Golf- Point Slope Form

In this lesson, you will create a golf game using the Python Editor. Your game code will randomly generate a location for the golf ball and the hole. To take a shot, the user will enter a linear equation in point-slope form that models the path for the shot. The program will plot the golf ball, hole and shot.

Objectives:

Programming Objectives:

- Use the input function and a variable to collect and store data from a user
- Use the randint() function to generate random integers.
- Use the plot library to plot points
- Use the plot library to plot a line segment

Math Objectives:

- Use the point slope form of a line to plot a line segment

Math Course Connections: This activity is recommended for Pre-Algebra or Algebra 1.

In this project, you will create a golf game. The golf game gives the user two random points, one for the golf ball and one for the golf hole. The user will enter a value for the slope. The program will plot the equation of the line through the golf ball x1,y1 using the slope.

Correct Shot
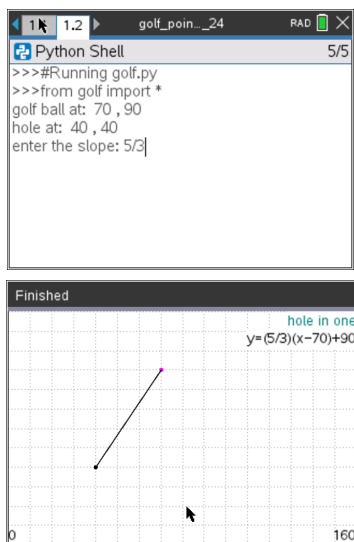
Incorrect Shot
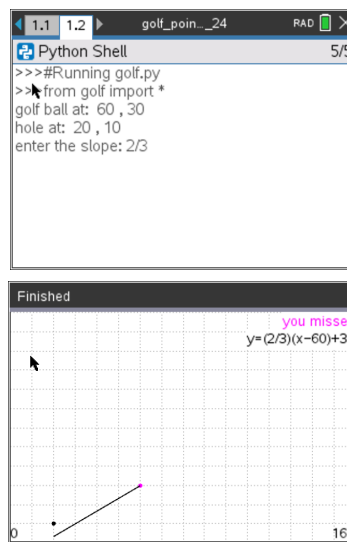
1. To create the golfing game, you will need to plot two points, the golf ball and the hole.
    Brainstorm the information you need plot the ball and hole.

    1.)

    2.)

    3.)

    4.)

    5.)

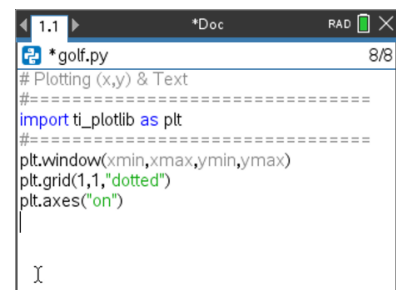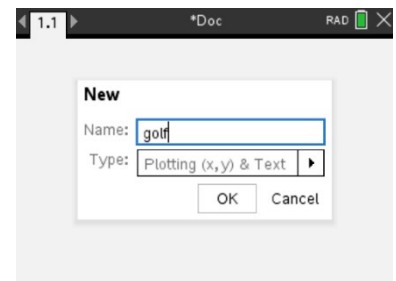2. If possible, compare your list to a partner's list.

   What do you have in common?

   Are there items you forgot?

3. The first step will be to create a python graphics document.

   Create a new python project named "golf".

   Select "Plotting (x,y) & Text" from the type menu.

4. You will need one more library, the random library.

   The random library contains the randint() function, which you will use to generate random integer coordinates.

   Place your cursor on the line below the **import ti_plotlib**
   Menu> Random> from random import *

5. Look back over your brainstormed list from step 1 and 2.
   Did you recommend drawing the coordinate grid?

   Before you can plot the points, you need to set up the graphing window.
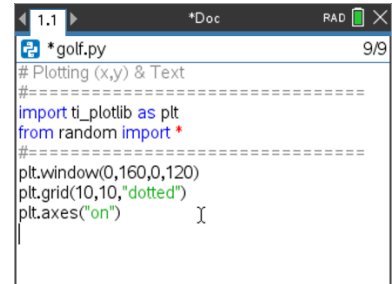
   For this game, let the domain be [0,160] and the range [0,120].

