In this project, you will create a dart game that uses factoring to earn darts. The game will generate 4 random factor problems. If you answer correctly, you will earn a dart. Answer two questions in a row, earn an additional dart. After factoring 4 questions, toss your darts to earn a score. How high can you score? Answer all 4 questions correctly, earn 6 darts. Can you score a perfect 30?

Objectives:

Programming Objectives:

- Use the randint() function to generate random integers.
- Use for loops to repeat code.
- Use selection statements to make decisions.
- Use the draw functions from the TI-Draw library to create circles and text.
- Use the sleep function from the Time library to create animation.

Math Objectives:

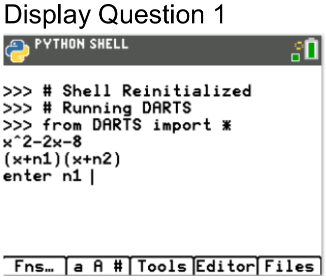
- Practice solving quadratic equations
- Use the distance formula to find the distance between two points
- Use the modulus operator, %, to find a remainder
- Use cosine and sine functions to draw circles.

Math Course Connections: This activity is recommended for Algebra 1 and Algebra 2.

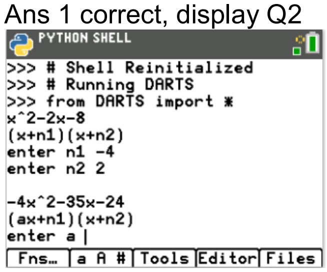
You will create a dart game that generates four random quadratic expressions of the form: $x^2 + bx + c$ or $ax^2 + bx + c$. For each question the player answers correctly, the player will get another shot at the dart board. At the end of the game, the dartboard shows the darts on the board as well as the score the round.

Example:

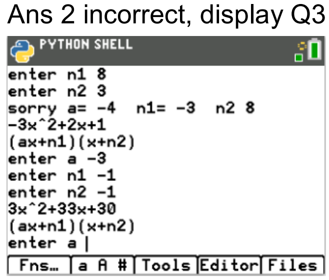
Display Question 1



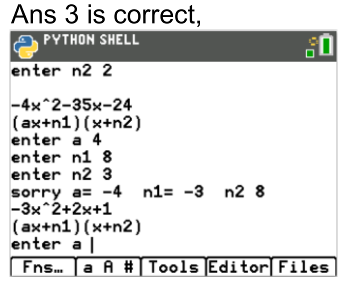
Ans 1 correct, display Q2



Ans 2 incorrect, display Q3

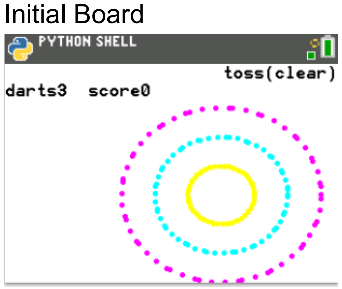


Ans 3 is correct,

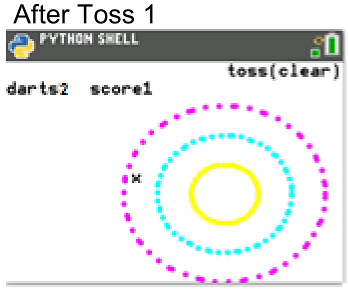


Last answer is incorrect. Earn 3 darts, 2 correct questions. One bonus for two correct in a row.

