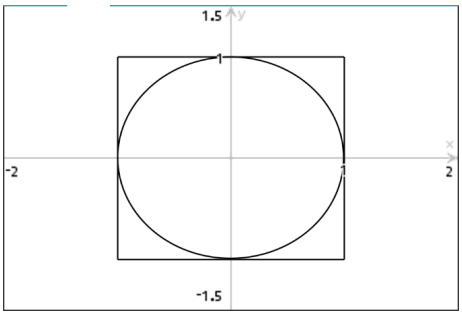
## Unit 5: The TI Modules — Application: Pi Darts

In this application, you will create a simulation that (eventually) approximates the value of **pi**, the ratio of the circumference of a circle to its diameter — about 3.14159. …

**Objectives:**

- Use random decimal numbers (uniform)
- Use a **while not escape()** loop
- Plot points on a "dart board"
- Display text on a plot screen

You have a square dartboard measuring **2** units on a side with a circle inscribed in the square. Toss some darts at random onto the board. Some will land inside the circle ("hits"), and some will land in the square but outside the circle ("misses").




The *ratio* of the "hits" to the total number of darts tossed is a function of the **areas** of the two shapes: circle/square.

The area of the square is ___.

The area of the circle is ___.

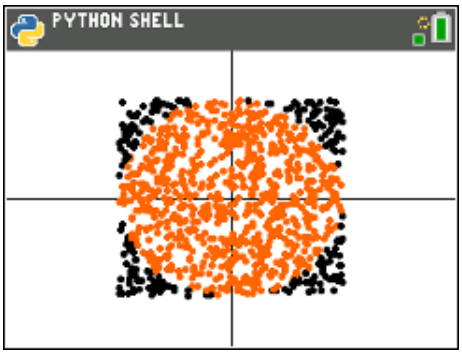
The *ratio* circle/square is ___.

Write a program to perform this simulation, and approximate the value of **pi**.

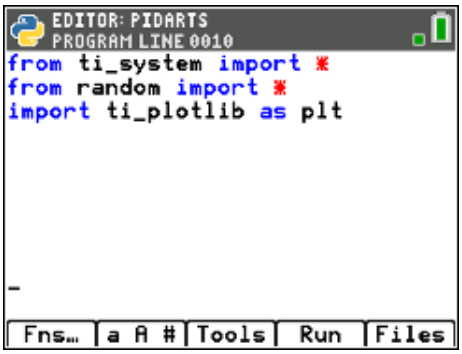
**Teacher Tip:** Area of square = 4_u$^2$ and area of circle is pi_u$^2$. The ratio is pi/4. Multiplying the ratio by 4 gives an approximation for **pi**.

1. The plan:
   - Use **escape()** to end the program
   - Select two **random** decimal (real) numbers between -1 and 1 and if the point is inside the circle count it as a hit
   - **Plot** the point on the graphics screen in a special color

   This plan requires tools from three modules: **ti_system**, **random** and **ti_plotlib**.




2. Begin a new program (PIDARTS) and get the three import statements from the **<Fns…> Modul** menus.

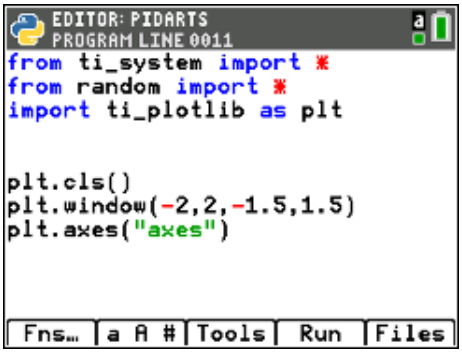


3. Use these **plotlib** setup commands to create a nice graphics screen. Select these commands from **<Fns…> Modul ti_plotlib**.

   **plt.cls()** #clear the screen

   **plt.window(-2, 2, -1.5, 1.5)** #a nice viewing window

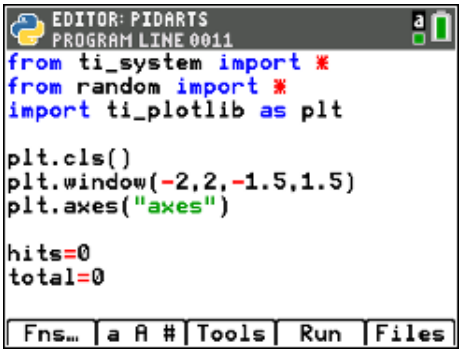
   **plt.axes("axes")** #show only the axes




4. Make two variables for counting the darts and counting the hits. Initialize both to 0.

   **hits = 0** # of darts inside the circle

   **total = 0** # of darts tossed




5. Write the loop that terminates when **[clear]** is pressed:

   Select **while not escape():**

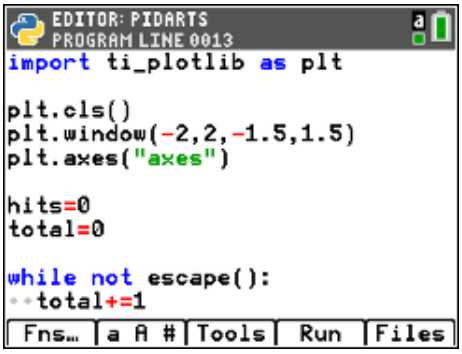
   from the **ti_system** module menu.