   Set the plt.window values for the xmin, xmax, ymin, and ymax.

   The domain and range have a large range of values; change the grid to mark every 10 spaces.

   After setting the window and the grid scale, press [ctrl] → [r] to run the program.
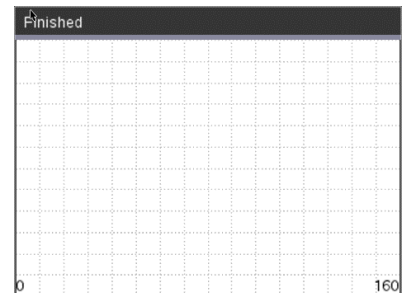
   Verify your graph matches the one on the right.
   Press **[esc]** to exit the graph screen.

6. You need an x value and a y value to plot a golf ball location.
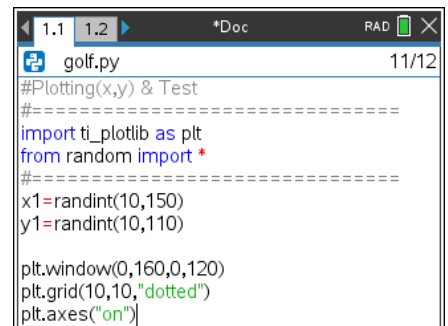   To keep the point on the page, generate random x1 and y1 values within the domain and range.
       ABOVE the window set up, add the two lines:
               x1 = randint(10,150)
               y1 = randint(10,110)
       **randint can be found under the menu.
               Menu> Random> randint

7. Add a print line to print the golf ball location to the screen.
       Menu> Built-ins> I/O> print

   print("golf ball at: ", x1, ",", y1)

   Notice the string "golf ball at: " and "," are in green.
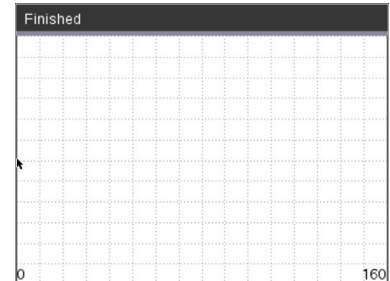   All string values are displayed in green

   Execute your code: [ctrl] → [r]

The empty graph will display.

Press [esc] and the console should show with the printed x1,y1 value

Execute the code a few more times. Verify the golf ball is placed at a different place each time.

Teacher Tip: This lesson uses double quotes for string values. Python allows for both double and single quotes. However, if a "string" starts with double quotes it must end with double quotes. Likewise, if it starts with a single quote is must end with a single quote.

8. You may choose to keep x1 and y1 as any integer between [10,150] and [10,110] or you may choose to make them multiples of 10 for easier linear calculations when making your golf swing.

    If you modify your code to

        x1 = randint(1,15)*10
        y1 = randint(1,11)*10

    your golf ball coordinates will always be multiples of 10.

9. Add in three lines of code to generate x2, y2 and print the location of the golf hole.

10. The user will enter the slope as a fraction. To store the fraction as an evaluated float value, use the eval function with the input function.
    Menu> Built-ins> I/O> eval
    Menu> Built-ins> I/O> input

    m = eval(input("enter the slope: "))

11. It is possible to generate the same coordinates for both the ball and the hole. To fix this, we will add the code:
    if x1==x2:
      x1+=10

    **Menu → Built-ins → Control → If

12. Execute the code [ctrl] → [r]

    Notice the display order has changed.

    The program first displays the coordinates for the golf ball and the hole. It also asks the user to input the slope.

    After the user enters a value for the slope, the window is displayed.