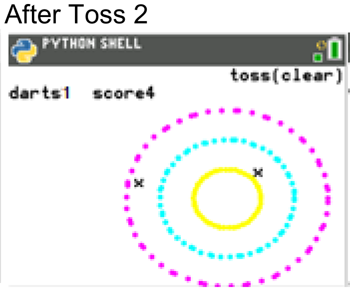
Initial Board



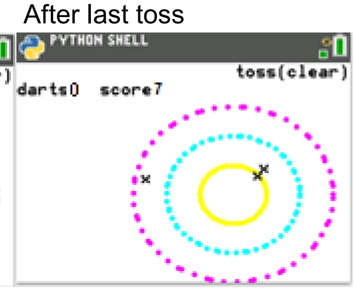
After Toss 1



After Toss 2



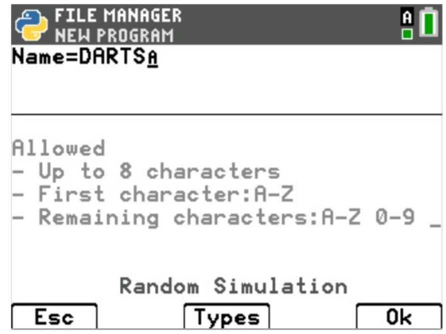
After last toss



1. The first step will be to create a python random simulation document.

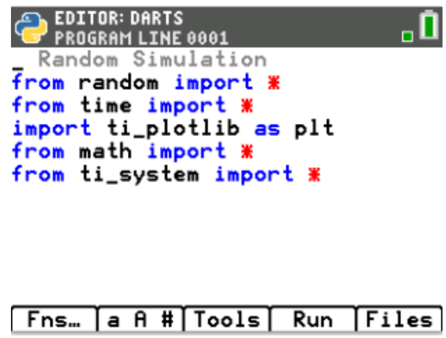
   Create a new python project named "darts".

   Select "Random Simulation" from the types menu.



2. You will need four more libraries: ti_plotlib, math, ti_system and time

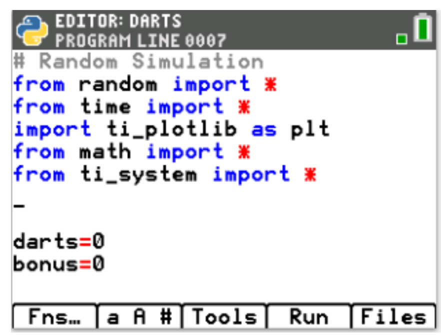
   The random library contains the randint() function.
   The time library includes the sleep() function.
   The ti_plotlib contains the plotting functions.
   The math library contains the square root function sqrt().



   Fns > Modul > Math> from math import *
   Fns > Modul > TI_plotlib > from ti_plotlib import *
   Fns > Modul > TI_system > from ti_system import *

3. The user will earn one dart for each correct answer. The user will earn a bonus dart for every two correct questions answered in a row. Create two variables, one to hold the number of darts, the other to keep track of the questions answered in a row.

   darts = 0
   bonus = 0

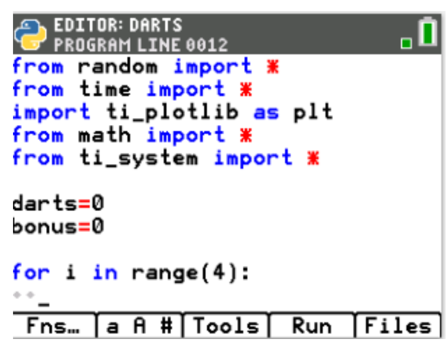


4. Initially, you will set the game to play for 4 rounds. Once the entire game is coded, you can easily change this to as many rounds as you like.

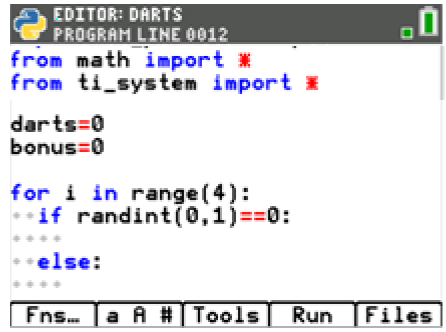
   Add a for loop that will repeat 4 times.

   Fns > Ctl > for index in range(size)

   for i in range(4):



5. Each question will either be an "easy" or "difficult" question. If you want to set the probability to 50% each question will be "easy", You need two choices. For that, use randint(0,1). If the number is a 0, generate an easy question, otherwise generate a "difficult" question. If you want more easy questions to appear, change the randint(0,1) to something like randint(0,3). That will generate numbers 0-3. Now say if randint(0,3) <3 generate an "easy" question. That will make each question have a 75% probability of being an "easy" question.



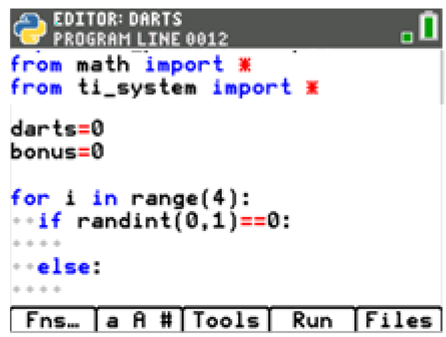
Add an if..else statement with randint values of your choice.

if randint(0,1) == 0:

else:

Fns > Ctl > if..else
Fns > Random > randint

6. Compare your code to the one on the right.
Make sure your indentation is correct. Python uses indentation to keep loops and selection statements together. The lines under the for statement have one level of indentation (two diamond spaces). Each line that is part of the if statement is also indented one level (two diamond spaces).



