# Using Python as a toolbox for student-directed virtual labs

Cite as: Phys. Teach. **57**, 60 (2019); https://doi.org/10.1119/1.5084937
Published Online: 21 December 2018

James Lincoln

**James Lincoln,** Column Editor,
PhysicsVideos.com, Newport Beach, CA 92658

# Using Python as a toolbox for student-directed virtual labs

***James Lincoln,*** Southern California AAPT, Newport Beach, CA 92658; james@physicsvideos.com

Python is a free, text-based programming language that has already been used by those who do physics research to great effect. But students can also learn to use it and, through its use, learn other aspects of physics—especially ones that are difficult or impossible to perform as physical labs. This article serves as a most-fundamental start for teachers who are interested in how they can use Python to enhance their physics instruction. Although there have been a few articles that have made references to using Python, none so far in this journal has described how quickly and effectively it can be used to simulate a virtual lab experience or aid in creative design. In this first of two articles, I introduce Python (and VPython) and provide some sample codes to guide beginners.

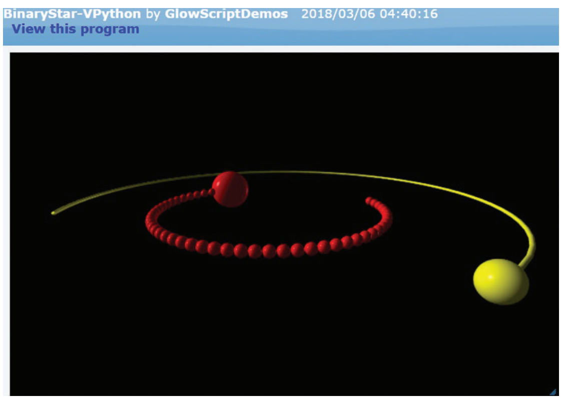


**Fig. 1. A screenshot of the binary star simulation from VPython.org. These programs are easy to make within your first-class period if you take the toolbox approach.**

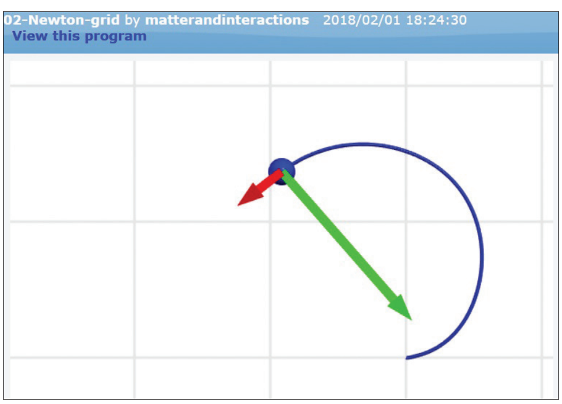


**Fig. 2. The author attempts to make a circle using a sim from VPython.org. The green force vector needs to be kept at a right angle to the red velocity vector.**

## What is Python?

Python is a computer programming language that is easy to read and write. It is free to download online from Python.org. There is a lot of help online[1] and there are many books[2] that can take you much further than this article. However, I advise that the fastest way to learn a coding language is to read and make sense of another author's code. When it comes to physics you will want to start by exploring the VPython resources (V for Visual), Fig. 1, which are found at VPython.org. The full potential for physics demonstrations using VPython seems to have been realized and leveraged by the authors of the *Matter and Interactions* text; see their many simulations on this site.[3]

Using Python to demonstrate a physics concept does not have to mean learning to write code from first principles. Rather we take pieces of code as needed, search for the objects as if from a toolbox, to program into a virtual environment and from those pieces manipulate given variables and perform virtual experiments, Fig, 2. With this toolbox approach, a lot of physics will be accessed rapidly even by an absolute novice coder.

## Getting started

I recommend using Python as if it is another way to perform labs. The coded collection of "physics equipment" you might want to borrow from can get you and your students excited to try out coding for the first time. The first step is to download the program and practice some sample code.

After installing the program, you will be using an IDLE screen, Fig. 3, in which you type or paste your coded instructions. When you want to see what your program is doing, you can "run" it and another window will pop up. If you can get over the challenge of

Bouncing Ball.py - C:\Users\James Lincoln\Desktop...