    The lines of code are executed sequentially, meaning in the order they appear on the screen. Before you added the input line, the code printed the two print statements. Then immediately displayed the graph window before you could see the printed statements.
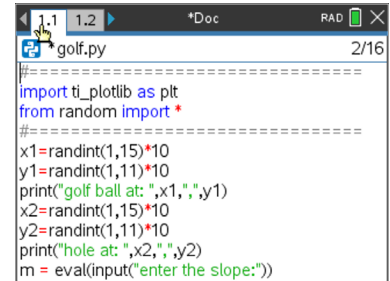
    The input requires the user to submit a value before the program executes the plotting code.

13. Your program generates the coordinates for the golf ball and the hole. It displays these values and asks for the slope between the two points. Finally, it displays an empty graph.

    All you have left is to plot the points and graph the line.

14. Scroll to the end of your code

You can plot a point in any color imaginable.
The command from the plot library is color(red, green, blue).

The values for each color parameter can be any integer from 0-255.

Example colors:

red	color(255,0,0)
green	color(0,255,0)
magenta	color(255, 0, 255)
plum	color(221,160,221)

The color function can be found using:

Menu> TI-PlotLib> Draw> color

Choose any values you like for red, green and blue as long as they are integers between 0-255.

```
y2=randint(1,11)*10
if x1==x2:
   x1+=10
print("hole at:",x2,",",y2)
m = eval(input("enter the slope:"))

plt.window(0,160,0,120)
plt.grid(10,10,"dotted")
plt.axes("on")

plt.color(255,0,255)
```

15. To plot the point, use the plt.plot (x, y, mark) command.
    Menu> TI-PlotLib> Draw> plot

Use your random integer variables x1 and y1 for the coordinates.

plt.plot(x1, y1, "o")

```
if x1==x2:
   x1+=10
print("hole at:",x2,",",y2)
m = eval(input("enter the slope:"))

plt.window(0,160,0,120)
plt.grid(10,10,"dotted")
plt.axes("on")

plt.color(255,0,255)
plt.plot(x1,y1,"o")
```
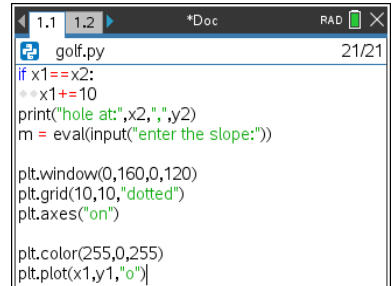
Execute the code a few times.   [ctrl] → r
Use [esc] to exit the graph page.

Does the code generate a new golf ball location each time?
Are you satisfied with the color of your golf ball?

If not, experiment with various values for red, green and blue.

16. Repeat the last two lines of code.

    Change the color to any color you like.
    Plot the coordinate x2,y2 to represent the golf hole.

    Execute your code. Verify the golf ball and the golf hole appear at different locations and have different colors.
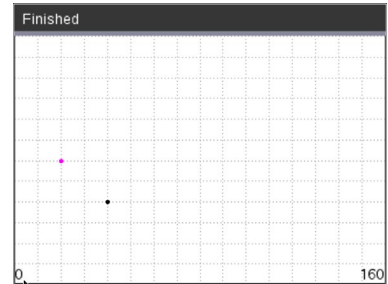
17. The last step is to graph an equation using the user's slope, the golf ball coordinate x1,y1 and the hole's x2 value.

    Recall, $y = m(x - x1) + y1$.

    Use m, x2, x1, and y1 to calculate the user's y value called uy

    $uy = m*(x2 - x1) + y1$

    Plot the line.