   The first action in the loop is to toss a dart. Just add 1 to the total variable:

   ♦♦**total += 1**

   This statement is indented because it is in the loop body.

   *Note:* **total += 1** *is the same as* **total = total + 1** *but requires less typing.*




6. Now choose two random *decimal numbers* between -1 and 1. They must be floating-point numbers, not integers, so use the **uniform**() function:

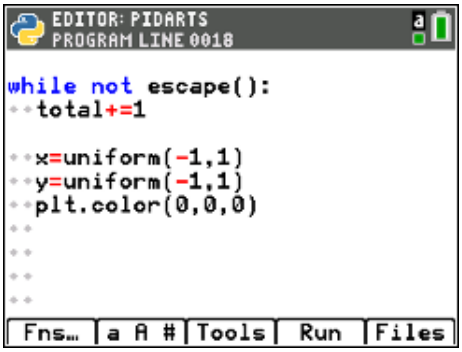
   ♦♦**x = uniform(-1, 1)**

   ♦♦**y = uniform(-1, 1)**

   The function **uniform()** selects a random floating-point value (includes decimals) between *min* and *max*.

   Find **uniform()** on **<Fns…> Modul random**.

   Set the plot color to black using **plt.color(0,0,0)** found on the **<Fns…> Modules ti_plotlib Draw** menu.



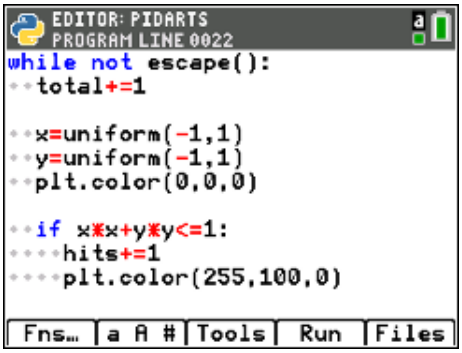
*The three arguments are integers in the range 0 ... 255 and represent the amount of red, green and blue to use in the color mixture. Feel free to choose other colors later.*

7. **if** the dart is inside the circle **…**

   Use the condition **x*x + y*y <= 1** because all the points inside the circle are within 1 unit of the origin. The distance of a point from the origin is given by **d = sqrt(x**2 + y**2)** from the Pythagorean Theorem. But the square root of 1 is 1 so we do not need the sqrt function.

   Count it as a "hit":

   ♦♦♦♦**hits += 1**




Change the plot color (we use orange for a hit):

♦♦♦♦**plt.color(255, 100, 0)**

*Note that these statements are indented **even more** because they are inside the* **if** *block which is inside the* **while** *block.*

8. Dedent (un-indent) the next line to exit the **if** block but remain in the **while** block. Plot the point.

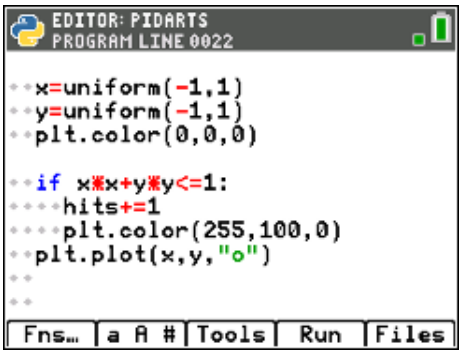
   Select

   ♦♦**plt.plot(x, y, "mark")**

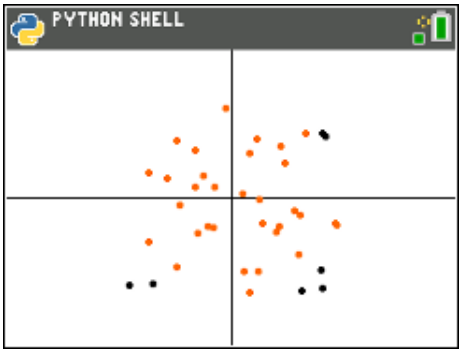
   from **<Fns…> Modules ti_plotlib Draw**.

   You will choose the "mark" on a sub-menu. If you want to change the "mark" later on take note because the menu only appears once. The marks are **o**, **x**, **+** and **.**

   You will have to fill in the variables **x** and **y** in the code.




