Assignment 3

Classes and Objects

Instructions:

1) Use of any inbuilt libraries is not allowed. All the required libraries are already imported.
2) For plagiarism, institute policies will be followed strictly.
3) **Make changes only in a3.py** and do not rename any file or create new files.
4) You need to submit a single zip file titled 'a3_yourrollnumber.zip' with your code files 'a3.py' in it. Make sure to follow the convention exactly, otherwise it will fetch 0 marks directly.
5) Make sure your code is well commented.
6) Use classroom discussion for any doubt.
7) Since automated tests will be performed, make sure your sample tests run successfully.
8) You have to **report upto max of 2 decimal places**. You have to round to the nearest place not truncate. E.g. 8.0 will remain 8.0 and 8.3 will be 8.3 but 8.375 will round to 8.38.

Reference material:

Some reading materials are included for reference. Make sure to go through it at least once before attempting the assignment.

1) Basics of matrix multiplications: https://www.mathsisfun.com/algebra/matrix-multiplying.html
2) Basics of 2D transformations: https://www.tutorialspoint.com/computer_graphics/2d_transformation.htm
3) A great video on how linear transformations can be done with matrices: https://youtu.be/kYB8IZa5AuE?list=PL0-GT3co4r2y2YErbmuJw2L5tW4Ew2O5B
4) Plotting circles: https://matplotlib.org/stable/api/_as_gen/matplotlib.patches.Circle.html
5) Reference to numpy docs: https://numpy.org/doc/stable/reference/

Goal:

- Your task is to write an **interactive python program** to apply transformations to an object and plot it using matplotlib. Your program should support the following objects:-
    1) A circle of radius r centered at (a,b).
    2) A polygon (vertices of which are specified by lists X and Y in clockwise or counter-clockwise order)

- Your 'a3.py' file will contain a single class named 'Shape' which contains the matrices with which you have to perform any transformations. 'a3.py' file will also contain 2 inherited classes

‘Polygon’ and ‘Circle’. You have to modify the Polygon and Circle classes only to implement the following functions:

1) __init__(): Initializations have to be done here.

2) translate():

   Input:
   translate() take in 2 arguments dx and dy (dy is optional). Here, dx is a number which implies how much to translate with along x-axis and similarly, dy along y-axis. If dy is not given, then assume equal translation dy = dx.

   Output (Polygon):
   Translate return x_new and y_new which are the list of transformed coordinates in the same order as the input.

   Output (Circle):
   return new x, y coordinates and the radius

3) rotate():

   Input:
   rotate() takes in 3 arguments deg (Φ), rx, ry (rx and ry are optional arguments which implies coordinates of arbitrary point from where rotation will take place). If rx and ry are not given, assume rotation from origin and deg (Φ) is a number which implies by which angle to rotate the shape.

   Output (Polygon):
   rotate() return x_new and y_new which are the list of rotated coordinates in the same order as the input.

   Output (Circle):
   return new x, y coordinates and the radius

4) scale():

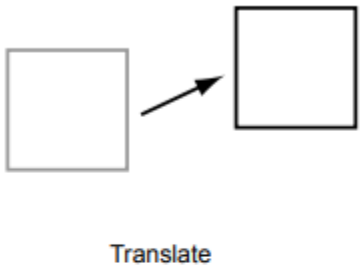
   Input:
   scale() takes in 2 arguments, sx and sy in case of a polygon and 1 argument s in case of a circle. Here, sx, sy and sr implies a number by which to scale. In case of a polygon, if only 1 argument is given, then assume sy = sx.

Note 1: Scaling has to be done with respect to the center of the shape. You can find the center of the shape using arithmetic mean of x coordinates and y coordinates.

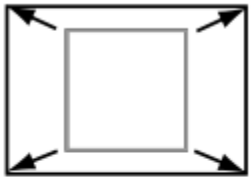
Note 2: For all transformations, you have to use the matrices from base class only after inheritance. For example,
For translation, T_t matrix has to be used.
Similarly, for scaling, T_s matrix has to be used and T_r has to be used for rotation.

Output (Polygon):
scale() return x_new and y_new which are the list of rotated coordinates in the same order as the input.

