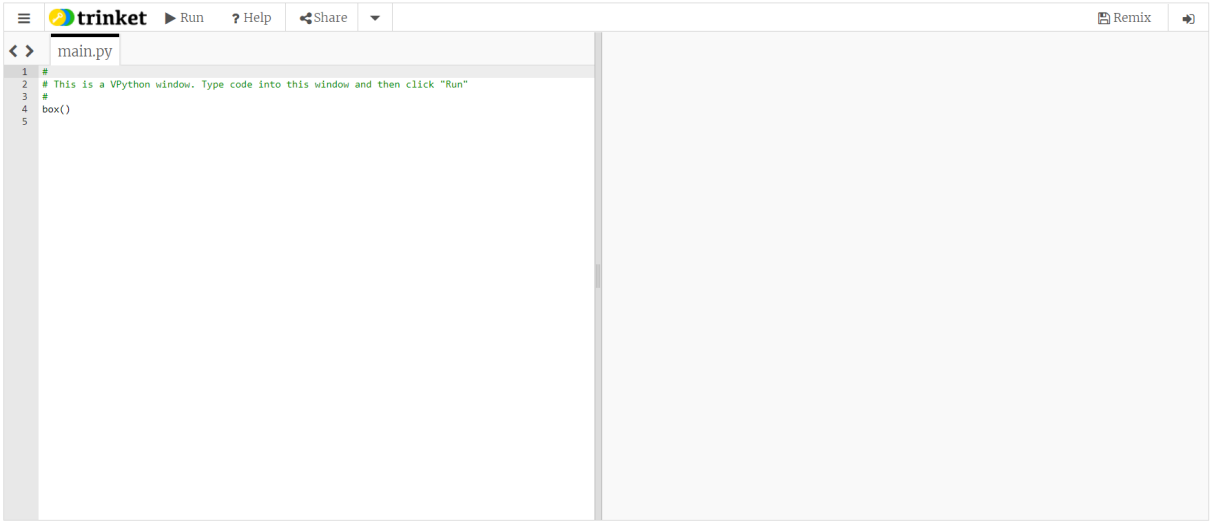
Session 1: Trinket.io

1) Empty Screen

First, we'll start with the empty screen at http://3d.spvi.net



2) Objects

Then we'll make some objects:
Then let's add a "ground" in a naive way:
I'll point out that the ball is "inside" the ground. Oh well. ;-) Let's go into breakout rooms to give the troops a chance to explore a bit. Have the students play around and get used to the options for various shapes. Note that the "hamburger" menu gives you more options, including lots of "help."
The Glowscript "help" link here takes you to a Trinket Screen:
But this screen has a link to "Glowscript Help" that gives lots of details of many GlowScript features and options (See below).

3) Variables and Attributes

After they've had a chance to play around with various shapes, we'll come back to the main room. We will talk about variables and attributes. Probably by now, some will have figured out how to fix the "ball in the ground" problem. There are two ways. First, you have to notice that the ball's radius is 0.2m, and the ground's height is 0.1m. The

ball's "position" (pos) vector is at the center of the ball. So the get the ball to just rest on the surface of the ground, its "y" position needs to be 0.25m:
You can do this by adding a "pos" attribute to the balls constructor:
Or you can modify the ball's "pos" attribute after you've created the ball.
Notice that "ball" is a variable, and "pos" is an attribute of that variable. It turns out that "pos" also has attributes! They are "x," "y," and "z." Try changing the "x," "y," and "z" attributes of the ball's "pos" attribute.
Now let's go back into break-out rooms and experiment with creating objects and modifying their attributes. Try changing position, shape, color, etc., and see what you can make!

4) Loops:

Next, we'll play with loops.
"For" loop: A straightforward kind of loop is a "for" loop. It executes the same code repeatedly, using a different value for one of its variables each time. Consider the following loop:
Note a few things:
• the rate() function keeps things from happening too quickly to see. rate(1) means the loop executes only one time per second.
• The "print()" function lets us print out values of variables while the loop executes.
• The "for" loop executes the code in the indented portion (called the scope or body) multiple times. Each time the variable "x" has a different value taken from the list of numbers in the loop's first line.
• The loop ends when there's a new line of code that's not indented.
"While" loop: Another common type of loop is a "while" loop. This loop does the same thing, but it uses a condition and an update instead:
Now, let's go back into breakout rooms and let the students experiment with simple loops. Encourage them to try "for" loops and "while" loops. If they want to see a "smoother" motion, they can reduce the size of the changes and increase the number they pass into the "rate" function (the higher the number, the faster the code runs).

5) More conditions

The "While" loop keeps running until its "condition" becomes "false." For the while loop above,
the condition is "x<4". So long as x stays less than 4, the loop keeps going. However, the loop
also has a statement "x = x + 1" which increases the value of x each time the loop executes.

The means the loop will eventually terminate (when x becomes equal to four, in this case). You
can use conditions to set bounds on a variable. Consider this loop:
What happens when x goes outside the range in between -4 and +4? What happens if you
change v0? What happens if you change the argument of the rate function? Experiment! Try
working with different shapes and modifying other attributes.

6) Gravity

Finally, we're going to add a changing velocity. A uniformly changing velocity is typical, for example, when you throw a ball in the air. Not only does the ball have a velocity, but the Earth
also pulls on the ball, and the velocity changes over time. Let's see how that might work.

Two important points here:
1) v = v + g*dt # v is *changing* with every loop
2) ball.pos = ball.pos * v*dt # the ball's position updates every loop
You can think of the velocity as the "current motion," and "g" is the answer to the question "How is the velocity changing with time?".

What's the problem with this code? How can we fix it? Use what you've learned and seen to
try to make the ball "bounce" correctly. Let's go into breakout rooms and make that happen.
When we get out, we'll go over the possibilities.
(Note one solution is posted here , actual code is below)