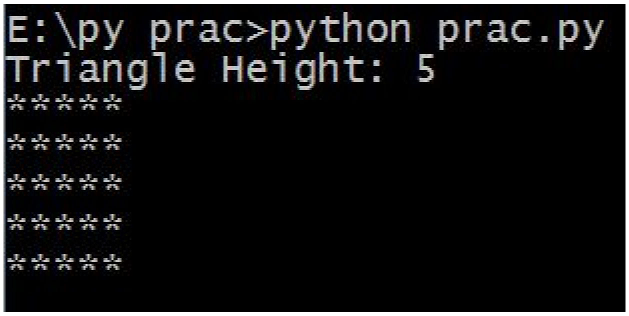
Conditional Statements (If-Else) and Loops (While, For):

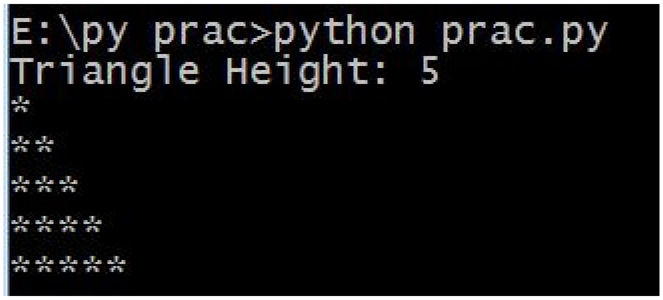
1. **In the following problems you will have to draw different symbols using the asterisk (*) symbol. Each symbol has a dimension, which you will have to take as input. You will have to write two versions of code for each symbol, in version 1 you will must use nested loop, while in version 2 you cannot use nested loop.**

a.



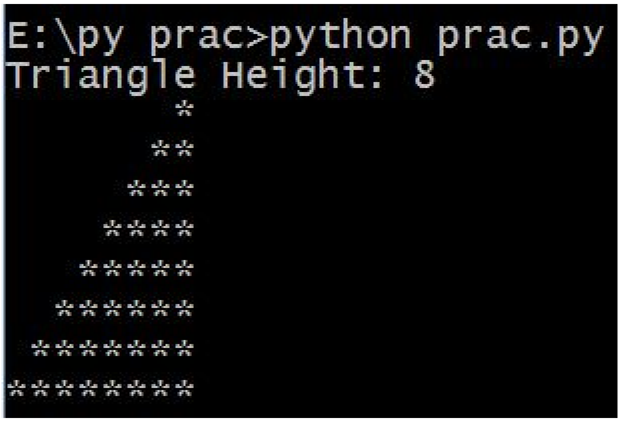
```
E:\py prac>python prac.py
Triangle Height: 5
*****
*****
*****
*****
*****
```

b.



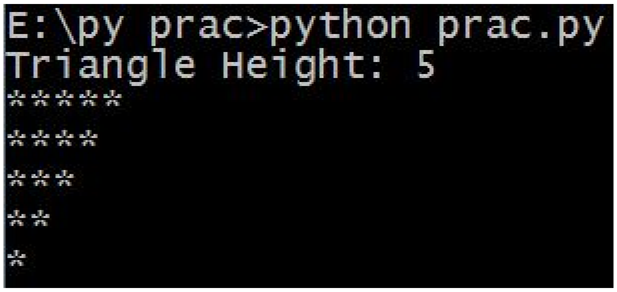
```
E:\py prac>python prac.py
Triangle Height: 5
*
**
***
****
*****
```

c.



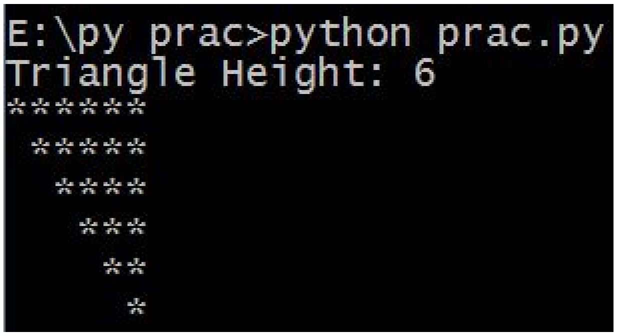
```
E:\py prac>python prac.py
Triangle Height: 8
       *
      **
     ***
    ****
   *****
  ******
 *******
********
```

d.