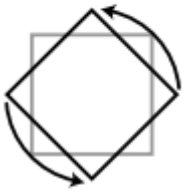
Output (Circle):
return new x, y coordinates and the radius



Translate



Scale



Rotate

5) plot()
Plot the initial and the transformed shape using matplotlib library. Plot the before and after transformation plots.
Before plot will be plotted with a dashed line and after plot with solid line.

Input Format:

First line contains a **verbose** which takes in 0 or 1 as Integer. If verbose is given as 1, then you have to show plots after every transformation along with printing the results otherwise just print the results. For results, see output format.

Next line contains an integer 0 or 1 described below denoting the figure you have to work on.

If User inputs 1, it represents a '**circle'** and the next line should contain separated integers **a b r** as specified above. In case user inputs 0, it represents a **'polygon'** and the next line contains a single integer n, the number of sides of the polygon. The next n lines contain space separated **x** and **y** denoting x-y coordinates of the polygon.

The next Line contains a positive integer **Q** which denotes the number of queries to be performed.

Next Q lines contain any one of the below queries:

In case of a Polygon, each query will be of type

a) **R theta (rx) (ry)**-> Rotation (theta: +ve -> clockwise, -ve -> anti-clockwise) about the origin by theta degrees
b) **S sx (sy)** -> Scale by a factor of x along x-axis and y along y-axis
c) **T dx (dy)** -> Translate by dx along x-axis and dy along y-axis
d) **P** -> Plot the shape (along with the shape just before the previous query). Hence, showing the transformation.

In case of a Circle, each query will be of type

a) **R theta (rx) (ry)** -> Rotation (theta: +ve -> clockwise, -ve -> anti-clockwise) about the origin by theta degrees.
b) **S sr** -> Scale by a factor of s along all the sides
c) **T dx (dy)** -> Translate by dx along x-axis and dy along y-axis
d) **P** -> Plot the shape (along with the shape just before the previous query). Hence, showing the transformation.

* () parentheses denote optional parameters.

Input Constraints:

1 <= T, Q <= 20
Shape = 0 or 1 (1 denotes a circle and 0 denotes a polygon)

-360 <= theta <= 360
-100 <= x, y, dx, dy, rx, ry, r <= 100 (initial coordinates, radius)
-10 <= sx, sy, sr <= 10
T, Q will be integer and the rest will be float.

Note: Each transformation will be performed on the shape obtained as a result of all the previous transformations in a cumulative manner. The first transformation will be performed on the original state of the shape as provided by the user.

Output Format:

For each query you need to output the following:

- For the circle, a b r space separated before the transformation and then in the next line updated a b r values. Here, a and b are x and y coordinates of center and r is the radius.
- For the polygon, results will be space separated x and y lists before transformation and then in the next line transformed x and y lists.
- If verbose = 1, after every transformation plot the previous and current state of your shape.

Sample Input 1: (Bold part denotes the input)

