Shuffleboard Slide with Proportions

In this project, you will create a disk gliding game (shuffleboard). The game lets us the user "aim" at a triangle scoreboard. Once the user presses "s", the disk will slide a random distance forward toward the scoreboard. The score will be determined based on the final resting location of the disk. You are tasked with writing the code to create the scoreboard and determine a score for each disk. The rest of the code will be provided by your teacher.

Objectives:

Programming Objectives:

- Use the draw_poly() function to draw triangles
- Use the set_color() function to set pen colors
- Use the randint() function to generate random integers.
- Use if statements to make conditional decisions

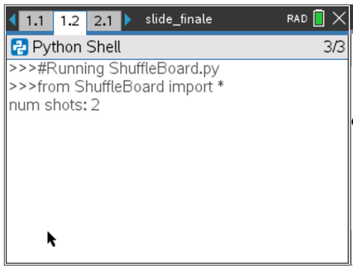
Math Objectives:

- Draw polygons on the coordinate plane
- Use ratios to transform units
- Decide whether two quantities are in a proportional relationship
- Use proportions to create scale polygons
- Draw geometric shapes using coordinates and technology

Math Course Connections: Middle School Mathematics

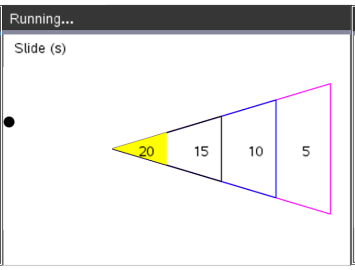
In this project, you will create a disk gliding game, "shuffleboard". The game lets the user "aim" at a triangle scoreboard. Once the user presses "s", the disk will slide a random distance forward toward the scoreboard. The score will be determined based on the final resting location of the disk. You are tasked with writing the code to create the scoreboard and determine a score for each disk. The rest of the code will be provided by your teacher.

Sample Game:

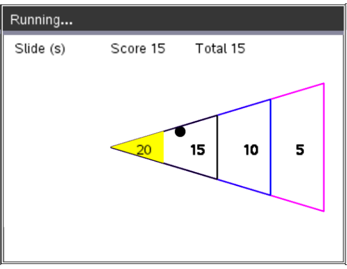


```
>>>#Running ShuffleBoard.py
>>>from ShuffleBoard import *
num shots: 2
```

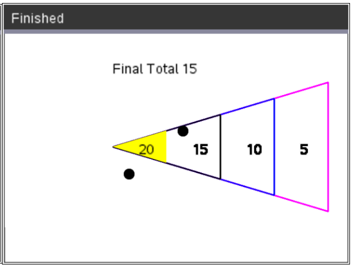
User selects 2 disks



Disk slides vertically until user presses "s" key



After "s" key is pressed, the disk slides random distance



Final score displayed after all disks are played

1. Obtain a copy of the "slide_template.tns" from your teacher. This contains the initial code for the project.

2. Let's examine the code.

Libraries needed for the projects

*You will add the 12 lines to draw the points board

Animation code for "s" slide key

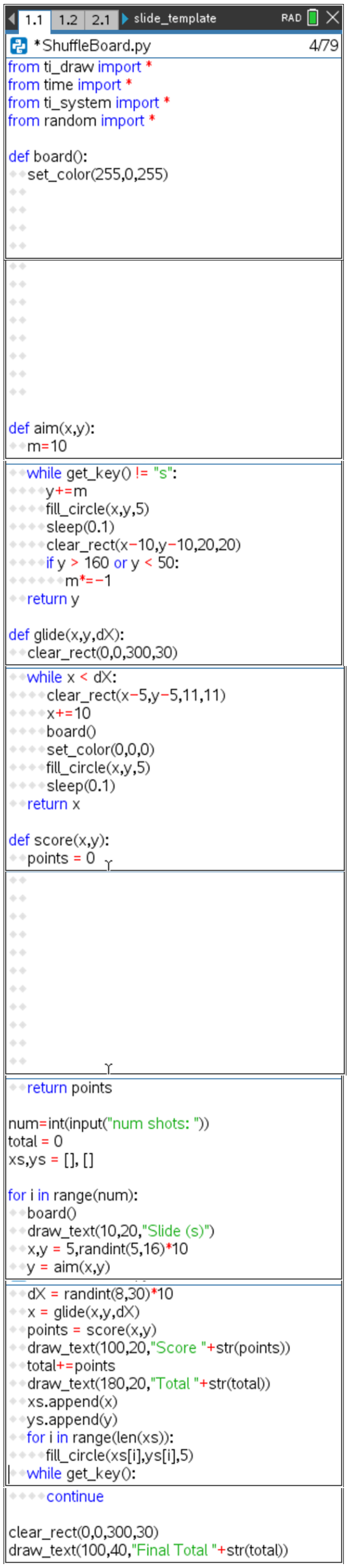
Animation code to slide the disk towards board

*You will add the 11 lines to score the location of the disk after the slide.