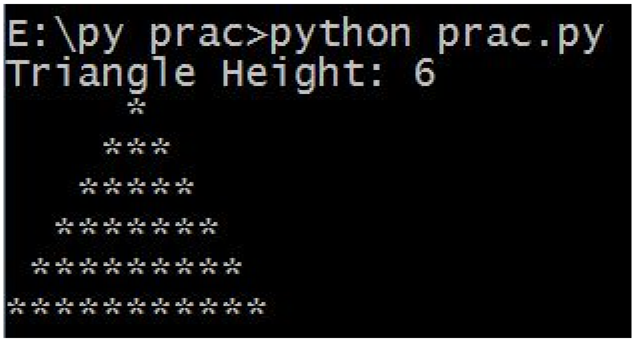
```
E:\py prac>python prac.py
Triangle Height: 5
*****
****
***
**
*
```

e.



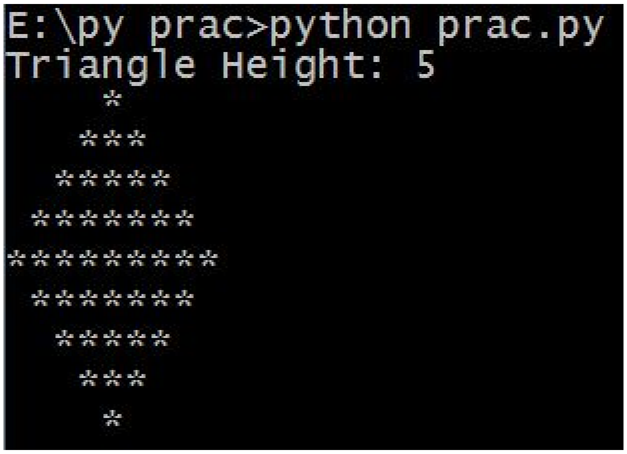
```
E:\py prac>python prac.py
Triangle Height: 6
******
 *****
  ****
   ***
    **
     *
```

f.



```
E:\py prac>python prac.py
Triangle Height: 6
     *
    ***
   *****
  *******
 *********
***********
```

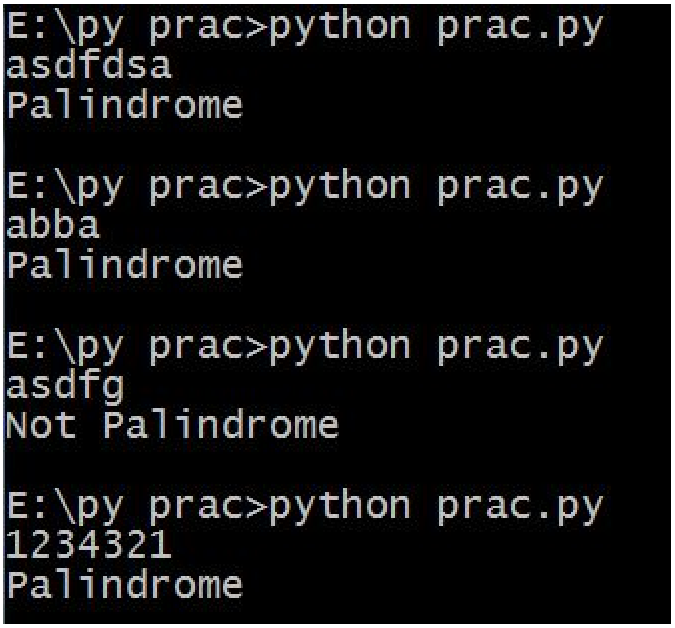
g. Technically, for this problem the triangle height is actually 2*input-1
Hint: check what range(10,0,-1) does



```
E:\py prac>python prac.py
Triangle Height: 5
    *
   ***
  *****
 *******
*********
 *******
  *****
   ***
    *
```

2. Take a binary string from user and print its decimal value
3. Take a string from user, print its reverse in console without using any library function except print()

4. **Take a string from user, and detect whether it is a palindrome.**
   Example:

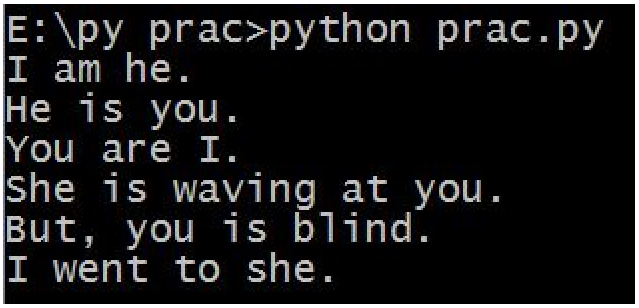


```
E:\py prac>python prac.py
asdfdsa
Palindrome

E:\py prac>python prac.py
abba
Palindrome

E:\py prac>python prac.py
asdfg
Not Palindrome

E:\py prac>python prac.py
1234321
Palindrome
```