7. In the first if statement, you will generate the "easy" questions in the form $x^2 + bx + c$.
These questions will be factored in the form (x + n1)(x + n2).

You could generate values for b and c and then try to break them apart into n1 and n2.
But, it is easier to generate n1 and n2, then use them to find the values for b and c.

Why? You might ask.

$$(x + n1)(x + n2)$$

$$x^2 + n2*x + n1*x + n1*n2$$

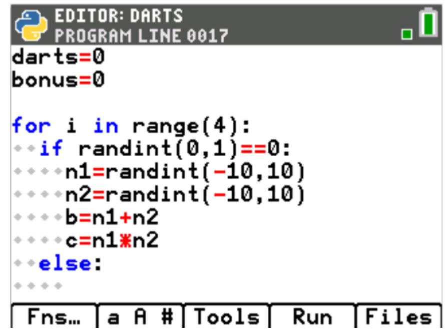
$$x^2 + (n1+n2)x + n1*n$$

The sum of n1 and n2 create the b value. The product of n1 and n2 create the c value.

8. Inside the if statement. Randomly generate n1 and n2
   Use these numbers to create b and c.

   n1 = randint(-10,10)
   n2 = randint(-10,10)
   b = n1+n2
   c = n1*n2

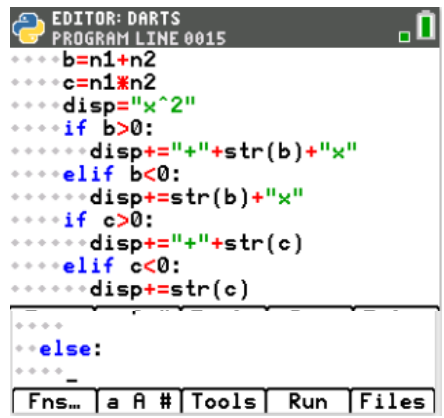
   **Hint: The Tools menu, [zoom], contains copy/paste line commands. These could be useful when creating n2.



9. To display the function correctly is a bit tricky. For example, if the sum is -3 and the product is 10, you want the display to be x^2-3x+10. If we coded the project with disp = “x^2” + b + “x+” + c, it would display correctly for this problem.

   However, if the values were b = 3 and c = -10, the display code above would display: x^23x+-10.

   Therefore, you will need an if statement.
   all displays start with x^2
   if the sum is positive,
   add “+” and b and “x”
   otherwise if the sum is negative,



Add the lines:

```
disp = “x^2”
if b > 0:
    disp += “+” + str(b) + “x”
elif b < 0:
    disp += str(b) + “x”
if c > 0:
    disp += “+” + str(c)
elif c < 0:
    disp += str(c)
```

Fns > Type > str

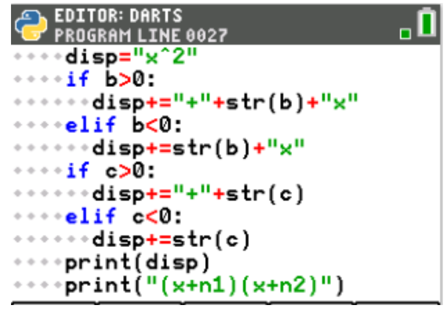
Hint: The [a A #] menu might help with some of the typing.

10. Now to display the question and the form of the input.
print(disp) will print the equation on the first line.
print("(x+n1)(x+n2)") will tell the user the form of question input.

Add the lines:

```
print(disp)
print("(x+n1)(x+n2)")
```

Fns > I/O > print



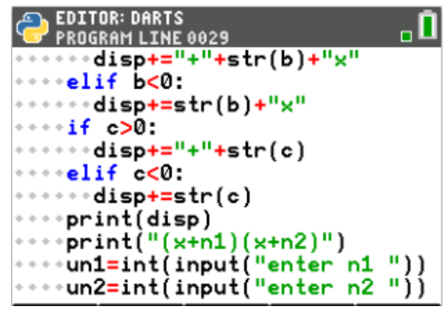
11. Ask the user for the values, n1 and n2.
Store these values in variables named un1 and un2.

```
un1 = int(input("enter n1 "))
un2 = int(input("enter n2 "))
```

