Homework 5: Plotting with Python

For the following exercises, write your code in one `.py` *file for each problem (or part, as you see fit).*

For parts that require written explanation or analysis, use the `print()` *function to print your answers to the screen when the script is run.*

Make sure that you scripts run without error in order to get credit. Do not hesitate to ask for help if needed!

Problem 1 (10 pts)

Pick three of your favorite (non-trivial) functions on a given domain and use `matplotlib` to plot them. Be sure to mess with at least the following plot attributes:

- Title
- Subplots
- Axis labels
- Curve color and thickness
- Tick markers
- Legend
- Saving the figure to a file on the disk
- Show the figure in the Python viewer

Basically, show me you can use all of the skills developed in these notes. Don't go crazy, but don't just copy what I've done either. Make an effort to fully understand these functions.

Hint: Remember that `matplotlib` doesn't plot continuously defined functions; you have to discretize the domain (perhaps using `numpy.linspace`) and plug that into the continuous function you have chosen (`numpy` arrays are ideal for this). Here is some sample code to get you started (this is only an example!):

```
def f( x ):
  <your function definition>

xdata = np.linspace(0.0,100.0,1001,endpoint=True)
ydata = f(xdata)
plt.plot(xdata,ydata,color='blue', linewidth=1.0, label='myPlot')
```

Problem 2 Projectile motion with flat ground (10 pts)

We want to illustrate the projectile problem with flat ground described in Lecture Notes 4. This should be a fairly straightforward application of the functions discussed above.

Prepare the following graphs, in both cases including the ground as a thick black line.

(a) several curves with different values of v_0 for a fixed value of θ_0;
(b) several curves with different values of θ_0 for a fixed value of v_0.

Make sure to include the values of the parameters v_0 and θ_0 in your graph.

Note: Remember that here we are plotting trajectories, $y(x)$ for fixed values of v_0 and θ_0, not the range, so you can't reuse the function you defined in the corresponding homework (it was only calculating the range).

You should be able to hover your mouse over the plots created in the Python viewer and read off the coordinates of any point. Use this approach to find the range of the projectile in a few cases that you plotted above, and add the range into the curve label in the legend.

Again, some sample code to get you started:

```
def y(x, v0, th0):
  "This function will return the y-coordinate for a given \
   x-coordinate and initial velocity."

  th0rad = th0 * np.pi/180.0 # in radians
  return h0 + np.tan(th0rad) * x
           - (g * x**2)/(2 * v0**2*np.cos(th0rad)**2)

xdata = np.linspace(0.0,100.0,1001,endpoint=True)

v01 = 10 # in meters per second
th01 = 40 # in degrees
ydata1 = traj(xdata, v01, th01)
plt.plot(xdata,ydata1,color='blue', linewidth=1.0, label='traj1')
     # note that your label will need to be more descriptive than this!
```

And then just repeat that for multiple trajectories.

If you want to be fancy, figure out how to define a function for the trajectory so you can just plug in the relevant values and make a list of `v0` and `th0` values and then generate a list of trajectories. Then you can use a loop to plot each one. It is possible to have very clean, succinct code for this problem, but that's optional.

Problem 3 Modification of the ground (10 pts)

In this problem, we want to replace the flat ground from the previous problem with a ground that starts sloping up at angle α some distance d_1 from the launching point, then gets flat again after another distance d_2. Mathematically, this translates to a ground defined as

$$\begin{cases} 0 & \text{for } x < d_1, \\ (x - d_1)\tan(\alpha) & \text{for } d_1 < x < d_1 + d_2, \\ d_2 \tan(\alpha) & \text{for } x > d_1 + d_2 \end{cases} \tag{1}$$

Note: For all graphs in this part and the next, use $d_1 = 6\,\text{m}$, $d_2 = 4\,\text{m}$, and a slope angle $\alpha = 30°$.

This case is a bit more tricky as the ground is represented by different functions for different values of the x coordinate. We can define a `ground` function using `if` statements, but the presence of the statements prevents the function to work for an `array` argument. Since our goal here is just to plot the ground, the easiest way to deal with this is to use a loop to calculate the data needed for the plot command. That is the approach you should use here (unless you want to find a better way as a challenge).