    The command is plt.line(x1, y1, x2, y2, "mode")
        Menu> TI-PlotLib> Draw> line
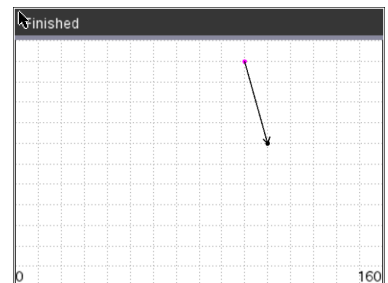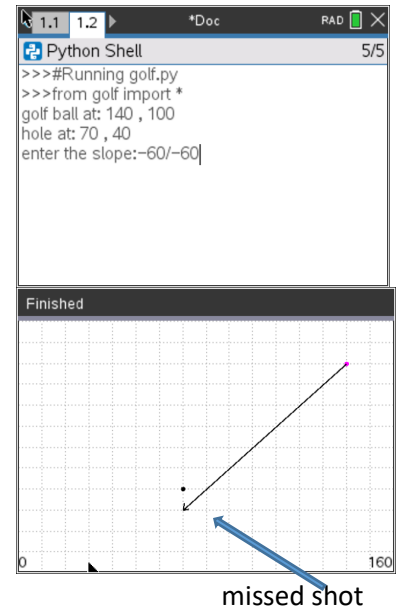
    plt.line(x1, y1, x2, uy, "arrow")

18. Try your game out.

    If you enter the correct slope, do you make the shot?
    The example at the top is an example of a correct shot.

If you enter the incorrect slope, do you miss the shot?
The example at the bottom is an example of an incorrect shot

missed shot

19. Shouldn't the program tell the user if the shot was a "hole in one" or a "miss"?

First, change the pen color.
Menu> TI PlotLib> Draw> color(red, blue, green)

Choose integer values between 1 and 255 for red, blue and green.
Examples: Teal 0, 128, 128
Magenta 255, 0, 255
Blue 0, 255, 0

20. Now check to see if the calculated value uy equals y2.

```
if uy == y2:
    plt.text_at(1, "hole in one", "right")
else:
    plt.text_at(1,"you missed","right")
```

*plt.text_at can be found in the Menu using:
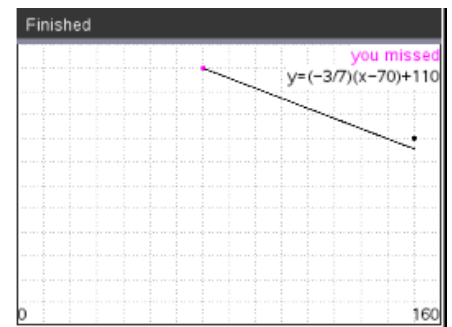Menu> TI-PlotLib> Draw> text_at

21. Execute your program many times.

If you answer a question correctly, does it say "hole in one"?
If incorrect, does it say "you missed"?

Can you modify the code to display "you missed"
and "hole in one" in different colors?
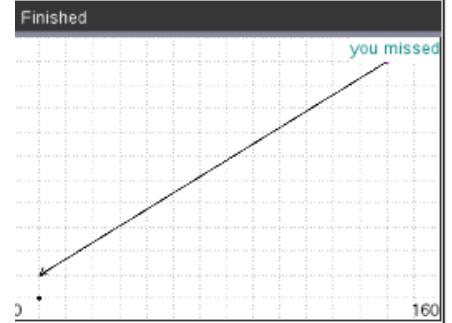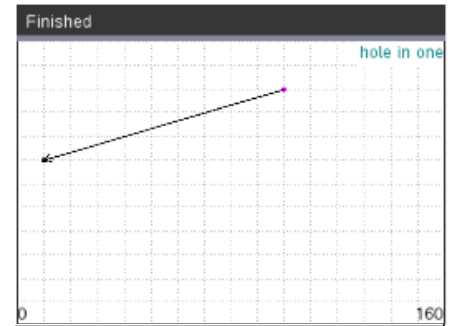
22. Extension: Add text to display the equation of the line segment

23. You need to store the fraction as text before the program evaluates the
    numeric approximation.

    Modify, the line:
      m = eval(input("enter the slope:"))
    to
      fraction = input("enter the slope:")
      m = eval(fraction)