Fns > Type > int
Fns > I/O > input

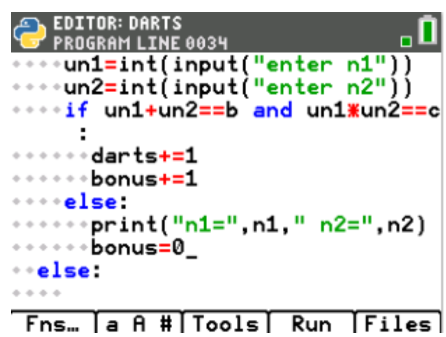
**By default, input returns a string value. String values are treated like characters not numbers. Place int() around the input casts the input as an integer.

**Hint: The tools menu, [zoom], has copy/paste commands that might be helpful.



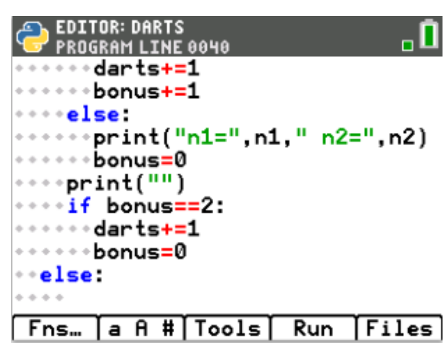
12. If the input is correct, add one to the number of darts and the bonus variable. Otherwise, print the correct result and set bonus back to 0.

```
if un1+un2==b and un1*un2==c:
    darts+=1
    bonus+=1
else:
    print("n1 =", n1, "  n2 =" ,n2)
    bonus=0
```



13. Unindent from the if statement. (Make sure you have 4 diamonds).
Print a blank line. If the user has answered two questions in a row correctly (bonus=2), then add a free dart and set the bonus back to 0.

```
print(" ")
if bonus==2:
   darts+=1
   bonus=0
```



14. Now to repeat the same process for the "difficult" questions.
    You will:
    - generate the values
    - display the question
    - get the user's input
    - check the user's input
    - add darts if need be, otherwise display the correct values.

15. The second quadratic form will factor into (ax + num1)(x + num2).

    How would you calculate b and c in this form?

    Use distribution to factor (ax + num1)(x + num2)

16. Did you get the following?

    (a*x + n1)(x + n2)

    ax^2 + a*n2*x + n1*x + n1*n2

    ax^2 + (a*n2+n1)x + n1*n2

    b = a*n2 + n1
    c = n1*n2

17. Add 6 lines of code.

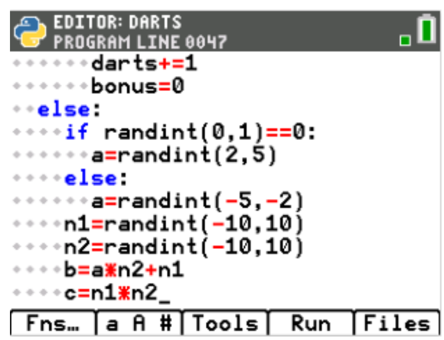
    a.) Generate a number from 0 to 1.
        If the number is a 0, variable a should be between 2 and 5
        otherwise a is a number between -5 and -2.
    b.) Generate n1 to be a random number between -10 and 10
    c.) Generate n2 to be a random number between -10 and 10.

18. Add the two lines to calculate b and c

    b = a*n2 + n1
    c = n1*n2

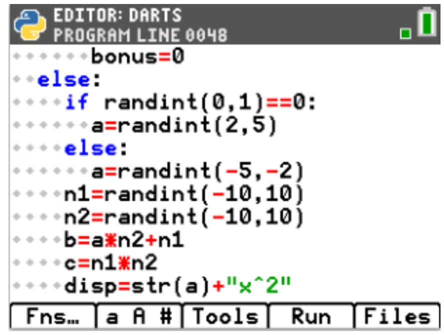
19. Check the last 8 lines of code. Did you do the following?

```
if randint(0,1)==0:
    a=randint(2,5)
else:
    a=randint(-5,-2)
n1 = randint(-10,10)
n2 = randint(-10,10)
b = a*n2 + n1
c = n1*n2
```