5. Take a integer from user. Print whether it is a prime or not.
6. Implement “Sieve of Eratosthenes”. You can see this tutorial to understand the algorithm: http://www.geeksforgeeks.org/sieve-of-eratosthenes/

Files:

1. **Create a file named “Hello.txt” using Python and write the line “Hello Python” in it. [Open the file in notepad/notepad++ to verify]**
2. **Write a Python script to open the file created in the previous task (i.e., Hello.txt) and show its content in the console.**
3. **Open the file “dummy.txt” [shared in google drive] using Python. You will have to print each sentence of that file in a separate line in console. Each sentence is ended with full stop (“.”). Your output should look like this:**

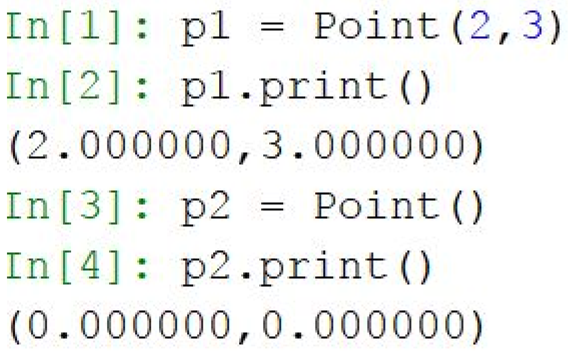


```
E:\py prac>python prac.py
I am he.
He is you.
You are I.
She is waving at you.
But, you is blind.
I went to she.
```

4. **Open the csv file “marks.csv”. Print the average, max, and min marks for CT1**
5. **Open the csv file “marks.csv”. Write a Python script to find out who got the highest marks in CT1 and who got lowest in CT2 ?**
6. **Open the csv file “marks.csv”. Write a Python script to find out the total marks for each student.**

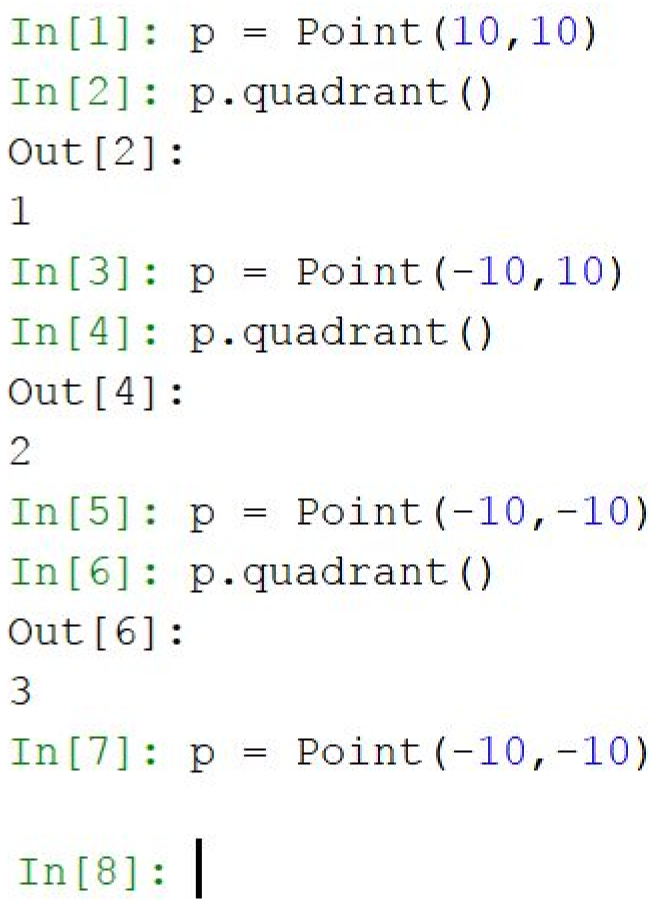
OOP

1. Define a class named Point which represents point in 2 dimensional space. A Point in a two dimensional plane has double x and double y coordinates. The initialization function takes the x,y coordinates of the point. However, if no argument is given the point is initialised to be the origin. The class will also have a print function. The sample behavior is shown below:



```
In[1]: p1 = Point(2,3)
In[2]: p1.print()
(2.000000,3.000000)
In[3]: p2 = Point()
In[4]: p2.print()
(0.000000,0.000000)
```

2. Now that you have already a Point class, it is time to improve it. Add a method that returns on which quadrant it falls in. The sample behavior is shown next:



```
In[1]: p = Point(10,10)
In[2]: p.quadrant()
Out[2]:
1
In[3]: p = Point(-10,10)
In[4]: p.quadrant()
Out[4]:
2
In[5]: p = Point(-10,-10)
In[6]: p.quadrant()
Out[6]:
3
In[7]: p = Point(-10,-10)

In[8]:
```

3. Do you think the above example covers all the cases? Think of ways to improve by adding more methods and/or improving the quadrant method.

