'CSE_BSE'}
```

Add Sets: To add items from another set into the current set, use the update() method

Example:

```
thisset = {"apple", "banana", "cherry"}
print(thisset)
tropical = {"pineapple", "mango", "papaya"}
thisset.update(tropical)
print(thisset)
```

```
{'cherry', 'apple', 'banana'}
{'cherry', 'banana', 'apple', 'papaya', 'pineapple', 'mango'}
```

Add Any Iterable: The object in the update() method does not have be a set, it can be any iterable object (tuples, lists, dictionaries et,).

Example: Add elements of a list to at set:

```
set_physics = {"Force",3.14, "Velocity", "Mass"}
print(set_physics)

mylist = ["Gravity", "Atom"]
print(mylist)

set_physics.update(mylist)
print(set_physics)
```

```
{3.14, 'Mass', 'Force', 'Velocity'}
['Gravity', 'Atom']
{'Gravity', 3.14, 'Mass', 'Force', 'Velocity', 'Atom'}
```

Removing Item from Sets:

- To remove an item in a set,
  - use the remove(),
  - or the discard() method.
  - pop() method

Example: Remove an item using the remove() method:

Note: If the item to remove does not exist, remove() will raise an error.

```
set_physics= {"Force",3.14, "Velocity", "Mass"}
print(set_physics)
set_physics.remove(3.14)
print(set_physics)

print("\n")
set_physics.remove("Time")
print(set_physics)
```

```
{'Force', 'Mass', 3.14, 'Velocity'}
{'Force', 'Mass', 'Velocity'}


Traceback (most recent call last):
  File "prgm22-Sets.py", line 140, in <module>
    set_physics.remove("Time")
KeyError: 'Time'
```

Example : Remove an item using the discard() method:

Note: If the item to be removed does not exist, discard() will **NOT** raise an error.

```
set_physics= {"Force",3.14, "Velocity", "Mass"}
print(set_physics)
set_physics.discard(3.14)
print(set_physics)

print("\n")
set_physics.discard("Time")
print(set_physics)
```

```
{'Mass', 'Velocity', 3.14, 'Force'}
{'Mass', 'Velocity', 'Force'}


{'Mass', 'Velocity', 'Force'}
```

pop() method to remove an item

- We can also use the pop(), method to remove an item,
- This method will remove the *last* item.
- Remember that sets are unordered,
- No idea about which item gets removed.
- The return value of the pop() method is the removed item.

Example: Remove the last item by using the pop() method:

```
set_physics= {"Force",3.14, "Velocity", "Mass"}
print(set_physics)
x=set_physics.pop()
print("Removed item: ",x)
print(set_physics)
```

n^{th} execution

```
{'Force', 3.14, 'Velocity', 'Mass'}
Removed item:  Force
{3.14, 'Velocity', 'Mass'}
```

$(n+1)^{th}$ execution

```
{'Mass', 'Force', 3.14, 'Velocity'}
Removed item:  Mass
{'Force', 3.14, 'Velocity'}
```

Activities:

1) Explore the use of clear () method in python-sets. Explain it with examples.

2) Explore the use of del keyword in python-sets. Explain it with examples.