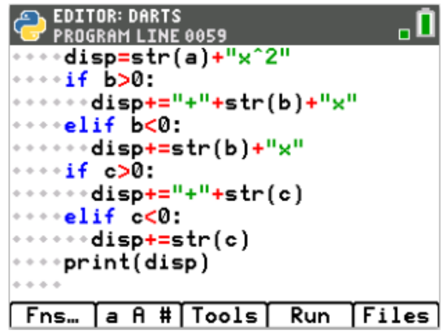
20. The display code for the “difficult” form is the exact same as the display code for the “easy” form. The only difference is the first line of code. Add the line:

    disp = str(a) + “x^2”



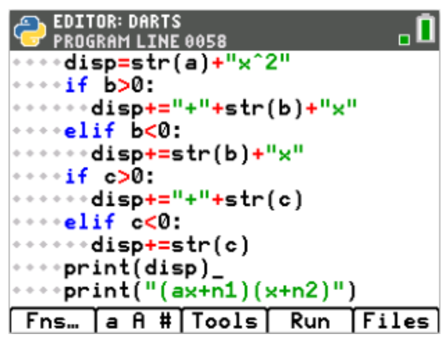
21. You could retype the 8 lines it takes to finish the display or you can copy and paste those line. Remember the tools menu, [zoom], has copy/paste line commands.

```
if b > 0:
   disp += “+” + str(b) + “x”
elif b < 0:
   disp += str(b) + “x”
if c > 0:
   disp += “+” + str(c)
elif c < 0:
   disp += str(c)
```



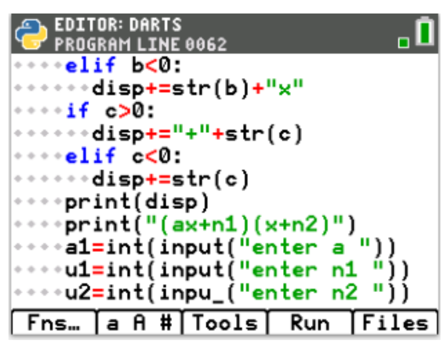
22. Add the display line:

    print(“(ax+n1)(x+n2)”)



23. Ask the user for a, n1, and n2. Store the values as a1, u1 and u2.

    a1 = int(input(“enter a “))
    u1 = int(input(“enter n1 “))
    u2 = int(input(“enter n2 “))



24. Check the user’s input against the correct values.

    This code will be very similar to the code you wrote for the “easy” form.

    Remember you generated a, n1, n2 then stored them as b and c.

(a*x + n1)(x + n2)

ax^2 + a*n2*x + n1*x + n1*n2

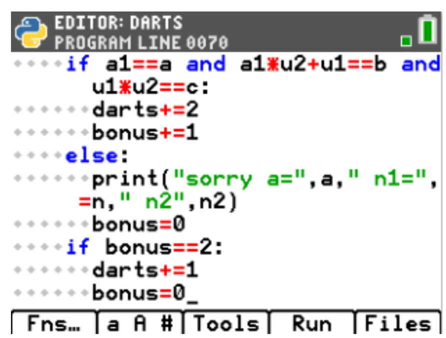
ax^2 + (a*n2+n1)x + n1*n2

You asked the user for a1, u1, u2.

Fill in the missing code statements below:

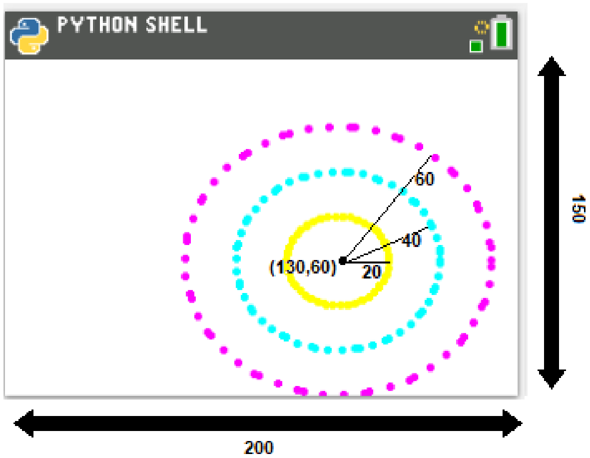
```
if ________==a and ____________ == b and ____________ ==c:

        darts ___________

        bonus ___________
else:
        print(___________________________________)

        bonus ___________

if bonus == ____:

        darts___________

        bonus___________
```

25. Does you code match the code below?




```
if a1==a and a*u2+u1==b and u1*u2==c:
   darts+=2
   bonus+=1
else:
   print("sorry a =", a, "  n1=", n1,"  n2=", n2)
   bonus=0
if bonus==2:
   darts+=1
   bonus=0
```

26. The dartboard will be drawn after all the questions have been answered. Therefore, unindent.



Draw the dart board using three concentric circles.