Get the number of disks, initialize some variables

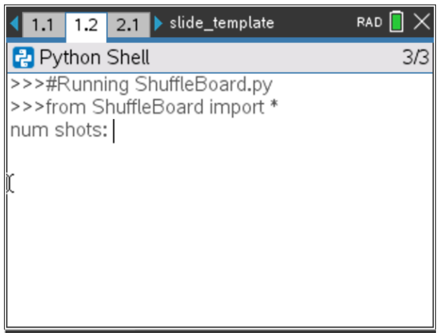
Use the functions to aim, slide and score each disk

Display the final score after clearing old values



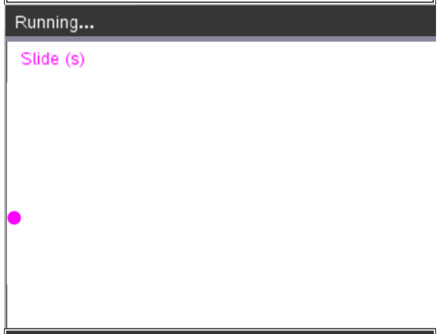
```
from ti_draw import *
from time import *
from ti_system import *
from random import *

def board():
  set_color(255,0,255)
  
  
  
  
  
  
  
  
  
  
  
  

def aim(x,y):
  m=10
  while get_key() != "s":
    y+=m
    fill_circle(x,y,5)
    sleep(0.1)
    clear_rect(x-10,y-10,20,20)
    if y > 160 or y < 50:
      m*=-1
  return y

def glide(x,y,dX):
  clear_rect(0,0,300,30)
  while x < dX:
    clear_rect(x-5,y-5,11,11)
    x+=10
    board()
    set_color(0,0,0)
    fill_circle(x,y,5)
    sleep(0.1)
  return x

def score(x,y):
  points = 0
  
  
  
  
  
  
  
  
  
  
  
  return points

num=int(input("num shots: "))
total = 0
xs,ys = [], []

for i in range(num):
  board()
  draw_text(10,20,"Slide (s)")
  x,y = 5,randint(5,16)*10
  y = aim(x,y)
  dX = randint(8,30)*10
  x = glide(x,y,dX)
  points = score(x,y)
  draw_text(100,20,"Score "+str(points))
  total+=points
  draw_text(180,20,"Total "+str(total))
  xs.append(x)
  ys.append(y)
  for i in range(len(xs)):
    fill_circle(xs[i],ys[i],5)
  while get_key():
    continue

clear_rect(0,0,300,30)
draw_text(100,40,"Final Total "+str(total))
```

3. Let's first execute the initial code (ctrl r), to see where we are.



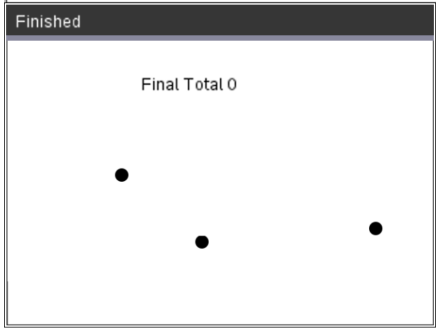
```
>>>#Running ShuffleBoard.py
>>>from ShuffleBoard import *
num shots:
```

Enter 3 for the number of shots.



```
Slide (s)
```

Observe the disk sliding back and forth before the "s" key is selected.



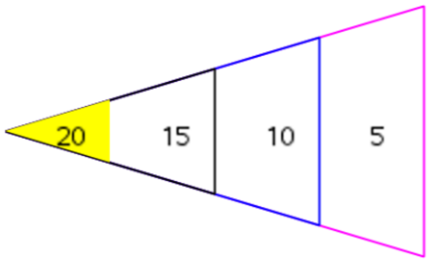
```
Final Total 0
```

After pressing "s" three times, the game is over. The final score displays.

Your disks will be in different locations. The location depends on when you press the "s" key as well as the random distance the disk travels.