9. Run the program now. You should see some dots appear. After a while, press **[clear]** (that's the "escape" key) and you will return to the Shell.



10. Our screen is lacking some information! Where's **pi**?
    In the Editor you can use **plt.text_at( )** to display "text," the value of a variable, or the result of an expression like **hits/total.**

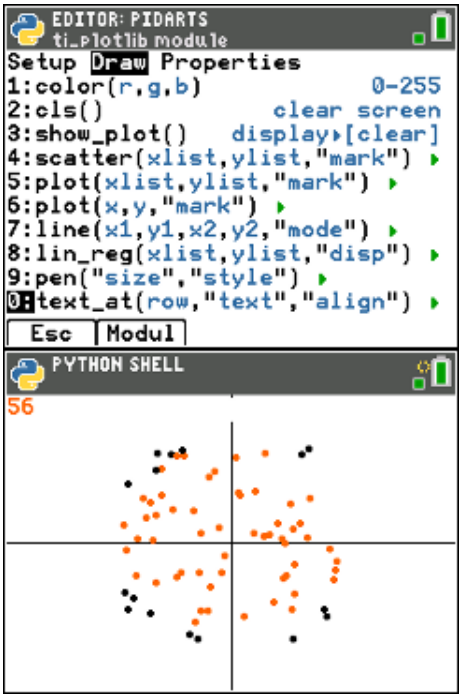
    **plt.text_at(1, str(hits), "left")**

    would show the value of the **hits** variable in the upper left corner of the screen. **str( )** is needed to convert the **hits** value into a string for the **text_at( )** function.

11. Adding

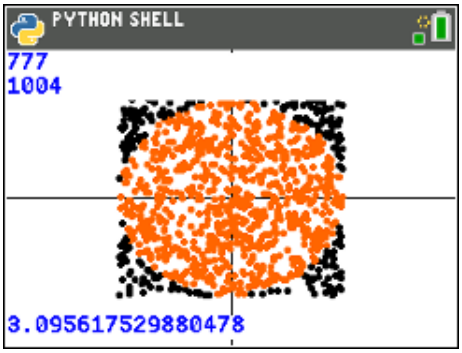
    **plt.text_at(1, str(hits), "left")**

    right below the **plt.plot( )** statement produces the number seen in the upper left corner. But the color used is not always the same. Sometimes it is black, and sometimes it is orange. Add another **plt.color( )** statement just before the **plt.text_at( )** statement to fix the color.



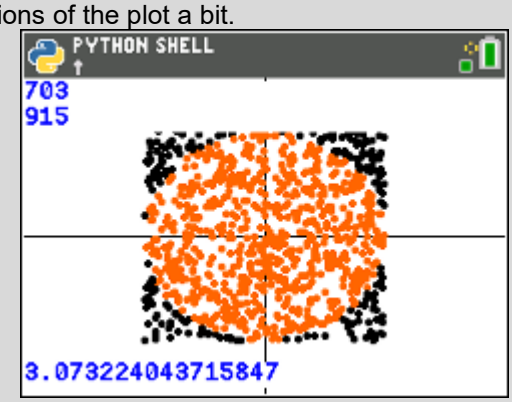

12. There are 12 rows available for **text_at( )**. Add more information to the display: the **total** number of darts tossed and the expression **hits/total * 4.**

    Why multiply by 4?




*Note: The* **text_at( )** *function impacts the entire line, not just the area where the text is displayed, so you need to leave room at the top and bottom of the screen for the text.*

**Teacher Tip:** The viewing window might need to be increased a bit to allow for the text on the screen**. text_at** draws a line of text *all the way across the screen*, so it might chop off the top and bottom portions of the plot a bit.




Multiply by 4 because the areas are **pi** and **4** so the ratio approaches pi/4. Multiplying by 4 yields an approximation for **pi**.

The complete program:

```
from ti_system import *
from random import *
import ti_plotlib as plt

plt.cls()
plt.window(-2,2,-1.5,1.5)
plt.axes("axes")

hits=0
total=0

while not escape():
  total+=1
  x=uniform(-1,1)
  y=uniform(-1,1)
  plt.color(0,0,0)        # black dots outside circle
  if x*x+y*y<=1:
    hits+=1
    plt.color(255,100,0)  # orange dots inside circle
  plt.plot(x,y,"o")       # change “o” to “.” for smaller dots

  plt.color(0,0,255)      # blue text
  plt.text_at(2,str(total),"left")
  plt.text_at(1,str(hits),"left")
  plt.text_at(12,str(hits/total*4),"left")

#plt.show_plot()  # keeps the graph screen on (after loop ends) until [clear]
is pressed again.
```