The window will be [200,150].

Each circle will be centered at (130,60)

To draw the circle, you will plot points using the parametric form of a circle.

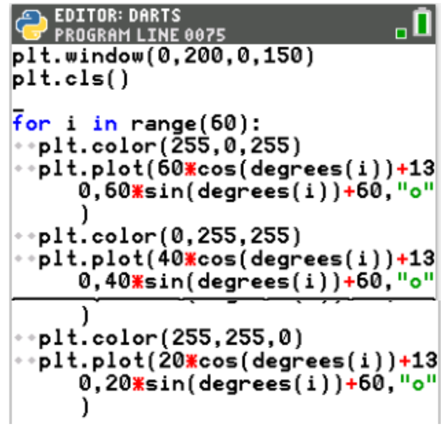
x = radius*cos(θ) + center$_x$ and y = radius*sin(θ) + center$_y$

Complete the template below:

```
for i in range(60):
    #magenta large circle
    plt.color(255,0,255)
    plt.plot(60*cos(degree(i)) + 130, 60*sin(degree(i)) + 60, "o")

    #blue middle circle (pick any color you like)

    plt.color(_____, _____, _____)

    plt.plot(________________________,________________________, "o")

    #yellow smallest circle (pick any color you like)

    plt.color(_____, _____, _____)

    plt.plot(________________________,________________________, "o")
```

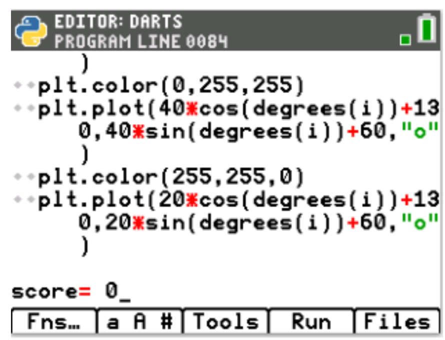
27. Code the loop for the circles. Feel free to change the colors.
Before the loop, clear the screen and set the window.

```
plt.window(0,200, 0,150)
plt.cls()
for i in range(60):
    plt.color(255,0,255)
    plt.plot(60*cos(degrees(i))+135, 60*sin(degrees(i))+60, “o”)
    plt.color(0,255,255)
    plt.plot(40*cos(degrees(i))+135, 40*sin(degrees(i))+60, “o”)
    plt.color(255,255,0)
    plt.plot(20*cos(degrees(i))+135, 20*sin(degrees(i))+60, “o”)
```



Fns > Modul > ti_plotlib > Draw > color
Fns > Modul > ti_plotlib > Draw > plot
Fns > Modul > math > Trig > cos
Fns > Modul > math > Trig > sin
Fns > Modul > math > Trig > degree

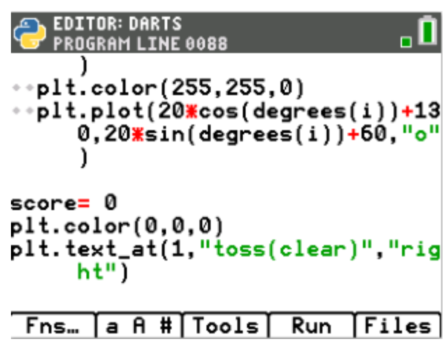
28. Create a score variable that starts at 0.



29. You will code the project to let the user press [clear] to toss a dart.
To let the user know this, draw “toss(clear)” to the screen.

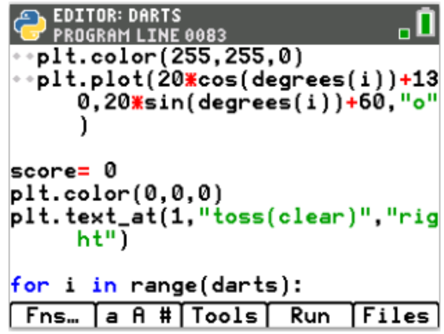
```
plt.color(0,0,0)
plt.text_at(1,”toss(clear)”,”right”)
```