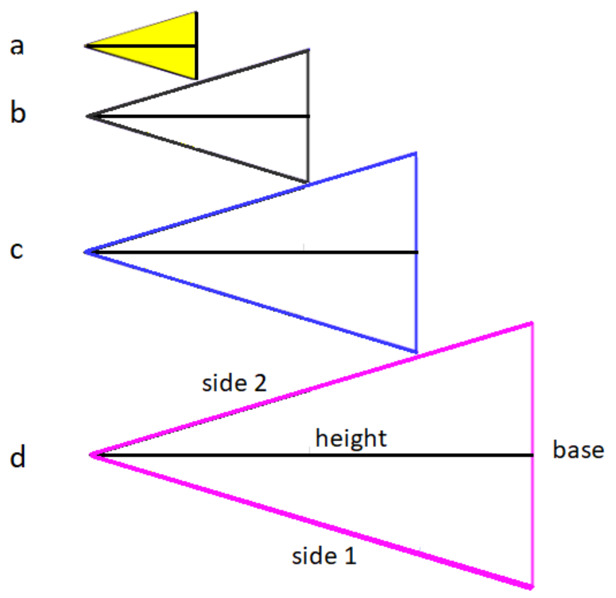
Doesn't look too much like a shuffleboard right now, but it will, after we add some additional code!

4. Before programming the score board, we need to investigate its composition. Once you understand the board's proportional relationships, you can code it using the drawing tools.



5. Below is a scaled and deconstructed version of the scoring board.

   Use a ruler to measure the height and base length for each triangle in centimeters.
   Then calculate $\frac{\text{height}}{\text{base}}$.



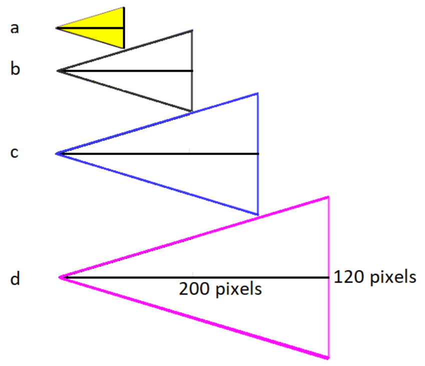
Triangle	Side 1 (cm)	Side 2 (cm)	Height (cm)	Base (cm)	$\frac{\text{height}}{\text{base}}$
a					
b					
c					
d					

6. What do you notice about $\frac{\text{height}}{\text{base}}$ for all 4 triangles?

7. There are a few more observations to be made.

	Ratio	Decimal	Percent
$\frac{\text{triangle a height}}{\text{triange d height}}$			
$\frac{\text{triangle b height}}{\text{triange d height}}$			
$\frac{\text{triangle c height}}{\text{triange d height}}$			
$\frac{\text{triangle a height}}{\text{triange c height}}$			

8. You know how the four triangles are related. **Now to scale these values to match the calculator screen**. The largest triangle is 200 pixels by 120 pixels. Use your relationships from the steps above to find the height and base for the three smallest triangles.



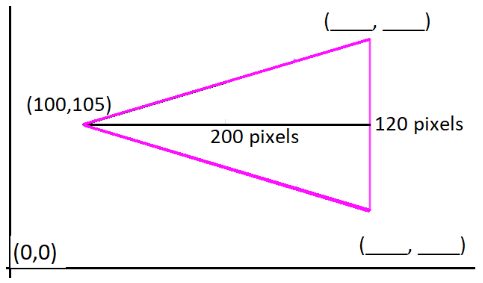
Triangle	Height (pixels)	Base (pixels)
a		
b		
c		
d	200	120

i. Show work or explain how you found the height and base length for triangle a.

ii. Show work or explain how you found the height and base length for triangle b.

iii. Show work or explain how you found the height and base length for triangle c.

9. The largest triangle will have a vertex at (100,105). The height is 200 pixels while the base is 120 pixels. Where are the other two vertices?



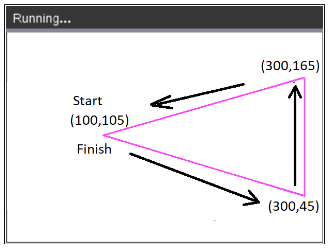
10. Look at the two vertices you calculated.

i. Because the highest vertex is directly above the lowest vertex, your x-coordinates should be the same.

ii. Subtract your smallest y-coordinate from the largest coordinate.
Is the distance 120 pixels?

iii. The y-value 105 should be the middle y-coordinate for the 120 pixel base.
Find the average of your two y-coordinates. Is the average 105?

