PYTHON PROGRAMMING
PYUNIT LAB PROGRAMS

1. **Isogram**

Determine if a word or phrase (**input1**) is an isogram. An isogram (also known as a "nonpattern word") is a word or phrase without a repeating letter, however spaces and hyphens are allowed to appear multiple times.
Examples of isograms:
lumberjacks
background
downstream
six-year-old
The function prototype is given below:

```
class Isogram():
        def is_isogram (input1):
                # Write Your Code
```

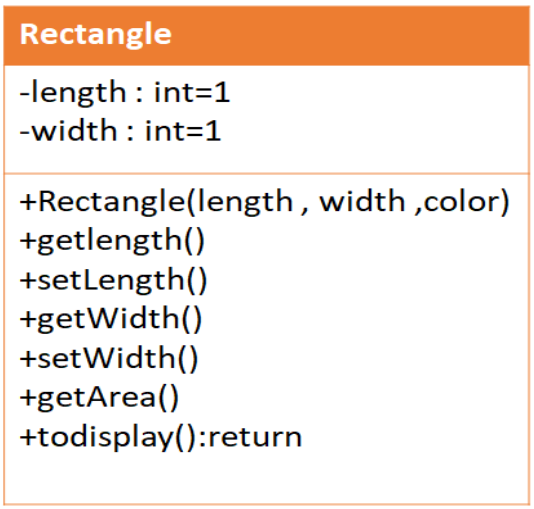
Assumptions:

- The input can contain both lowercase and upper case alphabets along with punctuations.
- The output should be a Boolean value.

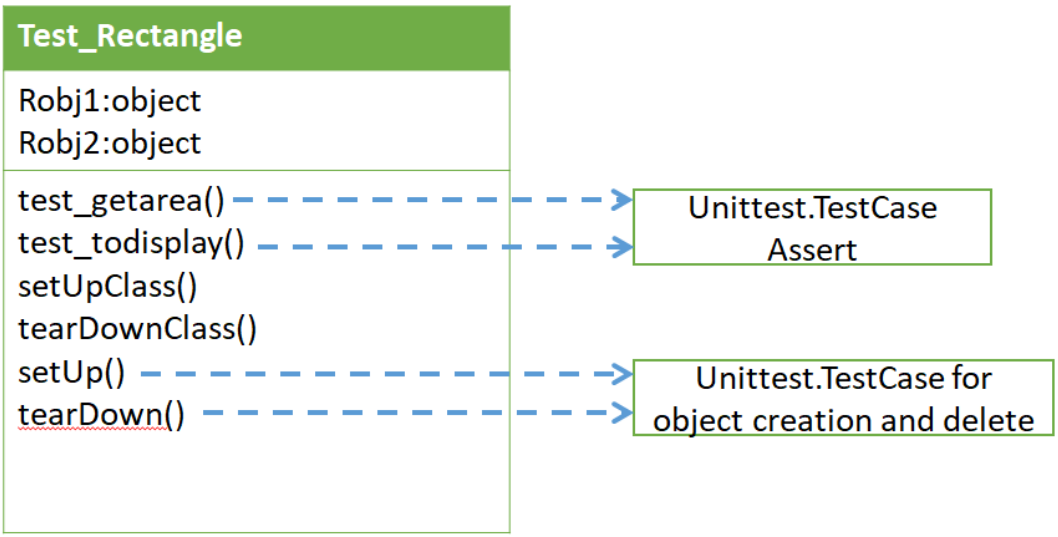
Sample Input	**Expected Output**
input1= isogram	True
input2= eleven	False

Write a TestCase class **IsogramTest** to test your program with at least **10** testcases.

2. i)Develop a python program for given class diagram ,



ii) Write a TestCase class for the below class diagram,



Assumptions:

test_getarea()	
Sample Input	**Expected Output**
Length: 10 Width=3	30
Length: 20 Width=7	14

test_todisplay()	
Sample Input	**Expected Output**
Length: 10 Width=3 Color='red'	Area of rectangle 30 red in color
Length: 20 Width=7 Color='green'	Area of rectangle 140 green in color

3. Write a TestCase class for the given program Employee.py with the following constrains,

i)Skip monthly_incentive testcase using skip methods
ii)Use Fixed Failure for apply_raise testcase

Employee.py

```
class Employee:

    raise_amt = 1.05

    def __init__(self, first, last, pay,noofyear):
        self.first = first
        self.last = last
        self.pay = pay
        self.noofyears=noofyear

    def email(self):
        return '{}.{}@email.com'.format(self.first, self.last)

    def fullname(self):
        return '{} {}'.format(self.first, self.last)

    def apply_raise(self):
```

```
        self.pay = int(self.pay * self.raise_amt)

    def monthly_incentive(self, incentiveamount):
        self.pay=int(self.pay+self.incentiveamount)
        return self.pay
    def yearly_incentive(self, incentiveamount):
        self.pay=int(self.pay+self.incentiveamount)
        return self.pay

    def yearofexprences(self):
        return self.noofyear

    def details(self):
        return [self.fullname,self.email,self.yearofexprences()]
```

4. **i) Sum Of Multiples**

Given a number **N (input1)** and a list of numbers **L (input2),** find the sum of all the multiples of the numbers in the list up to that number **N**. The function prototype is given below:

```
class SumOfMultiples():
        def get_sum_of_multiples(input1, input2):
                # Write Your Code
```

Assumptions:

- All input numbers are non-negative integers, i.e. natural numbers including zero.
- A list of factors must be given, and its elements are unique and sorted in ascending order.

Sample Input	Expected Output	Explanation
input1= 4 input2= [3, 5]	3	• The given number is 4 • The factors are 3 and 5 • The only multiple of 3 in the range 1 to 4 = 3 • The multiple of 5 in the range 1 to 4 =0 • The sum of all the multiples of the numbers = 3 + 0 = 3

ii) Leap in Python

Given a year, return Ture if it is a leap year.
The tricky thing here is that a leap year in the Gregorian calendar occurs:

- on every year that is evenly divisible by 4
- except every year that is evenly divisible by 100
- unless the year is also evenly divisible by 400

The function prototype is given below:

```
class LeapYear():
```

Write a TestCase class **TestSumOfMultiples** and **TestLeap** to test your program with at least **10** testcase and run the both testcase class using suite.