24. To print the equation in the form y = m(x – x1) + y1, you will need to combine words such as "y=" with variable values.
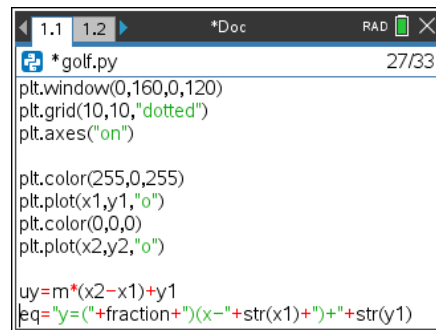
    In coding, letters, numbers, and symbols are called string values. When you put quotes around string values the python editor changes the font color to green.

    To concatenate, or add together, strings and variables use the plus symbol. In algebra, you combine like terms. In Python, you cannot combine a string with a numeric variable unless you type cast the numeric value to a string using the function str().

    For example, "y=(" + fraction is a valid concatenation because the variable fraction is already a string storing the text entered from the user.

    However, to combine ")x-" + x1 is not valid because x1 is a float variable containing a numeric value. To combine these values using type casting you would enter ")x-" + str(x1).

    To store the entire equation as one string variable named eq type
        eq = "y=(" + fraction + "(x-" + str(x1) + ")+" + str(y1)

25. To display the equation you will use the text_at() command. The text_at command has three parameters, or values it needs to function.
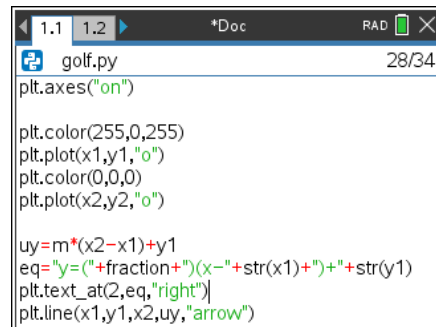
        plt.text_at(row, text, alignment)

    The row values range from 1 at the top to 13 at the bottom of the screen. The text is a string value so it is typed using quotes unless using a variable. The alignment can be left, center or right.

    If you type   plt.text_at(2, eq, "right")   your equation will appear in the top right corner of the screen on the second line.

    Decide where you want to display the equation, then add the last line of the code.

    *plt.text_at can be found in the Menu using:
            Menu> TI-PlotLib> Draw> text_at

26. Challenge:

    Modify your code to ask the user for the slope AND the y-intercept.

Use the equation y = mx + b to calculate the user's
y value (uy) using m, b, x2.

**To save this file and make a new copy
    Doc> File> Save   Will save this file

    Doc> File> Save As   Will allow you to create a copy to
                      modify for the slope intercept form.

Teacher Notes:
```
#Possible Challenge Solution:
# Plotting (x,y) & Text
#==============================
import ti_plotlib as plt
from random import *
#==============================
x1=randint(1,15)*10
y1=randint(1,11)*10
print("golf ball at: ",x1,",",y1)
x2=randint(1,15)*10
y2=randint(1,11)*10
if x1==x2:
 x1+=10
print("hole at: ",x2,",",y2)
fraction =input("enter the slope: ")
m = eval(fraction)

#new code for the challenge
b=eval(input("enter the y-intercept: "))

plt.window(0,160,0,120)
plt.grid(10,10,"dotted")
plt.axes("on")

plt.color(255,0,255)
plt.plot(x1,y1,"o")
plt.color(0,0,0)
plt.plot(x2,y2,"o")

#altered uy code for the challenge
uy=m*x2+b

#altered eq code for the challenge
```

```
eq = "y=(" + fraction + ")x + " + str(b)

plt.text_at(2,eq,"right")
plt.line(x1,y1,x2,uy,"arrow")

plt.color(0,128,128)
if uy == y2:
  plt.text_at(1, "hole in one", "right")
else:
  plt.color(255,0,255)
  plt.text_at(1,"you missed","right")
```