11. To draw the triangle, start at (100,105). Draw a segment to (300,45). Then draw a segment to (300,165). Finally draw a segment back to (100,105).

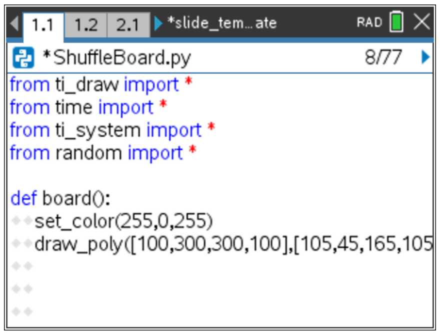


The command draw_poly, requires the list of x-values in order of travel as well as the y-values in order of travel. The x values are [100, 300, 300, 100]. The y-values are [105,45,165,105].

Add the following code to line 8 just below the set_color in the board function:

draw_poly([100,300,300,100],[105,45,165,105])
menu → More Modules → TI-Draw → shape → draw_poly

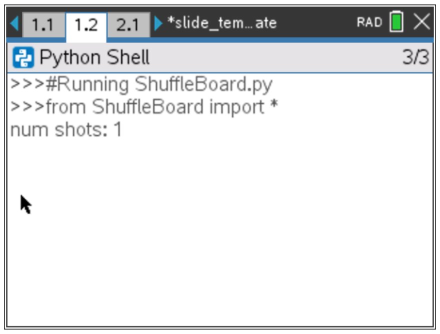
Make sure the draw_poly line is indented two spaces.
That means there should be two diamonds in front of this line.



```
from ti_draw import *
from time import *
from ti_system import *
from random import *

def board():
  set_color(255,0,255)
  draw_poly([100,300,300,100],[105,45,165,105
```

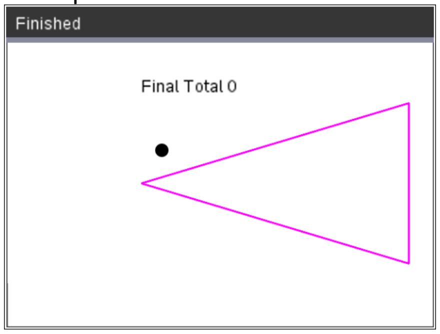
12. Execute your code (ctrl r). Enter “1” for the number of shots. You need at least one shot for the board to appear.

If everything is typed correctly, you shouldn’t have any errors in your code. If you have an error, fix the error before you continue. Mostly likely, your error is on the draw_poly line of code. Make sure all the parenthesis (), square brackets [] and commas, are correct.



```
>>>#Running ShuffleBoard.py
>>>from ShuffleBoard import *
num shots: 1
```

Example Run:



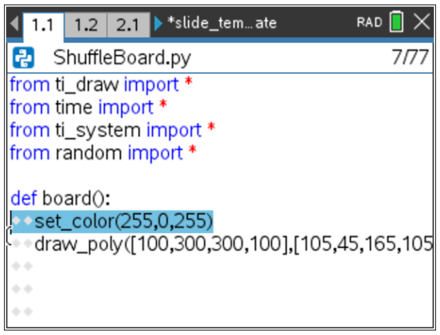
13. The line set_color(255, 0, 255) sets the pen color to magenta before the darw_poly line is executed. Magenta is made with 255-red, 0-green, 255-blue.

    This set_color function needs three colors, red, green, and blue.

set_color(255,0,0) is red.	255 red, 0 green, 0 blue
set_color(0,255,0) is green.	0 red, 255 green, 0 blue
set_color(0,0,255) is blue.	0 red, 0 green, 255 blue
set_color(255,255,0) is yellow.	255 red, 255 green, 0 blue

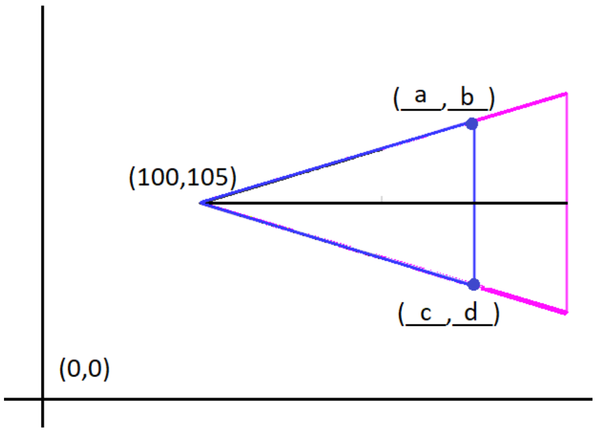
    You can pick any integer values from 0 to 255 for each red, blue, green.