Fns > Modul > TI_plotlib > Draw > text_at

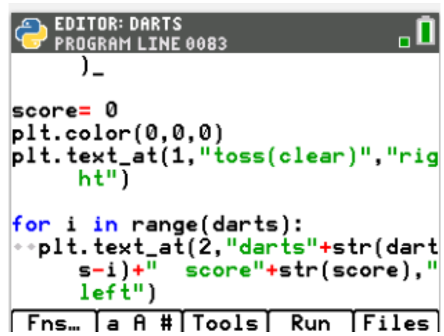


30. Now to toss the darts.
    Create a loop that will repeat itself for each dart.

    for i in range(darts):



31. For each toss of a dart, you will erase the old dart count and draw a new one.
    Add the line:
    plt.text_at(2, “darts”+str(darts-i)”+ “   score”+str(score),”left”)



32. We don’t want to throw the dart until the user presses [clear].
    Add the lines:

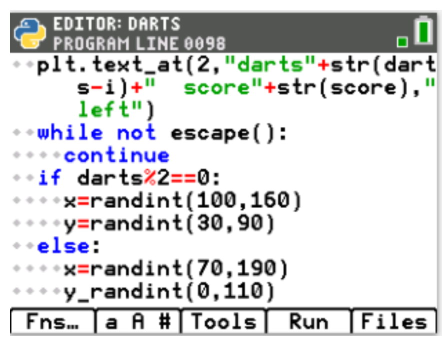
    while not escape():
        continue

    Fns > Modul > TI System > while not escape

33. For some shot variation, we will say:
    If the shot was on an “even” dart throw,
        generate a targeted shot x: [100,160] and y: [30,90].
    Otherwise
        generate a wide shot x: [70,190] and y: [0, 110].

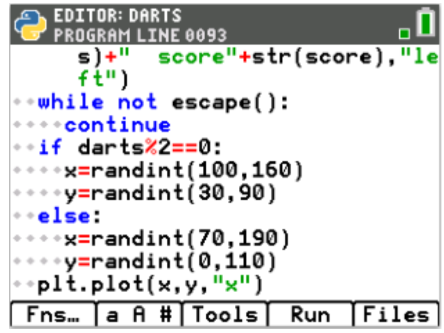
    The modulus symbol % is used to find the remainder in division.
    Even dart numbers such as 8, 6 and 4 all return 0 if you type darts%2.
    Odd dart numbers such as 9, 7 and 5 all return 1 if you type darts%.

    Add the following lines:
    if darts%2==0:
        x=randint(100,160)
        y=randint(30,90)
    else:
        x=randint(70,190)
        y=randint(0,110)



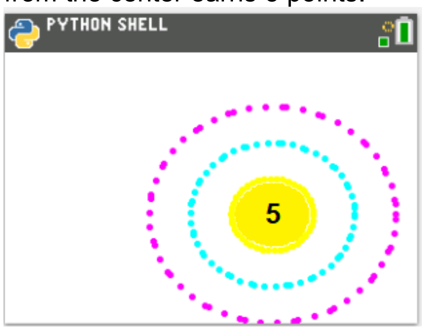
34. Plot the point

    plt.plot(x,y,"x")

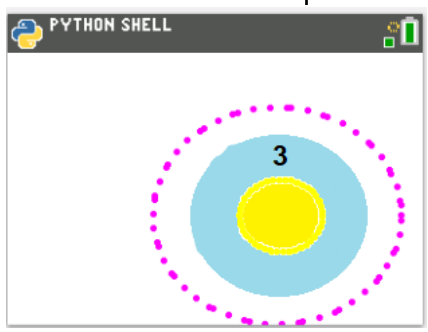


35. Points are scored based on the distance the dart lands from the bullseye.

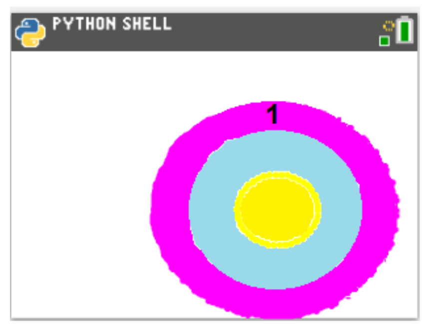
    Any dart that lands within 20 units from the center earns 5 points.



    Any dart that lands within 40 units from the center earns 3 points.



    Any dart that lands within 60 units from the center earns 1 point.

