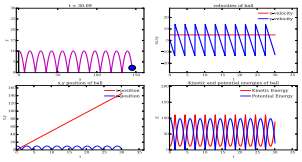
Reach for the Stars



# An Introduction to Numerical Modeling with Python

Adam Dempsey

## Purpose

The goal of this activity is to introduce high school physics students to modeling simple physical systems on their computers. The numerical modeling is done in the Python programming language, which offers both a large library of scientific and mathematical functions as well as a comprehensive learning network (*e.g* online documentation, forums, and interactive lessons). Python is also a highly versatile language that is used in many disparate disciplines ranging from scientific computing, system administration, website and software development, and statistical analysis. Learning Python as early as possible gives students a significant advantage when applying for colleges, undergraduate (or even high school) research positions, and ultimately jobs.

This activity was originally given as part of a series of lessons found at adamdempsey90.github.io. These activities serve as an intermediary between lessons focused on learning syntax, *e.g*, Codecademy and self-directed numerical modeling problems. The emphasis is a balance between learning how to translate physical problems into code and learning new Python functions and structures.

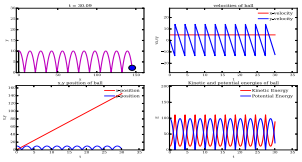
The first lesson walks students through how to simulate a bouncing ball. They are given Python code that is mostly complete and are asked to fill in several missing code fragments. The second lesson is structured the same way, but simulates the more interesting double pendulum. For the bouncing ball activity, students fill in the blanks for a simple energy conserving numerical integrator. The students will then run a simulation of a ball at some height being launched with some velocity. If energy is not conserved or if the ball is allowed to deform, then the ball will not bounce to exactly the same height. Providing students with mostly working code allows them to focus more on the simulation and the physics instead of the coding. However, the coding requirements should sitll pose a challenge to students who have had little exposure to programming. Classes with more programming experience can sacrifice the physics for more computer science by filling in larger parts of the overall program.

## Overview

1. Spend 15 minutes reiterating/presenting the introduction section of the activity to the class. Depending on the level of the students, it may be worthwhile to get into the derivation of the update equations.
2. Have students go through all sections of the website.
    - The activity is meant to be as self-guided as possible. The teacher should take this time to walk around the class and help students who are stuck and/or go through difficult sections as a class. This will depend on the programming level of the class.
3. Students that finish early should attempt some of the challenges found in the more advanced material section.

## Student Outcomes

- SWBAT run a program in Python
- SWBAT translate the equations of motion for a ball into Python code.



- SWBAT use a computational model of a bouncing ball to simulate balls of different elasticity.

## Standards Addressed

- Science and Engineering Practices
  - Developing and Using Models
  - Using Mathematics and Computational Thinking
  - Constructing Explanations and Designing Solutions
  - Analyzing and Interpreting Data
- Disciplinary Core Ideas
  - PS2.A: Forces and Motion
  - PS2.B: Types of Interactions
  - PS3.A: Definitions of Energy
  - PS3.B: Conservation of Energy and Energy Transfer
  - PS3.C: Relationship Between Energy and Forces
  - ETS1.B: Developing Possible Solutions
- Crosscutting Concepts
  - Cause and Effect
  - Energy and Matter
  - Systems and system models
  - Structure and Function

## Time

- The lesson was designed for a 1.5 hour class. Some students may need more or less time.

## Level

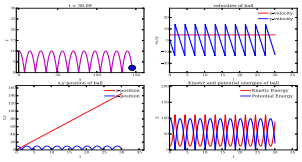
- This activity (and other activities in this series) are geared toward junior and senior level physics/computer science students.

## Materials and Tools

- A computer (or laptop) for every student (or small groups of students). Any type of operating system will work (Windows, Mac OSX, or Linux). However, *The lesson will not work on tablets, mobile devices, or chromebooks.*
- Have students (or the IT department) install a Python distribution, either Anaconda or Canopy, on the machines. Note that the lessons are written assuming the students are using the Canopy distribution.

## Preparation

- The instructor should check to make sure the Python installation was successful on all of the computers.
- The instructor should also be comfortable enough with the activity to help students who are stuck.



## Prerequisites

- Students (and the teacher) should complete the first few units of Codecademy. This can be done as homework or, ideally, in the classroom. The teacher can also use other resources for learning the Python language.

## Background

- Students should be familiar with two-dimensional kinematics and kinetic/potential energy
- Students should also have some basic knowledge of the Python language.

## Teaching Notes

- As this is a self-guided activity, the teacher should mostly be helping students who are stuck and introducing the physics of the bouncing ball at the beginning of the class.
- It is important that the students have some exposure prior to Python before starting this activity. The teacher should be sure that students have had time to go through the first few units of Codecademy or a similar learning device.

## Assessment

- If the students successfully completed the activity, they should have animations of the ball bouncing. These can be checked by the instructor against the examples given on the website.
- Students can also submit their completed Python scripts to be checked against the instructor's completed (and working) script.

## Additional Information

- The follow up activity on the double pendulum may be more exciting for students as it demonstrates the concept of chaos. This activity can be given either before, after, or in place of the bouncing ball activity described here.