    You may choose to change this color or keep it magenta. If you choose to change the color, execute your code to ensure you've entered valid integers.



```
ShuffleBoard.py                    7/77
from ti_draw import *
from time import *
from ti_system import *
from random import *

def board():
  set_color(255,0,255)
  draw_poly([100,300,300,100],[105,45,165,105
```

14. In step 8, you found the height and base for the 2nd largest triangle. Use these values to find the other two vertices (a,b) and (c,d).



(a,b) = (,)

(c,d) = (,)

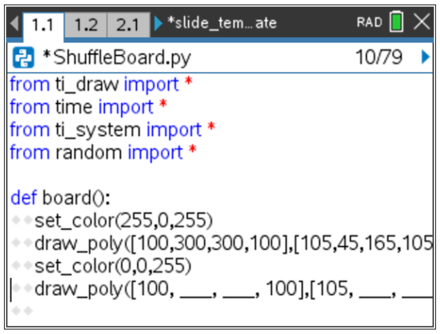
15. Starting at the point (100,105), trace around the smaller triangle.

    Record the x-values for the vertices in order [100, ____ , ____, 100]

    Record the y-values for the vertices in order [105, ____, ____, 105]

16. Change the color in the board function to blue, or any other color. Draw the new triangle.

    set_color(0,0,255)
    draw_poly([100, ___, ___, 100],[105, ___, ___, 105])



```
*ShuffleBoard.py                   10/79
from ti_draw import *
from time import *
from ti_system import *
from random import *

def board():
  set_color(255,0,255)
  draw_poly([100,300,300,100],[105,45,165,105
  set_color(0,0,255)
  draw_poly([100, ___, ___, 100],[105, ___, ___
```

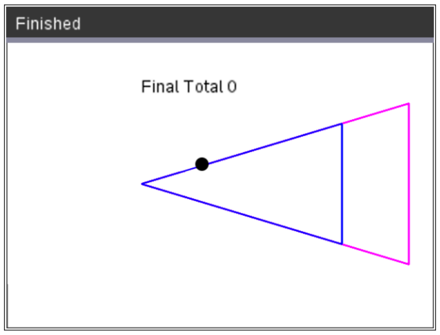
(You shouldn't have ___ in your

code. You should have the correct values you found in step 15).

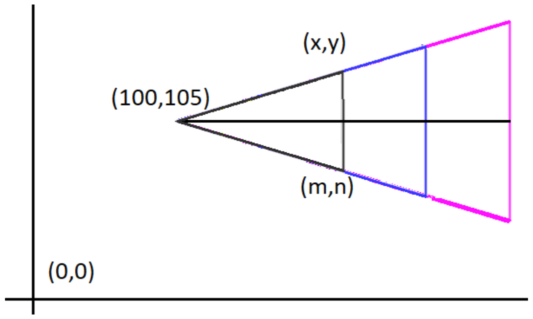
17. Execute your code (ctrl + r). Enter "1" for the number of shots.

    Does it give you the picture to the right?
    (Your disk could be in a different location)

    If you have an error, fix it. If the new triangle isn't correct, re-evaluate the numbers in your last draw_poly.



18. Repeat the same process for the 2nd smallest triangle.
    Look back at step 5 to find the dimensions for the triangle.
    Find the values for the two vertices

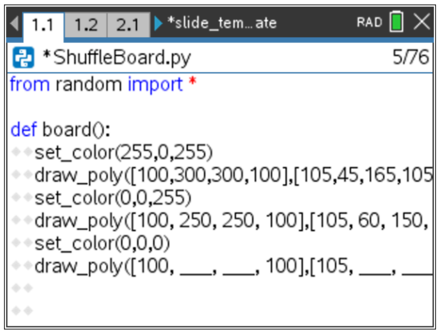


(x,y) = (,   )

(m,n) = (,   )

19. Change your pen color to black or a color of your choosing.
    Plot your new triangle.

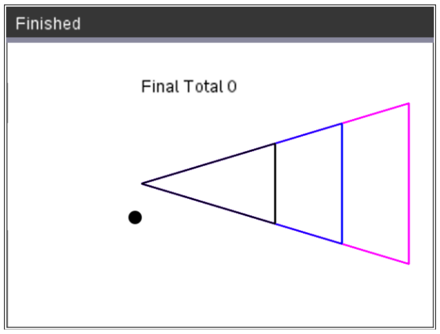
```
set_color(0,0,0)
draw_poly([100, ___, ___, 100],[105, ___, ___, 105])
```