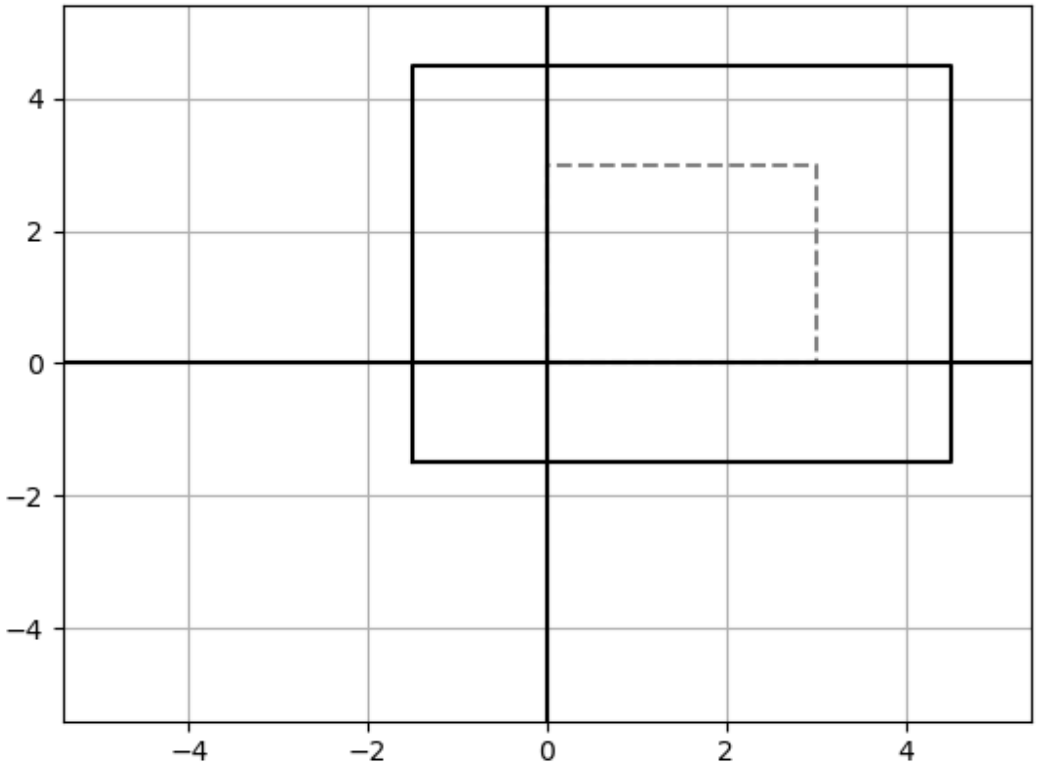
verbose? 1 to plot, 0 otherwise: **0**
Enter the number of test cases: **1**
Enter type of shape (polygon/circle): **0**
Enter the number of sides: **4**
enter (x1, y1): **1 1**
enter (x2, y2): **1 5**
enter (x3, y3): **5 5**
enter (x4, y4): **5 1**
Enter the number of queries: **4**
Enter Query:
1) R deg (rx) (ry)
2) T dx (dy)
3) S sx (sy)
4) P

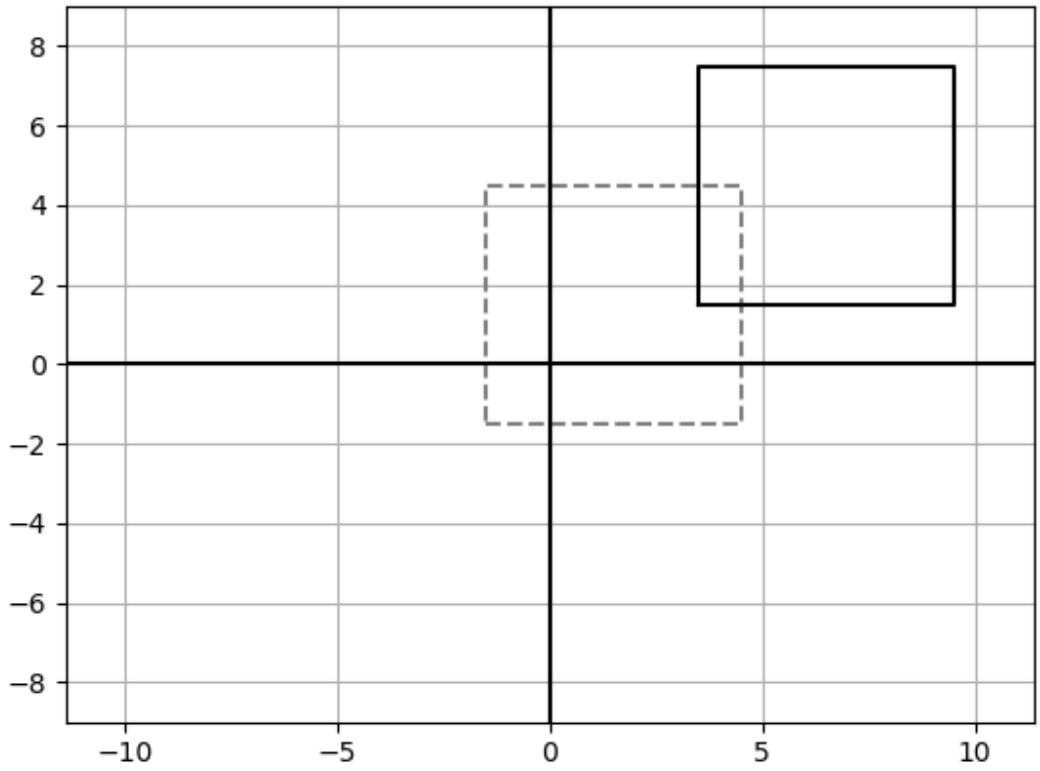
T 2 2
1.0 1.0 5.0 5.0 1.0 5.0 5.0 1.0
3.0 3.0 7.0 7.0 3.0 7.0 7.0 3.0
R 90
3.0 3.0 7.0 7.0 3.0 7.0 7.0 3.0
3.0 7.0 7.0 3.0 -3.0 -3.0 -7.0 -7.0

