PYTHON PROGRAMMING

MODULE 4 – PART 5

SETS AND SET OPERATIONS, FROZENSET

OUTLINE

- Sets
- Create and Assign
- Access and Add Elements
- Remove Elements
- Basic Operations
- Set Operations
- Built-in Functions and Methods
- Frozenset

SETS

- Unordered
- Unindexed
- Unique elements
- Elements cannot be updated, but elements can be removed or added (hence mutable in the sense that items can be added or removed but existing values cannot be updated on the basis of positions)
- List, set, dictionary cannot be elements of a set

Creating set

```
s1={1,2,'abc',(1,2,3)}

#using set() function. Empty set, s2={} will create a dictionary
s2=set()

#takes an iterable as an argument
s3=set([1,2,3])

#repeated elements are considered only once
s4={2,3,4,2,1,4,3}
print(s1)
print(s2)
print(s3)
print(s4)
```

OUTPUT

```
{1, 2, (1, 2, 3), 'abc'}
set()
{1, 2, 3}
{1, 2, 3, 4}
```

Access and Add elements

Since there is no indexing, specific elements cannot be accessed.

```
s1={1,2,'abc',(1,2,3)}

#accessing set
print(s1)

#can add one element
s1.add(3)
print(s1)

#can add multiple values
s1.update('a','b')
print(s1)

OUTPUT

{1,2,'abc',(1,2,3)}
{1, 2, 3, (1, 2, 3), 'abc'}
{1, 2, 3, (1, 2, 3), 'b', 'abc', 'a'}
```

Remove elements

```
s1={1,2,'abc',(1,2,3)}
s1.discard(1)
s1.remove(2)
print(s1)

#removes arbitrary item
print(s1.pop())
print(s1)
s1.clear()
print(s1)
```

OUTPUT

```
{'abc', (1, 2, 3)}
abc
{(1, 2, 3)}
set()
```

Basic Operations (+ and * cannot be applied)

```
s1={1,2,3}
print(1 in s1)
print(1 not in s1)
for i in s1:
    print(i)
```

OUTPUT

```
True
False
1
2
3
```

Set Operations

```
s1={1,2,3,5}
s2={4,5,6}
print(s1.union(s2))
print(s1|s2)
print(s1.intersection(s2))
print(s1&s2)
print(s1.difference(s2))
print(s1-s2)
print(s2.difference(s1))
print(s1.symmetric_difference(s2))
print(s1^s2)
```

OUTPUT

```
{1, 2, 3, 4, 5, 6}
{1, 2, 3, 4, 5, 6}
{5}
{5}
{1, 2, 3}
{1, 2, 3}
{4, 6}
{1, 2, 3, 4, 6}
{1, 2, 3, 4, 6}
```

Functions Discussed Earlier

1) set(sequence)
2) add(element)
3) update(element1, element2,…)
4) discard(element)
5) remove(element)
6) pop( )
7) clear( )
8) union(set)
9) intersection(set)
10) difference(set)
11) symmetric_difference(set)

Functions Discussed Next

1) len(set)
2) max(set)
3) min(set)
4) sum(set)
5) any(set)
6) all(set)
7) sorted(list, reverse=True/False, key=myFunc)
8) enumerate(set)
9) intersection_update(set)
10) difference_update(set)
11) symmetric_difference_update(set)
12) copy( )
13) isdisjoint(set)
14) issubset(set)
15) issuperset(set)

Built in functions and methods

```
set1={4,2,8,10,6,4,2}

#repeated elements are counted only once
print(len(set1))
print(max(set1))
print(min(set1))

#repeated elements are considered only once
print(sum(set1))
set2={0}
set3={'0'}
set4={0,1,2,3}

#returns True even if one element has boolean value True
print(any(set1),any(set2),any(set3),any(set4))

#returns True only if all elements have boolean value True
print(all(set1),all(set2),all(set3),all(set4))
print(sorted(set1,reverse=True))

#pairs set elements with an index value and returns as list
print(list(enumerate(set1)))
```

Output

```
5
10
2
30
True False True True
True False True False
[10, 8, 6, 4, 2]
[(0, 2), (1, 4), (2, 6), (3, 8), (4,10)]
```

```
s1={1,2,3}
s2={3,4,5}
s3={5,6,7}
s4={7,8,9}

#performs intersection and updates the original set
s1.intersection_update(s2)
print("After intersection update s1 and s2 :")
print(s1)
print(s2)

#performs difference and updates the original set
s2.difference_update(s3)
print("After difference update s2 and s3:")
print(s2)
print(s3)

#performs symmetric difference and updates the original set
s3.symmetric_difference_update(s4)
print("After symmetric difference update s3 and s4:")
print(s3)
print(s4)
```

OUTPUT

```
After intersection update s1 and s2 :
{3}
{3, 4, 5}
After difference update s2 and s3:
{3, 4}
{5, 6, 7}
After symmetric difference update s3 and s4:
{5, 6, 8, 9}
{8, 9, 7}
```

```
s1={1,2,3}
s2=s1.copy()
print(s1)
print(s2)
s3={2,3}
print(s1.isdisjoint(s3))
print(s3.issubset(s1))
print(s1.issuperset(s3))
```

OUTPUT

```
{1, 2, 3}
{1, 2, 3}
False
True
True
```

The frozenset

Frozenset can be considered as an ***immutable set***. In general, frozenset() is an inbuilt function that takes any iterable object as argument and freezes them (makes them immutable). Therefore, a frozenset can be a key for dictionary but a set cannot.

Syntax : frozenset(iterable_object)

```
#converts set into a frozenset
set={1,2,3
}print(frozenset(set))

#converts list into frozenset
list=[1,2,3]
print(frozenset(list))

#frozenset as key for dictionary
dictionary={frozenset(set):1, 'abc':2, 'def':3}
print(dictionary)
```

Output

```
frozenset({1, 2, 3})
frozenset({1, 2, 3})
{frozenset({1, 2, 3}): 1, 'abc': 2, 'def': 3}
```

Study at Home

Notes

Learn topics better with a little less effort utilizing notes by subject experts.

Syllabus

View or download syllabus for Bsc Computer Science under Calicut University.

Video Lessons

Better learning experience through video lectures from skilled personnel.

Question Papers

Familiarize with exam patterns through our library of previous year question papers.

Quiz

Test your knowledge of topics through various quizzes.

Solved Questions

Vast collection of questions and answers related to each topic.

Programming Laboratory

Practicals made easy with complete list of tested programs.

Software Downloads

Practice programming by downloading and installing respective softwares.

Teaching Resources

Find PPTs and related resources for teaching aid here.

teachics.

The Computer Science Learning Platform.

Thank You