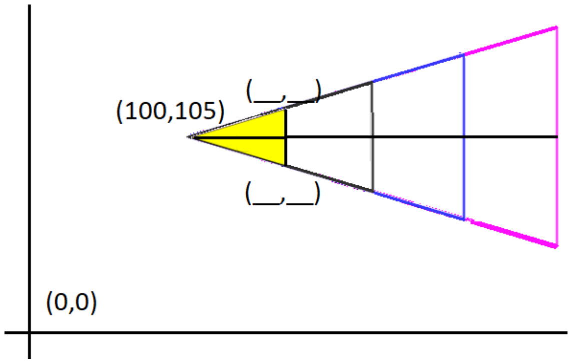
```
from random import *

def board():
  set_color(255,0,255)
  draw_poly([100,300,300,100],[105,45,165,105
  set_color(0,0,255)
  draw_poly([100, 250, 250, 100],[105, 60, 150,
  set_color(0,0,0)
  draw_poly([100, ___, ___, 100],[105, ___, ___
```

(You shouldn't have ___ in your code. You should have the correct values you found in step 18).

Execute your code. Verify the new triangle is in the correct location.

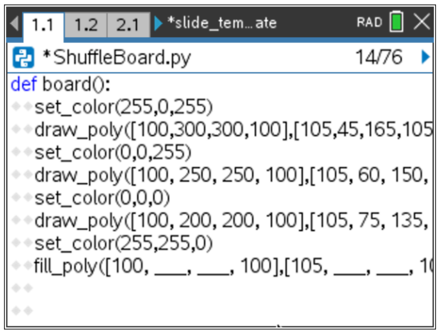


20. Repeat the same process for the smallest triangle.
    Look back at step 8 to find the dimensions for the triangle.
    Find the values for the two vertices



21. Instead of draw_poly, you will use fill_poly to fill in the triangle.
    If r = 255 and g = 255, set_color(255,255,0) will produce yellow.
    You may choose a different color if you like.

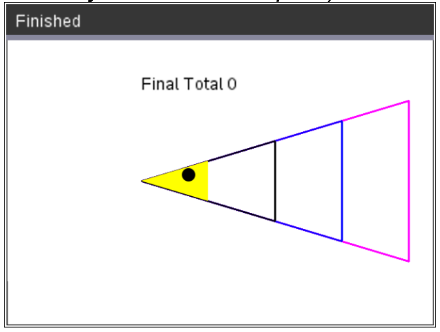
    set_color(255,255,0)
    fill_poly([100, ___, ___, 100],[105, ___, ___, 105])



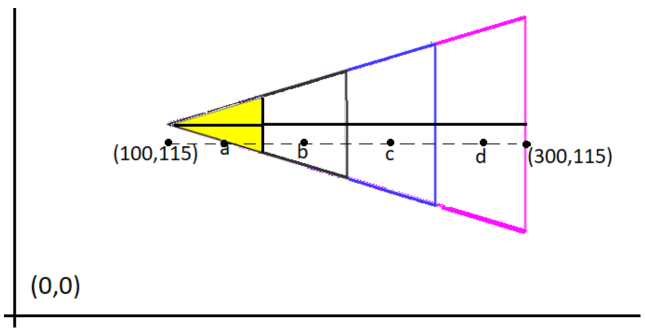
```
def board():
    set_color(255,0,255)
    draw_poly([100,300,300,100],[105,45,165,105
    set_color(0,0,255)
    draw_poly([100, 250, 250, 100],[105, 60, 150,
    set_color(0,0,0)
    draw_poly([100, 200, 200, 100],[105, 75, 135,
    set_color(255,255,0)
    fill_poly([100, ___, ___, 100],[105, ___, ___, 1(
```

(*You shouldn't have ___ in your code. You should have the correct values you found in step 20).*

Verify your code works.