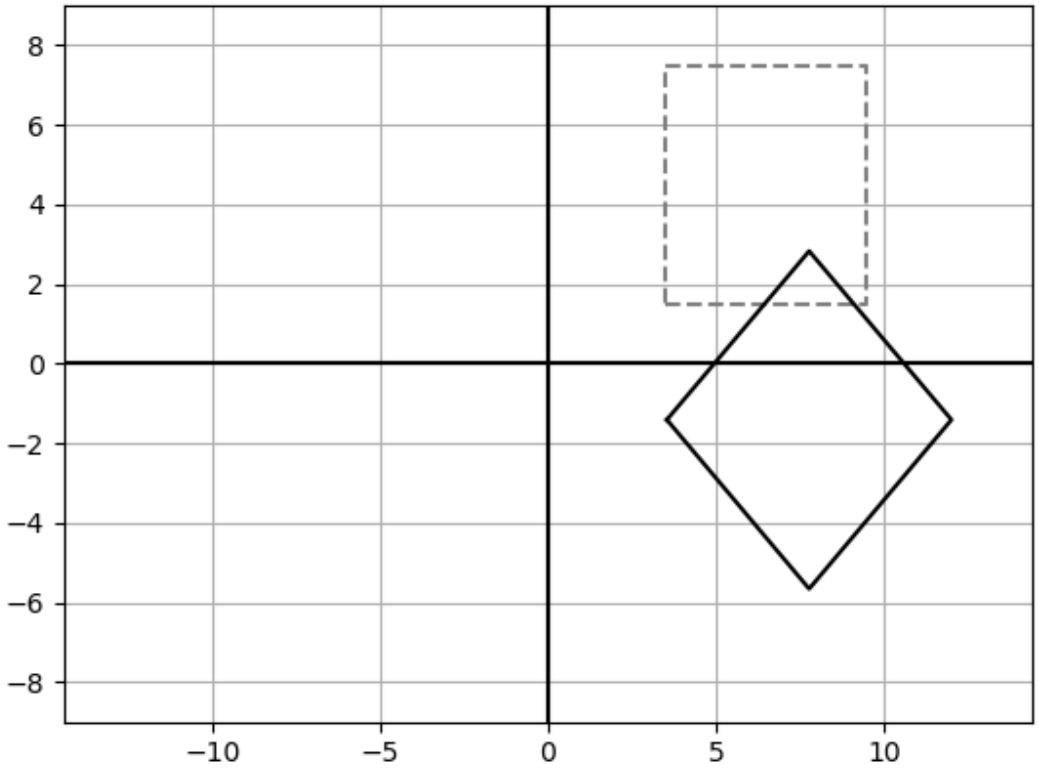
R -45
3.0 7.0 7.0 3.0 -3.0 -3.0 -7.0 -7.0
4.24 7.07 9.9 7.07 0.0 2.83 0.0 -2.83
S 2
4.24 7.07 9.9 7.07 0.0 2.83 0.0 -2.83
1.41 7.07 12.73 7.07 0.0 5.66 0.0 -5.66