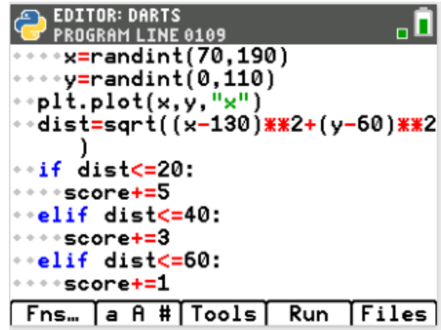


    If the dart is located at (x,y) and the center is at (130,60), what is the equation that represents the distance between the two points?

36. Use the equation from above to fill in the blanks below. Then code the lines on your project. Remember to use **2 instead of ^2 when coding the distance.

```
dist=____________

if dist <= 20:
  score += 5
elif dist_________:
  score+=3
elif dist_________:
  score+=1
```

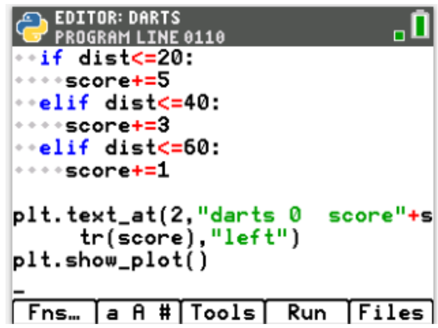
37. Does your code match the code to the right?



38. Lastly, update the score and add the line plt.show_plot()
    This will keep the picture on the screen until you press [clear] after tossing the last dart.

    plt.text_at(2, “darts 0   score”+str(score),”left”)
    plt.show_plot()

    **Hint: Copy and paste the display line. Then, modify the number of darts to 0.




```
# Random Simulation
from random import *
from time import *
import ti_plotlib as plt
from math import *
from ti_system import *

darts=0
bonus=0

for i in range(4):
  if randint(0,1)==0:
    n1=randint(-10,10)
    n2=randint(-10,10)
    b=n1+n2
    c=n1*n2
    disp="x^2"
    if b>0:
      disp+="+"+str(b)+"x"
    elif b<0:
      disp+=str(b)+"x"
    if c>0:
      disp+="+"+str(c)
```

```
  elif c<0:
    disp+=str(c)
  print(disp)
  print("(x+n1)(x+n2)")
  un1=int(input("enter n1 "))
  un2=int(input("enter n2 "))
  if un1+un2==b and un1*un2==c:
    darts+=1
    bonus+=1
  else:
    print("n1=",n1," n2=",n2)
    bonus=0
  print("")
  if bonus==2:
    darts+=1
    bonus=0
else:
  if randint(0,1)==0:
    a=randint(2,5)
  else:
    a=randint(-5,-2)
  n1=randint(-10,10)
  n2=randint(-10,10)
  b=a*n2+n1
  c=n1*n2
  disp=str(a)+"x^2"
  if b>0:
    disp+="+"+str(b)+"x"
  elif b<0:
    disp+=str(b)+"x"
  if c>0:
    disp+="+"+str(c)
  elif c<0:
    disp+=str(c)
  print(disp)
  print("(ax+n1)(x+n2)")
  a1=int(input("enter a "))
  u1=int(input("enter n1 "))
  u2=int(input("enter n2 "))
  if a1==a and a1*u2+u1==b and u1*u2==c:
    darts+=2
    bonus+=1
  else:
    print("sorry a=",a," n1=",n1," n2",n2)
```

```
     bonus=0
    if bonus==2:
     darts+=1
     bonus=0


plt.window(0,200,0,150)
plt.cls()

for i in range(60):
  plt.color(255,0,255)
  plt.plot(60*cos(degrees(i))+130,60*sin(degrees(i))+60,"o")
  plt.color(0,255,255)
  plt.plot(40*cos(degrees(i))+130,40*sin(degrees(i))+60,"o")
  plt.color(255,255,0)
  plt.plot(20*cos(degrees(i))+130,20*sin(degrees(i))+60,"o")

score= 0
plt.color(0,0,0)
plt.text_at(1,"toss(clear)","right")

for i in range(darts):
  plt.text_at(2,"darts"+str(darts-i)+"  score"+str(score),"left")
  while not escape():
    continue
  if darts%2==0:
    x=randint(100,160)
    y=randint(30,90)
  else:
    x=randint(70,190)
    y=randint(0,110)
  plt.plot(x,y,"x")
  dist=sqrt((x-130)**2+(y-60)**2)
  if dist<=20:
    score+=5
  elif dist<=40:
    score+=3
  elif dist<=60:
    score+=1

plt.text_at(2,"darts 0  score"+str(score),"left")
plt.show_plot()
```