22. Now to label the point values.
    Points a, b, c and d will be halfway between the left and right boundary of the scoring zones. Where should each point lable be placed?



a. (,   )

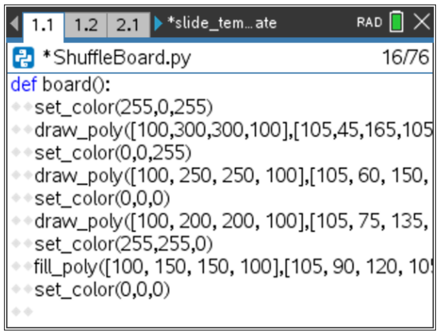
b. (,   )

c. (,   )

d. (,   )

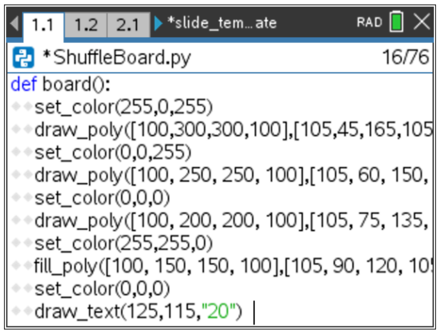
23. Change the pen color back to black.

    set_color(0,0,0)

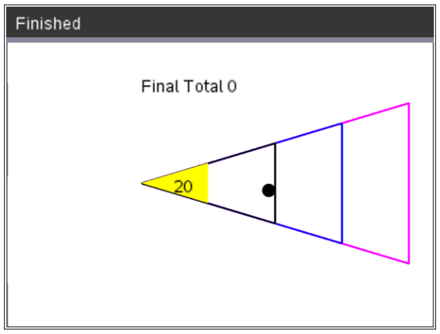


```
1.1  1.2  2.1  *slide_tem…ate   RAD
*ShuffleBoard.py                16/76
def board():
  set_color(255,0,255)
  draw_poly([100,300,300,100],[105,45,165,105
  set_color(0,0,255)
  draw_poly([100, 250, 250, 100],[105, 60, 150,
  set_color(0,0,0)
  draw_poly([100, 200, 200, 100],[105, 75, 135,
  set_color(255,255,0)
  fill_poly([100, 150, 150, 100],[105, 90, 120, 10
  set_color(0,0,0)
```

24. The command draw_text(x,y,string) will draw on the screen. To draw the first value type draw_text(125,115,”20”)

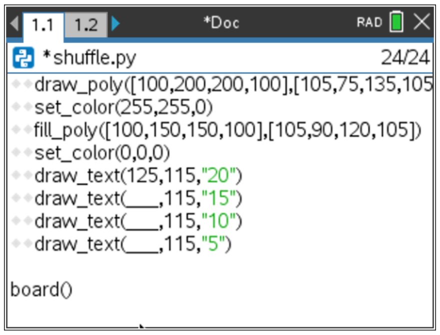


```
1.1  1.2  2.1  *slide_tem…ate   RAD
*ShuffleBoard.py                16/76
def board():
  set_color(255,0,255)
  draw_poly([100,300,300,100],[105,45,165,105
  set_color(0,0,255)
  draw_poly([100, 250, 250, 100],[105, 60, 150,
  set_color(0,0,0)
  draw_poly([100, 200, 200, 100],[105, 75, 135,
  set_color(255,255,0)
  fill_poly([100, 150, 150, 100],[105, 90, 120, 10
  set_color(0,0,0)
  draw_text(125,115,"20")
```



25. Use the values you found in step 22 to draw the remaining values.

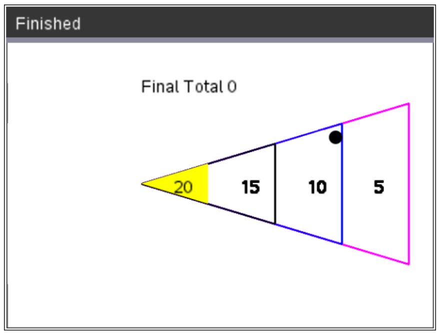
    draw_text(_____,115,"15")
    draw_text(_____,115,"10")
    draw_text(_____,115,"5")



```
1.1  1.2      *Doc              RAD
*shuffle.py                     24/24
  draw_poly([100,200,200,100],[105,75,135,105
  set_color(255,255,0)
  fill_poly([100,150,150,100],[105,90,120,105])
  set_color(0,0,0)
  draw_text(125,115,"20")
  draw_text(___,115,"15")
  draw_text(___,115,"10")
  draw_text(___,115,"5")

board()
```

(You shouldn’t have ___ in your code. You should have the correct values you found in step 22).

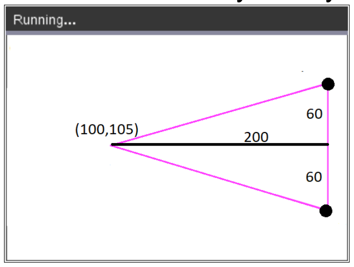
Execute your code. Make sure the labels appear in the correct location.