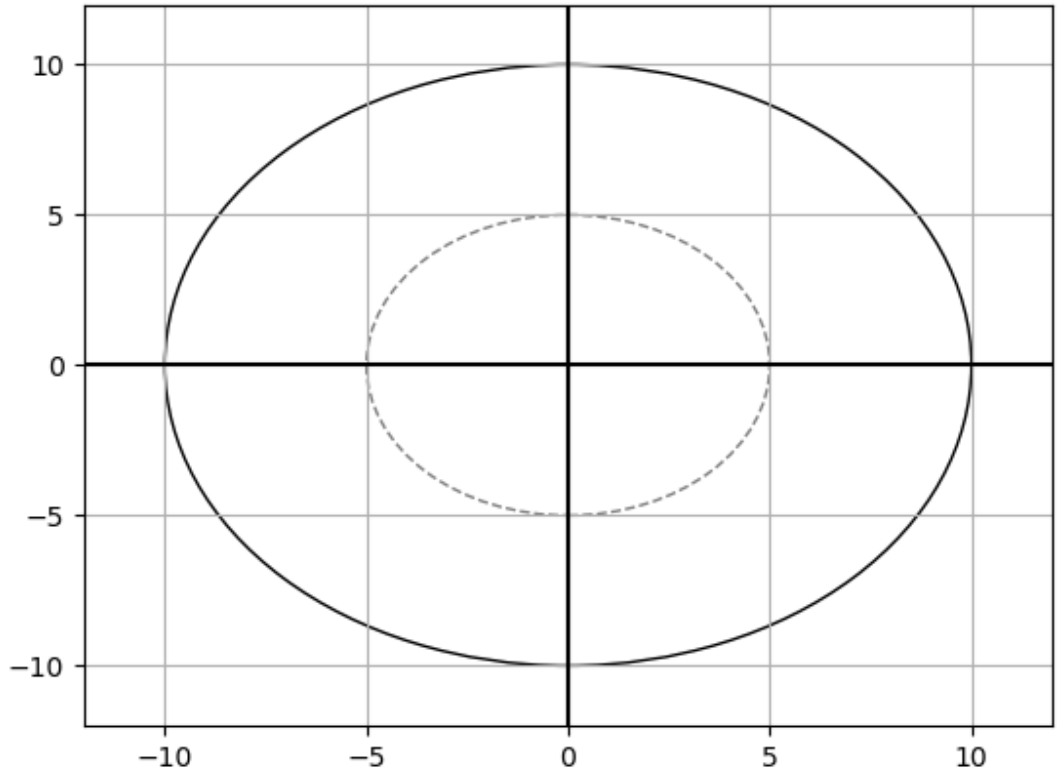
Sample Input 2: (Sample plots)
1
2
0
4
0 0
3 0
3 3
0 3
3
S 2 2
T 5 3
R 45
1
0 0 5
3
S 2
T 5 3
R 45

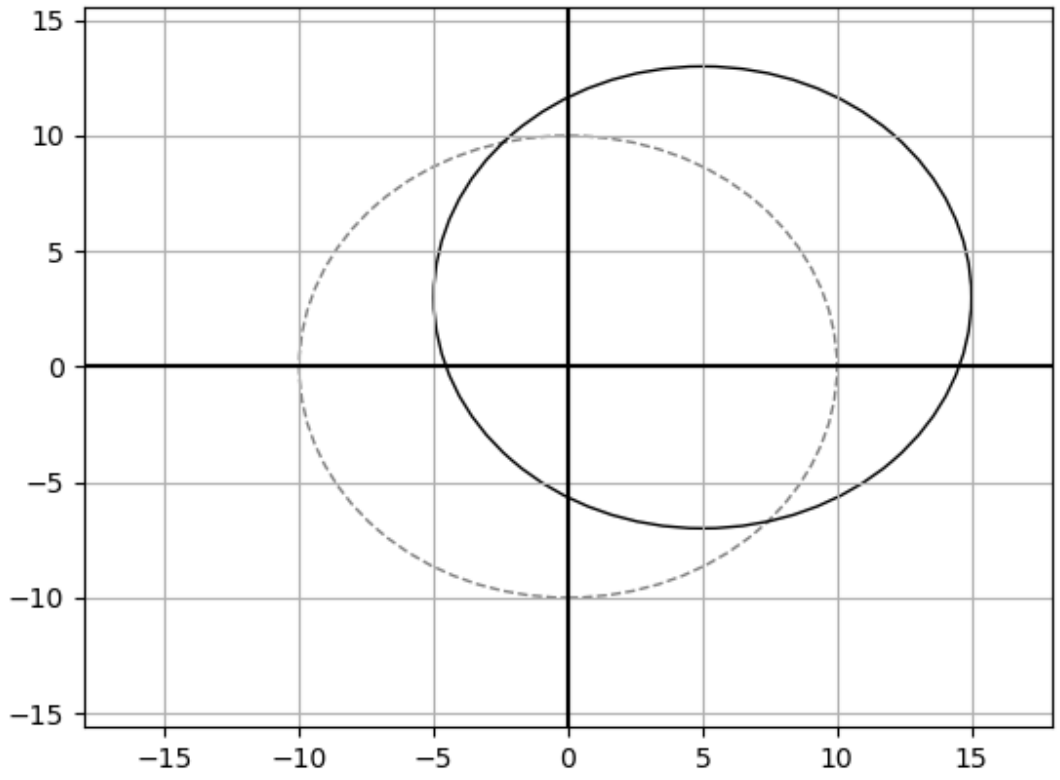
Output:

