Scientific Computing: Basic Methods
J. Bürki, M. Block and R. Barniol Duran

Assignment 2: Variables, Numpy module, Functions

General instructions

- Be sure to use a plain text editor (i.e., NOT Word) when writing your code. If using `replit`, that is already the case; on Windows, Notepad++ is a good choice, while Xcode is good on Macs.
- Follow instructions in each problems on what files need to be submitted and how they should be named so the Autograder can recognize them.
- For parts that require written explanation, 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!
- Check the output of the Autograder for any issue that should be fixed. In case of "Unexpected error", email me so I can take a look as it may be an issue with the Autograder rather than your code.
- Take ownership of your learning! Remember that you are responsible for the work you turn in. Simply copying somebody else's answers, copying from the Internet, using AI to generate your code, sharing your code (or part of your code) in any way, or copying it from someone else will be considered academic dishonesty. Please, contact me if you have any questions about collaborations.

Problem 1

Modify your code from assignment 1 to use things we have learned in the set of notes:

- Use variables to store the data for each of the curves and use those in the `plot` commands to make them more readable;
- Modify the sine function to it covers at least two periods and is smooth;
- Add another curve plotting the cosine function using the same set of x-coordinates as for the sine function, but using a different color and line style; also try to make the line thicker and make sure to add a label;
- Define a function $f(x)$ of a single variable that calculates $\frac{1}{\sqrt{2}}\left(\sin(x)+\cos(x)\right)$;
- Add the function $f(x)$ to plot, again with a different color and line style, adding a label.

What to submit: For this problem, submit your makefile, renamed as `makefile1`, the python script, and the created figure; use the naming convention `yourName_hw2_p1`.

Problem 2

You've learned in your classical mechanics course that the trajectory of a projectile is given by the time-dependent equations

$$x(t) = x_0 + v_0 t \cos\theta_0 \tag{1}$$

$$y(t) = y_0 + v_0 t \sin\theta_0 - \frac{1}{2} g t^2 \tag{2}$$

where (x_0, y_0) is the starting position, v_0 is the starting speed, g the acceleration of gravity and θ_0 defines the initial direction of the projectile above the horizontal. You've also learned that the range of a projectile thrown from the ground is

$$R(v_0, \theta_0) = \frac{v_0^2}{g}\sin(2\theta_0), \tag{3}$$

which is maximum when $\theta_0 = 45°$.

In this problem, we'll want to graphically check the accuracy of the latter formula and its limitations, by creating two plots:

(a) Plot A will show trajectories for a projectile thrown from the floor ($y_0 = 0$) at various angles around 45°, as well as a black dot at the landing point give by the range equation (3).

(b) Plot B will be similar to plot A, but with the projectile thrown from a finite height $y_0 = 5\,\mathrm{m}$. Again, plot a dot, this time of the same color as the corresponding trajectory, at the landing point given by the range equation.
Answer the following questions using `print` statements in your code: Why is this not the actual landing point? Is the range still maximal for $\theta_0 = 45°$ in this case?

A few things to keep in mind while writing your code:

- Set the values of the initial position, speed, and acceleration of gravity through variables at the beginning of your code. Use `x0, y0, v0` and `g` as variable names and use the values $v_0 = 20\,\mathrm{m/s}$ and $g = 9.8\,\mathrm{m/s}^2$.
- Define functions of the variables t and θ_0 for the coordinate functions $x(t), y(t)$ given by equations (1) and (2). Call them `Xpos` and `Ypos` respectively. Also define a function `range` for the range equation (3) that has a single input variable θ_0.
Important note: Trigonometric functions in `Python` (and most languages) expect an angle in radians, but we'd like our functions to take an angle in degrees. Make sure to perform the appropriate angle conversion.
- Time is the independent variable in the trajectory equations (1) and (2). How can we generate the appropriate coordinate arrays to use in the `plot` commands?
- Make sure the various curves in each plot have different colors and line styles, and to include a legend identifying the starting angle for each curve.
- Use appropriate commands to restrict the plotting range appropriately (for example, we don't care that mathematically, the trajectories can continue after hitting the ground).
- The dots at the landing points might be difficult to distinguish. Use the figure tools to zoom in and save the zoomed figure (include `zoom` in the file name)

What to submit: For this problem, submit your makefile, renamed as `makefile2`, the python script, the two figures created by the script and the zoomed-in figures; use the naming convention `yourName_hw2-p2` and make sure the figure files contain, respectively, the string `figA` and `figB`.