4. Add another method in the class named “distance(point)”, which takes another Point object and return the distance between the two points (One object is on which the method is being called and the other one is the one which is being passed in the method).

5. Congratulations, you boss Mr. Musk is happy with your work with the points. Now he wants a new class "Cloud" with the following requirements:

A cloud is a collection of Points, i.e., a cloud object can hold/save a number of Point objects. Each cloud has a limit (maxLimit). The cloud object can add new points or remove old points. At any instance it can tell how many Points it currently holds. Given a point, it can check whether it is already in Cloud. Methods isFull and isEmpty will check whether the Cloud is full or empty. If possible implement the add function in such a way that if I add one point, it immediately lets me know whether I can add more points or not.

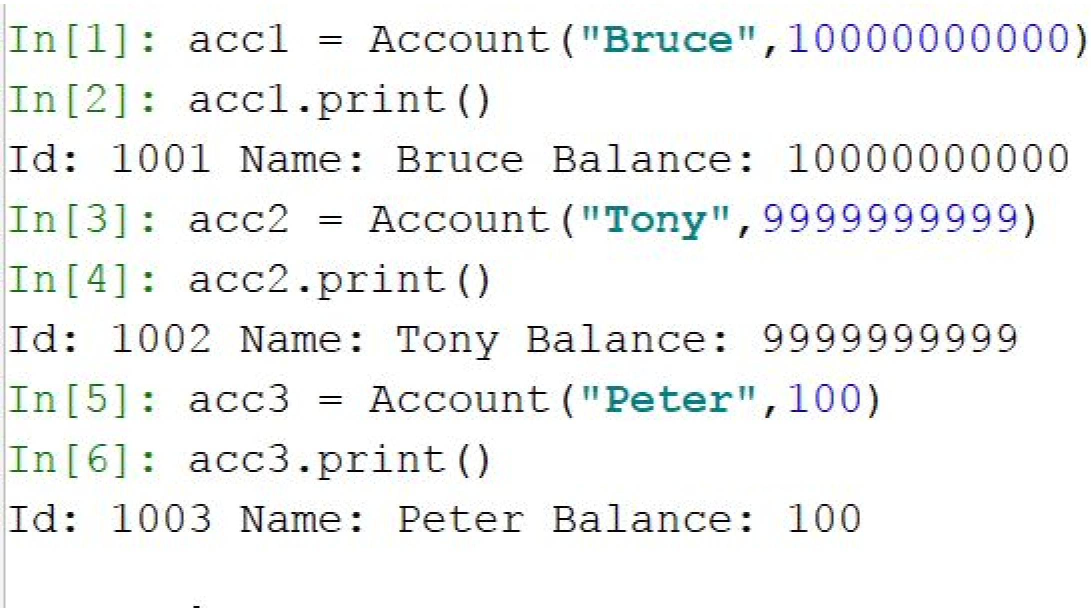
There will a print method that prints all the point the cloud currently holds. The extremes method returns null if the Cloud is empty, and an tuple of Points: leftmost, rightmost, top, bottom otherwise. The centerP method creates a center Point (average x and average y of the Cloud). The minDist method returns null if the Cloud is empty, and an tuple of the two Points in the Cloud that are closest together otherwise.

Now, design the class "Cloud" according to your Mr. Musk's needs. Also write a script calling different "Cloud" methods to demonstrate its functionalities, so that he is impressed on you and give you a big raise and also a brand new Tesla!!

6. As you have a lots of money now from the raise, create a class for a bank account. Each account is identified by a unique number.
The class will have the following properties:
a) unique id
b) account holder's name
c) balance

The class will provide the following functionalities:
a) creating the account with an initial balance
b) deposit(amount) method that takes an amount and adds it to the balance
c) withdrawal(amount) method that takes an amount and deducts that amount from the account
d) print() method that prints the id, name, and balance
e) history() method that prints the history of last five transactions(deposit and withdrawal)

The id will be assigned as follows: The first account object created will have the id 1001, the next one's id will be 1002, and so on like the following:



```
In[1]: acc1 = Account("Bruce",10000000000)
In[2]: acc1.print()
Id: 1001 Name: Bruce Balance: 10000000000
In[3]: acc2 = Account("Tony",9999999999)
In[4]: acc2.print()
Id: 1002 Name: Tony Balance: 9999999999
In[5]: acc3 = Account("Peter",100)
In[6]: acc3.print()
Id: 1003 Name: Peter Balance: 100
```

Assume initial balance and all deposit/withdrawals are in integer.