Using Python to Calculate Gravitational Force

Zachary Hafen

Purpose

Computational sciences use computers to help answer questions about our universe. One of the most important steps in computational science is the implementation of mathematical in a computer language of choice. This lesson covers one such example of that: the calculation of the force due to gravity between two masses, implemented in Python. While the example is narrow, the steps contained within are generalizable to any analytic formula, in any field.

Overview

Students will work individually or in groups on coding and testing a function in Python that calculates the force due to gravity between two masses. Students will first create and edit a new Python file. In that file they will create a function to calculate the force between two objects. Next, they will check that their function works, by downloading and using a Python script provided as part of this lesson. Finally, they will make sure their function is well-documented.

Student Outcomes

Students will be able to:

- Create a Python function that implements a mathematical formula
- Demonstrate increased familiarity with the following Python concepts: functions, python modules, arrays/lists, the math module or the numpy module, and plotting in Python
- Download and include external Python modules
- Assess the accuracy of a function using plotting

Standards Addressed

NGSS Practices: "Analyzing and Interpreting Data", "Using Mathematics and Computational Thinking", and "Obtaining, Evaluating, and Communicating Information".
NGSS Standards:
HS-PS2-4 Motion and Stability: Forces and Interactions, i.e. use mathematical representations of Newton's Law of Gravitation and Coulomb's Law to describe and predict the gravitational and electrostatic forces between objects.
HS-ESS1-4 Earth's Place in the Universe, i.e. using computational representations to predict the motion of orbiting objects, which relies on the implementation of mathematical formula, as described here.
HS-ETS1-4 Engineering Design, i.e. using a computer simulation to model real world systems, one part of which is checking the robustness of your model by comparing to your expectations.

Time

Two class periods: one primarily for writing the code, and one for checking and finishing up.

Reach for the Stars is a GK-12 program supported by the National Science Foundation under grant DGE-0948017. However, any opinions, findings, conclusions, and/or recommendations are those of the investigators and do not necessarily reflect the views of the Foundation.

Level
11th Grade Science or Computer Science, especially classes with students that have both some programming and physics experience.

Materials and Tools
One computer per student or group of students.
- The computer must have Python installed.
- The student must be able to download the following Python module: https://northwestern.box.com/s/3ex51t49jfkmb0eb28doii1c2gb4qj7q

Preparation
The teacher should make sure that each student (or each group of students) has access to a computer that can run and edit Python files.

Prerequisites
Students should be familiar with Python prior to this lesson, especially the following concepts: functions, python modules, arrays/lists, the math module or the numpy module, and plotting in Python. Students should also be familiar with forces, especially with the force of gravity. Students should be able to calculate the two-dimensional force due to gravity between two objects by hand.

Background
One of the main strengths of computational sciences is the ability to take work that would have impossibly long by hand, and to automate it easily. A crucial step along the way is to take the human-readable mathematical formula, and make it also readable by the machine. This lesson covers exactly that: translating mathematical formula into a programming language, Python.

Teaching Notes
Give each student a copy of this document:
https://northwestern.box.com/s/lvft5k93qttuq0h56g9ar45xrnspcf2p

The goal of the lesson is to get, from each student or group of students, a well-commented function that calculates the force due to gravity between two masses, along with an accompanying plot showing that it works.

There are two primary stages the students will go through, so be prepared to help them through both:
1. Writing the function
2. Making sure the function it works.

This is a largely freeform lesson. The students will need to stumble their way to the answer. This may be challenging for them, but if they succeed they will understand what they did better.

To start off, with the first step request the students to create a function that calculates the force between any two objects with arbitrary mass and position in the x-y plane. The above document should provide sufficient tips for them to get started. Walk around the room and help them when they get stuck.

When the students get to the second step, the step is done primarily through running the Python module n_body_checks.py, included above, in the materials section. This module will take in the function that the students wrote, and if the student function is correct it will make an accurate plot of the force on one of the masses due to the other mass.

The section "Advanced checks" is optional, at the teacher's discretion.

Common stumbling blocks:

1. When the student tries to use n_body_checks.py, it *needs to be in the same folder* as the file the student is writing.
2. The student may use something like "theta = math.atan(dy / dx)" to calculate the angle the force is at. This is fine for forces in the first and third quadrant, but will be incorrect for other quadrants. One way around this is to calculate the force of gravity in the form $\vec{F} = -\frac{Gm_1m_2}{r^3}\vec{r}$

Assessment

The easiest way to check if the students succeeded is to look at the plots they hand in. At a glance, the teacher should be able to see if it looks like a reasonable result. This is especially true if the student completed the "Advanced checks" section, and turned in the plot from that. As an additional assessment, the teacher can glance through the code, and see if it's well-organized and documented.