4-Aug-2020

Assignment No. 1

21118

Name- Shubham Chemate.
SE-1 (E-1)

Problem statement:

In second year engineering class, group A student's play cricket, group B student's play badminton and group C student's play football.
Write a Python program using function to compute the following.

a> list of students who play both cricket & badminton

b> list of students who play either cricket or badminton but not both.

c> Number of students who play neither cricket nor badminton

d> Number of students who play cricket & football but not badminton.

(Note: while realizing the group, duplicate entries should be avoided, Do not use set Inbuilt functions)

Objectives:

1> To understand the concept ~~of~~ operations of set.
2> To understand primitive functions of list data structure in Python

Outcomes

1> To implement set operations using list data structure in python.
2> To write menu driven, modular program in python.
3> To implement user defined functions in python.

Hardware Requirements.

~~processor~~
Manufacturer: Acer

Model: Swift Sf 314 - 55G

Processor: Intel(R) Core(TM) i5-8265U CPU @ 1.60 GHz
1.80 GHz

Intalled Memory(RAM): 8.00 GB (7.85GB usable)

System Type: 64-bit Operating System, x64-based
processor

Pen and Touch: No pen @ Touch Input is available

Software Requirements:

Operating System: Windows 10 Home Single language
: (Version: 1903)

Python Version: 3.8.5

VS Code (text editor): Version: July 2020 (Version 1.48)

Theory

Concepts:

Set: A collection of well defined, distinct objects

Set operations: Union, Intersection, Difference

Union: ~~common~~ elements of two sets, present in either set

Intersection: common elements of two set ~~is~~.

Difference: difference of set A with B is defined as elements of A which are not in B.

Class: Class is a blueprint of objects. It provides way to implement various oop concepts (eg. data hiding, abstraction)

objects: object is an instant of a class. for same class there can be multiple objects.

oop concepts: data hiding, abstraction

data hiding: work through function and don't play directly with actual data.

Abstraction: showing important details & hiding unnecessary things.

Also list is used to implement set and basic python syntax knowledge is ~~requir~~ used.

ADT:

- Abstract data type (ADT) is a type (~~o~~ class) for objects whose behavior is defined by a set of value and set of operations.
- The defination of ADT only mentions what operations are to be performed but not how these operations will be implemented.
- It doesn't speify how data will organised in memory & what algorithm will be used for implemeatation.
- As a ADT, sets have two key characteristics
    - They contain no duplicates
    - The item within the sets are unordered
-

class declaration:

class Set is declared with empty list and length as a class member variables ~~empty~~ list will be used to store set elements & length will be the count of elements in a set.
The ~~empty~~ list & length both variables are initialized inside the constructor.
Other methods in the class include addElement(), intersection(), union(), and difference().
High level python code for class is as below.

```
class Set:
    def __init__(self).
        self.myset = []    # ~~empty~~ list to store set element
        self.length = 0    # count of elements in set

    def addElement(self, element)   # adds element to
    # code                              myset
```

```
def intersection(self, anotherSet):   # ~~get~~ return intersection
    # code                            of set with another set
def isContain(self element):          # check wheather
    # code                            element is present in
                                      set @ not
def union( self, anotherSet):         # return union of set
    # code                            with another set
def difference(set, anotherSet):      # return difference of
    # code                            set with another set
```

Algorithms: for set operations:

```
Algorithm createSet(set):
{ //This function will create an Empty set named as
  //set
    1. set = []
}
```

```
Algorithm addElement(element, set)
{ // This function will add element to the set
    1. read element
    2. if (element not in set) then
        2.1 Add element to the set
}
```

```
Algorithm unionSet(set1, set2, unionSet)
{ //union will be union of set1 & set2
    1. read set1 & set2
    2. set union []
    3. for (element in set1) do
        3.1 if (element not in union) then
            3.1.1 Add element to union
    4. For(element in set 2) do
        4.1 if(element not in union) then
            4.1.1. Add element to union.
    5. return union.
}
```

```
Algorithm Intersection set (set1, set2, intersection)
{ // intersection will be intersection of set1 & set2.
    1. read set1 & set2
    2. set intersection = []
    3. for (element in set) do
        3.1 if (element in set1) do
            3.1.1 Add element to intersection.
    4 return intersection.
}
```

Analysis of Algorithm:

Method / Algorithm	Time Complexity
1. addElement()	O(n) where n = no. of elements in set
2. Create set()	O(1)
3. Unionset()	O(mn) where m,n = size of 1st & 2nd set
4. Intersectionset()	O(mn) where m,n = size of 1st & 2nd set
5. isContain()	O(n) where n = size of set
6. difference()	O(mn) where m,n = size of 1st & 2nd set

Test Cases: consider A = cricket players, B = badminton player & C = football players (a,b,c,d → for problem statement)

Test Case No	Test Case Description	Expected Output	Actual Output

1.	A = ~~empty~~ {A,B,C}
	B = {A,B,C}
	C = {A,B,C}
	ie. all set with same elements
		a. {A,B,C}
		b. No students
		c. ~~No student~~ 0
		d. ~~No student~~ 0
			a. A,B,C
			b. No student
			c. 0, No student
			d. 0, No student

2.	A = empty
	B = {A,B,C,D}
	C = {C,A,E,F}
	ie. three sets with one empty
		a. ~~[illegible]~~ {}
		b. {A,B,C,D}
		c. {2}
		d. 0
			a. No students
			b. {A,B,C,D}
			c. 2, {E,F}
			d. 0, No students

3.	A = {A,B,C,D}
	B = {C,D,E}
	C = {E,A,B,F,G}
	ie. three set with some common elements.
		a. {C,D}
		b. {A,B,E}
		c. 2
		d. 2
			a. {C,D}
			b. {A,B,E}
			c. 2, {F,G}
			d. 2, {A,B}

Applications:

1) mathematical structures ~~applications~~: many mathematical structures like graphs, manifolds, rings & vector spaces can be represent using sets.

2) Many mathematical theorems are derived using set theory eg properties of real numbers, & natural numbers

3) Also set theory is foundation for mathematical analysis, topology, abstract algebra & discrete mathematics.

source: wikipedia.org

Conclusion:

The set theory is one of the important aspect of discrete mathematics. Many mathematics proofs are derived using the sets. But before going into the details it is essential to knao properties of set & basic operations on them.

This assignment covers all basic operations of set using list data type. All basic operations are covered in pretty much detail that one can move to the advanced mathematics of set from here.