In this problem, we basically want to create a plot similar to that of the previous problem including 3 different trajectories for the rock, one landing on each "part" of the ground (vary both v_0 and θ_0 to get your trajectories).

As in the previous problem, use the graphical window to "read out" the coordinates of the landing point of the rock in each case, and use these values to:

- stop the trajectories at the landing point (*Hint:* try to redefine the array of x positions...)
- add a (visible) black circle at the landing point.
- use the coordinates of the landing points as tick marks and labels on the vertical axis (labels should have only one digit after the decimal point).
- add a label at each landing point indicating the range, as in $R = 4.5$ m (*Hint:* Try the `annotate` method).

Exploration

Now pick a fixed value of θ_0 and **find graphically** the minimum value of v_0 for which you can reach the "higher ground."

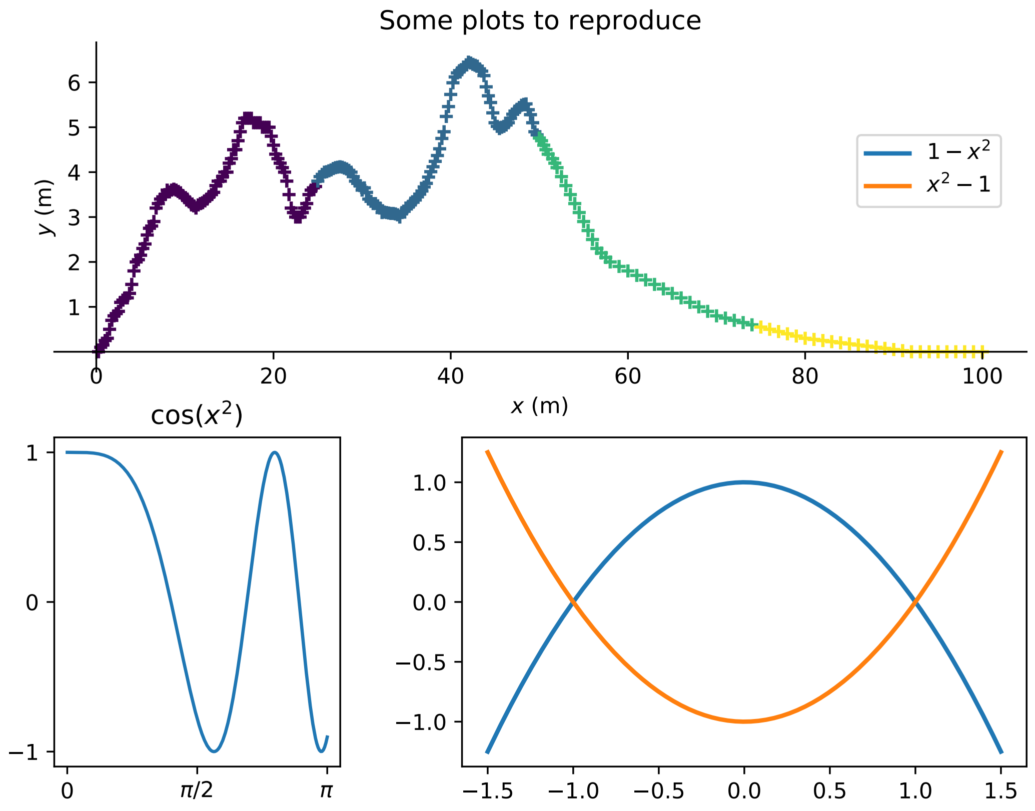
Create a plot with this "critical" trajectory drawn as a thick, red line, and add a couple other trajectories that land below and above the plateau's edge.

Problem 4 (10 pts)

Reproduce the graph below as exactly as you can.

The data from the top panel comes from the attached file `'ground.dat'`. The file contains three columns of numbers: `x position`, `color code`, `ground height`. The first and third columns correspond, respectively, to the x and y axes of the plot, while the `color code` is used to set the colors of the various points.

Hint: Look up documentation for the `scatter` method on how to set the different colors.


Some plots to reproduce
y (m)
x (m)
$1-x^2$
x^2-1
$\cos(x^2)$
π/2
π