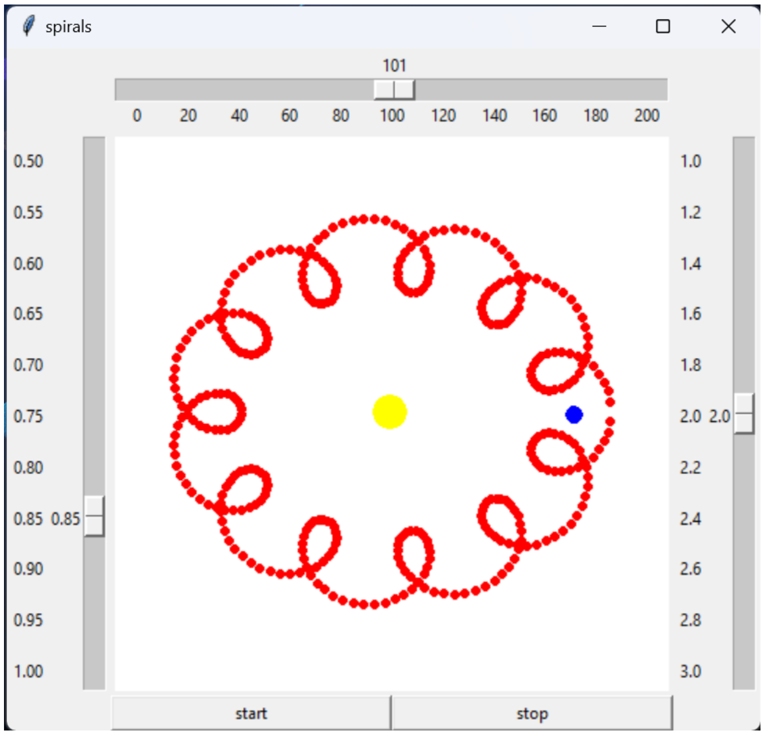
MCS 260 Project Four : Spirals due Wednesday 2 August at 10am

The goal of this project is to write Python scripts to draw spirals as shown below:



The solution is defined by three `.py` files.

1. The first script `earth.py` models the trajectory of the planet orbiting the sun.
2. The second script `moon.py` computes the path of a satellite circling the planet.
3. The third script `spirals.py` plots the trajectories.

The module `earth.py` contains the definition of the class `Earth` and a main program which prints the position of the planet for each day of the year. Given the initial distance to the sun $r > 0$ and eccentricity $0 < e < 1$, the formula for the coordinates (x, y) is

$$(x, y) = (r\cos(\alpha), r\sin(\alpha)e), \quad \alpha \in [0, 2\pi].$$

At day d, the corresponding angle is $\alpha = d2\pi/365$, for d running from 0 to 365. For $r = 10$ and $e = 0.85$, the output of the main program in the module `earth.py` is

```
$ python earth.py
( 10 , 0 ) at day 0
( 9.998518392091162 , 0.14631352732459482 ) at day 1
( 9.994074007397048 , 0.2925836987933388 ) at day 2
...
( 10.0 , -2.0818995585505007e-15 ) at day 365
```