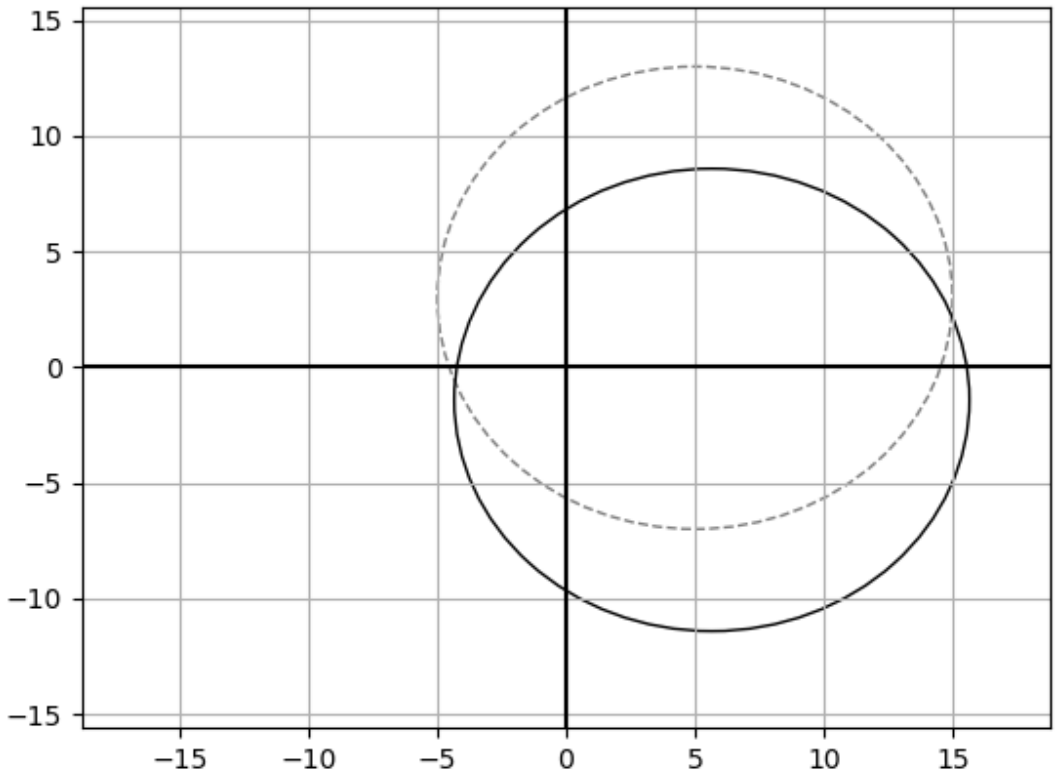

4
2
0
−2
−4
−4
−2
0
2
4


8
6
4
2
0
−2
−4
−6
−8
−10
−5
0
5
10


8
6
4
2
0
−2
−4
−6
−8
−10
−5
0
5
10


10
5
0
−5
−10
−10
−5
0
5
10


15
10
5
0
−5
−10
−15
−15
−10
−5
0
5
10
15


15
10
5
0
−5
−10
−15
−15
−10
−5
0
5
10
15