File Edit Format Run Options Windows Help

```
#SETTING THE STAGE
import turtle
wn = turtle.Screen()
wn.bgcolor("black")
wn.title("Bouncing Ball")
wn.setup(1000, 1000)
#DRAW THE FLOOR
marker = turtle.Turtle()
marker.color("red")
marker.penup()
marker.goto(-400, -300)
marker.pendown()
marker.forward(800)
marker.right(90)
marker.forward(20)
marker.goto(-400, -320)
marker.color("black")
#INVENT THE BALL
ball = turtle.Turtle()
ball.shape("circle")
ball.color("blue")
ball.penup()
ball.speed(0)
ball.goto(-400, 200)
ball.pendown()
#INITIAL SPEED
ball.dx = 2
ball.dy = 0
#TURN ON GRAVITY
gravity = 0.1
#CYCLE THE TIME DURATION
while True:
    ball.dy -= gravity
    ball.sety(ball.ycor() + ball.dy)
    ball.setx(ball.xcor() + ball.dx)
    #BOUNCE AND RESTITUTION
    if ball.ycor() < -300:
        ball.dy *= -.8
```

**Fig. 3. The segment of Python code I used to animate the bouncing ball. The "turtle" is a cursor that can be programmed to move, draw, and change shape.**

 DOI: 10.1119/1.5084937

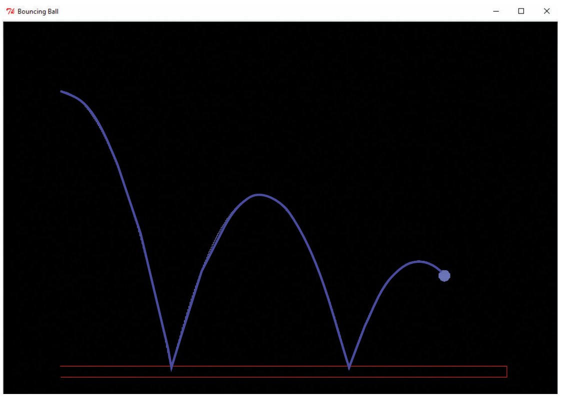


**Fig. 4. In this animation, the ball is losing 36% of its energy with each bounce. The "pen down" function allows for the trace.**

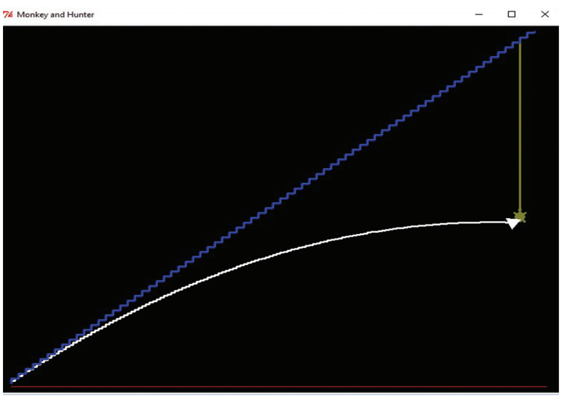


**Fig. 5. The Monkey and the Hunter – perhaps the best physics demo – can be recreated in Python. But now, we can adjust gravity! How much better will students understand the purpose of this demo if they take the time to create it themselves from scratch?**

understanding a falling projectile, then you will be able to learn how to code more complicated systems. For example, you might want to design a model solar system, demonstrate Kepler's laws, or perhaps see whether a binary star system can be stable enough to support planets.

As an illustrative example, I present a bouncing ball that loses 20% of its velocity in each bounce, Fig. 4. In the code, I have included many #HINTS that tell you what I was doing in each step. From this fundamental beginning it is possible to invent other scenarios easily such as the Monkey and the Hunter, Fig. 5, and depict objects falling under the influence of air resistance (either $F = -bv$ or $F = -Bv^2$ forces, etc) and even relativistic behavior. The code for this is included online as an appendix.[4]

## Conclusion

Python might be the most appropriate way to introduce programming into your physics class while still teaching physics. With many of us being asked to teach more engineering, but not necessarily being experienced in that field, programming physics can serve as a worthy answer. Some instructors make a good case[5] that having students program is a highly effective way to teach and one of the most effective ways to experiment with the laws of physics.

## References


1. My code is here is partially inspired by this YouTube video series by Christian Thompson on animating a bouncing ball: https://www.youtube.com/watch?v=HHQV3ifJopo.
2. John R. Leeman, "A student's guide to Python for physical modeling," *Am. J. Phys.* **85**, 399 (May 2017).
3. These simulations are found on the VPython.org website. As for the text, see Ruth Chabay and Bruce Sherwood, *Matter and Interactions,* 4th ed. (Wiley, 2014).
4. See the appendix at *TPT Online,* http://dx.doi.org/10.1119/1.5084937 under the Supplement tab.
5. Ruth Chabay and Bruce Sherwood, "Computational physics in the introductory calculus-based course," *Am. J. Phys.* **76**, 307 (April 2008).