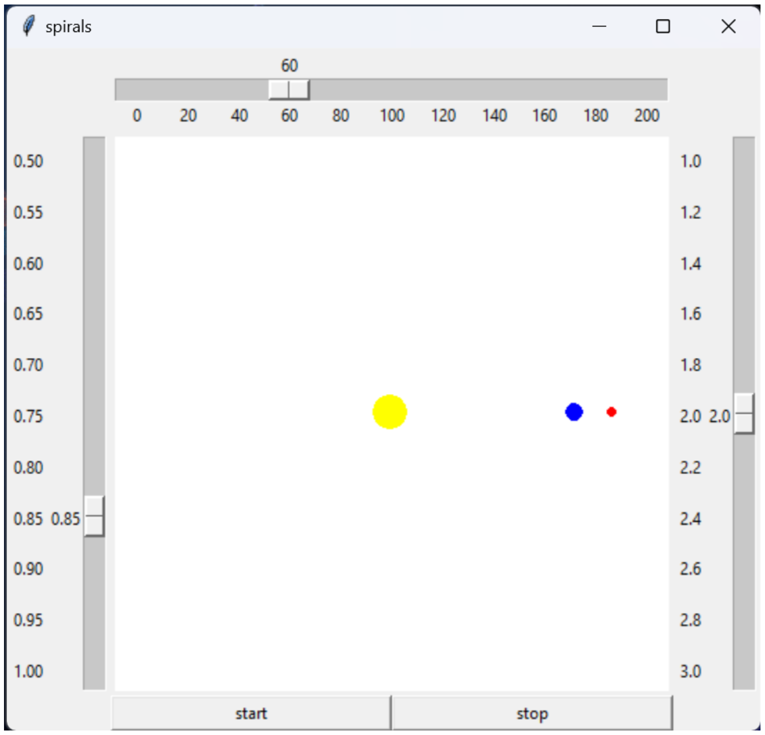
The module `moon.py` imports the class `Earth` from `earth` and computes the spiral path of the satellite circling the orbiting planet, making a complete turn every month, twelve times a year. For regularity, the number of days in each month is $D = 365/12$. Given in (X, Y) the coordinates of the planet, and for the distance R of the moon to earth, the formula for the coordinates (x, y) is

$$(x, y) = (X + R\cos(\beta), Y + R\sin(\beta), \quad \beta \in [0, 2\pi].$$

At day d, the corresponding angle is $\beta = d2\pi/D$, $D = 365/12$, for d running from 0 to 365. For $r = 10$, $e = 0.85$, $R = 3$, the output of the main program in the module `moon.py` is

```
At day 0, earth is at ( 10 , 0 ) and
           moon is at ( 13 , 0 ).
At day 1, earth is at ( 9.998518392091162 , 0.14631352732459482 ) and
           moon is at ( 12.934738631991909 , 0.7616270269304524 ).
...
At day 365, earth is at ( 10.0 , -2.0818995585505007e-15 ) and
             moon is at ( 13.0 , -1.0899356512411444e-14 ).
```

The module `spirals.py` imports the definitions of the classes `Earth` and `Moon`, respectively from the modules `earth` and `moon`. At the launching of `spirals.py`, we see:



The plot on the previous page shows the state just before completing one year.

The characteristics of the GUI are specified below:

1. The title of the window is `spirals`. The screenshot is taken on Windows, where the title does not appear in the center.
2. The top scale controls the refresh rate of canvas. The resolution is 1 and the scale goes from 0 to 100, with tick intervals at 20 with a resolution of 1.
3. The left scale controls the eccentricity e of the orbiting planet, ranging between 0.50 and 1.00 with a resolution of 0.05 and tick interval 0.05 as well.
4. The right scale controls the distance R of the circling satellite to the planet, ranging tween 1.0 and 3.0 with a resolution of 0.01 and tick interval of 0.2.
5. The canvas in the center is square, of dimension 400 pixels. The sun is shown as a yellow circle with 12 pixels as radius, the radius of the circle representing the planet is 6 pixels, and the satellite is represented by a red circle with a radius of 3 pixels.
6. The animation starts when the user clicks on the start button and pauses when the stop button is clicked. The orbit of the planet is not shown, only the blue dot moves. After each year, the entire spiral trajectory of the satellite is wiped from canvas.

Some important points:

1. For this project you may work in pairs. A pair consists of two programmers, not three or more. If you decide to work in a pair, then you must send me an email with the name of your partner and with the email address of your partner in the copy of the email, before 5pm on Thursday 27 July. If working in a pair, then only one solution should be submitted.
2. Do not submit programs that do not run. It is much better to submit an incomplete program that runs than a program that does not run.
3. The terminal output and layout of GUI must be as shown.
4. The first line of your Python programs must be

   ```
   # MCS 260 Project Four by <Author(s)>
   ```

   where you replace the `<Author(s)>` by your names.
5. Classes must be used and every module, class, and method must have a clear documentation string.
6. Upload your solution as files with the `.py` extension into gradescope before 10am on Wednesday 2 August.
7. The late deadline is 5pm on Wednesday 2 August, but solutions that are submitted late are subject to a penalty of 10 points off.

If you have questions or difficulties with the project, please ask for help during office hours.