Programming and Problem Solving through Python Language
O Level / A Level

Chapter - 5: Sequence Data Types

Python Collections (Arrays)

There are four collection data types in the Python programming language:

- **List** is a collection which is ordered and changeable. Allows duplicate members.
- **Tuple** is a collection which is ordered and unchangeable. Allows duplicate members.
- **Set** is a collection which is unordered and unindexed. No duplicate members.
- **Dictionary** is a collection which is unordered, changeable and indexed. No duplicate members.

Set

- A Set is a collection which is unordered and unindexed. In Python sets are written with curly brackets.
- The sets in python are typically used for mathematical operations like union, intersection, difference and complement etc.
- The Set is a datatype available in Python which can be written as comma-separated values (items) between parentheses.
- Items in a Set need not be of the same type.
- The set list is unordered means the items will appear in a random order on the screen.

Creating Set

```
set1 = { 'physics', 'chemistry', 1997, 2000 }
set2 = { 1, 2, 3, 4, 5  }
set3 = { "a", "b", "c", "d" }
```

Access Items

We cannot access individual values in a set. We can only access all the elements together.

```
set1 = { 'physics', 'chemistry', 1997, 2000 }
set2 = { 1, 2, 3, 4, 5, 6, 7  }
print ("set1: ", set1)
print ("set1[1]: ", set1[1)         # Index not allowed in Set
print ("tset2[3]: ", set2[-1])      # Negative Index not allowed in Set
```

Output−

```
set1:  {1997, 'physics', 'chemistry', 2000 }
```

Negative Indexing

Negative indexing not allowed in Sets.

Range of Indexes (Slicing)

Range of index or slicing is not allowed in Sets.

Range of Negative Indexes

Range of index or slicing is not allowed in Sets.

Updating Set

Once a set is created, you cannot change its items, but you can add new items.

```
set1 = { 12, 34.56 }
set2 = { 'abc', 'xyz' }

# Following action is not valid for sets

# set1[0] = 100
# set3 = set1 + set2
```

Adding new item in Set

- To add one item to a set use the **add()** method.
- To add more than one item to a set use the **update(collection)** method.

```
s = { 10, 20, 30, 40 }
s = t.add(60 )
print (t)
s = t.update( [ 70 , 80] )
print (t)
 s = t.update( 'ABC' )
```

Output

```
{ 10, 60, 30, 40, 20 }
{ 80, 10, 60, 30, 40, 20,70 }
{ 80, 10, 60, 30, 40, 20,70, 'A', 'B', 'C'}
```

Loop Through a Sets

You can loop through the **Set** items by using a for loop:

```
set = { "apple", "banana", "cherry"  }
for x in list:
    print(x)
```

Check if Item Sets

To determine if a specified item is present in a Set use the "**in**" keyword:

```
set = { "apple", "banana", "cherry"  }
if "apple" in set:
    print("Yes, 'apple' is in the fruits Set")
```

Length of Set

To determine how many items a **Set** has, use the **len( )** function.   e.g. print(len(set))

Removing Item from the Set

- To remove an item in a set, use the **remove()**, or the **discard()** method.
- If the item to remove does not exist, **remove()** will raise an error.
- If the item to remove does not exist, **discard()** will NOT raise an error.
- The **pop()** method to remove the last item. As the sets are unordered, so we will not know what item gets removed.
- The **clear()** method empties the set.   eg   set.clear()
- The **del** keyword will delete the set completely.   eg. del set

```
set = { "apple", "banana", "cherry"  }
set.remove("banana")
set.discard("apple")
```

Union of Sets

- The union operation on two sets produces a new set containing all the distinct elements from both the sets.
- We can use the **union**() method that returns a new set containing all items from both sets.
- We can use the **update**() method that inserts all the items from one set into another.
- We can use the **|** for union operation on two sets.

```
DaysA = {"Mon","Tue","Wed" }
DaysB = { "Wed","Thu","Fri","Sat","Sun" }
D1 = DaysA | DaysB
D2 = DaysA.union( DaysB  )
D3 = DaysA.update( DaysB  )
print(D1)
print(D2)
print(D3)
```

Output

```
{'Thu', 'Sat', 'Tue', 'Sun', 'Mon', 'Fri', 'Wed'}
{'Sat' , 'Thu', 'Tue', 'Sun', 'Mon', 'Wed', 'Fri'}
```

Intersection of Sets

- The intersection operation on two sets produces a new set containing only the common elements from both the sets.
- We can use the **&** for union operation on two sets.
- We can use the **intersection**() method returns a set that contains the similarity between two or more sets

```
DaysA = { "Mon", "Tue", "Wed" }
DaysB = { "Wed","Thu","Fri","Sat","Sun" }
D1 = DaysA & DaysB
D2 = DaysA.intersection( DaysB  )
print(D1)
print(D2)
```

Output
```
{'Wed'}
{'Wed'}
```

Difference of Sets

- The difference operation on two sets produces a new set containing only the elements from the first set and none from the second set.
- We can use the **-** for union operation on two sets.
- We can use the **difference**() method returns a set that contains the difference between two sets. The returned set contains items that exist only in the first set, and not in both sets.

```
DaysA = { "Mon", "Tue", "Wed" }
DaysB = { "Wed","Thu","Fri","Sat","Sun" }
D1 = DaysA - DaysB
D2 = DaysA.difference( DaysB  )
print(D1)
print(D2)
```

Output
```
{'Mon', 'Tue'}
{'Mon', 'Tue'}
```

Disjoint Sets

- The **isdisjoint**() method returns True if none of the items are present in both sets, otherwise it returns False.

```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}

z = x.isdisjoint(y)
print(z)
```

Output
```
False
```

Compare Sets

- We can check if a given set is a subset or superset of another set. The result is True or False depending on the elements present in the sets.
- The **issubset**() method returns True if all items in the set exists in the specified set, otherwise it retuns False. <= can also be used for subset.
- The **issuperset**() method returns True if all items in the specified set exists in the original set, otherwise it retuns False. >= can also be used for superset.

```
DaysA = set(["Mon","Tue","Wed"])
DaysB = set(["Mon","Tue","Wed","Thu","Fri","Sat","Sun"])
SubsetRes = DaysA <= DaysB
SupersetRes = DaysB >= DaysA
print(SubsetRes)
print(SupersetRes)

SubsetRes = DaysA.issubset(DaysB)
SupersetRes = DaysB.issuperset(DaysA)
print(SubsetRes)
print(SupersetRes)
```

Output

```
True
True
True
True
```

symmetric_difference

- **The symmetric_difference**() method returns a set that contains all items from both set, but not the items that are present in both sets.

```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}

z = x.symmetric_difference(y)
print(z)
```

```
Output    {'google', 'microsoft', 'banana', 'cherry'}
```

Assignment

1. Define Sets
2. Write the output from the following code:

```
x = {"a", "b", "c"}
y = {"c", "d", "e"}
z = {"f", "g", "c"}

result = x.intersection(y, z)
print(result)
```

```
t1=(10,20,30)
 print len(t1)
```

3. Write a program to input ‘n’ numbers in two sets and merge the Set in the following manner.

Example

```
 T1 =(10,30,50)
 T2=(20,40,60)
T=(10,20,30,40,50,60)
```