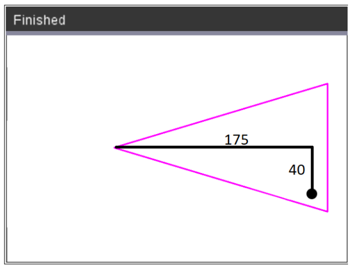


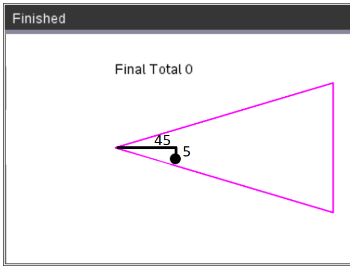
26. Your game is almost complete. All you need to do is code the scoring mechanism. How do you score points? How do you know if the disc is inside a scoring region or outside a scoring region?

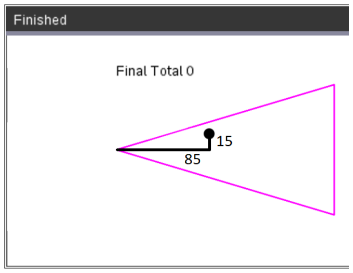
    Let's look for a scoring pattern.

    Caculate the relationship between the distance the disc travels past the closest vertex on the x-axis and the distance from the line of symmetry.

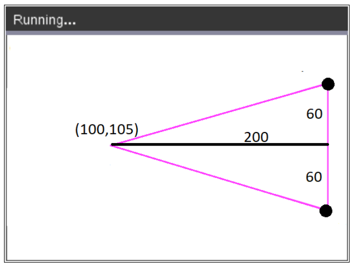


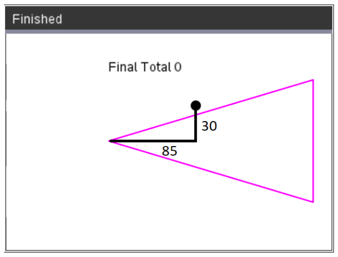


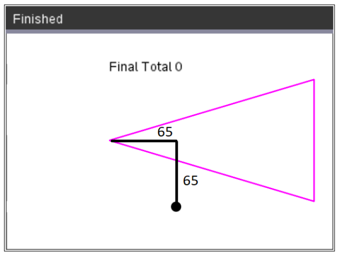


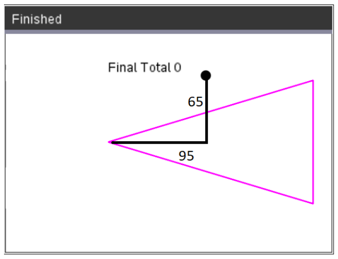


$\frac{60}{200} = 0.3$ ——≈ ——≈ ——≈









$\frac{60}{200} = 0.3$ ——≈ ——≈ ——≈

Look at all the shots that ***would*** earn points. How do they compare to the gameboard boundary of 0.3?

Look at all the shots that would ***not*** earn points. How do they compare to the gameboard boundary of 0.3?

Complete the following statement.

In order to earn points,

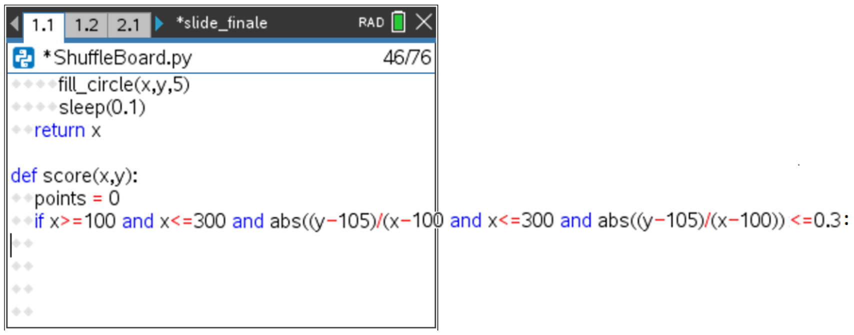
the disc x value must be greater than or equal to 100, less than or equal to _________

and $\frac{y\ distatance\ past\ 105}{x\ distance\ from\ 100}$ must be ____________

27. Now to translate your word statement above, called ***psuedo code*** into Python.

if x >= 100 and x <= 300 and abs((y – 105)/(x – 100)) <= 0.3:

Add this line of code to the score function. Make sure you include a colon, : , at the end of the line.



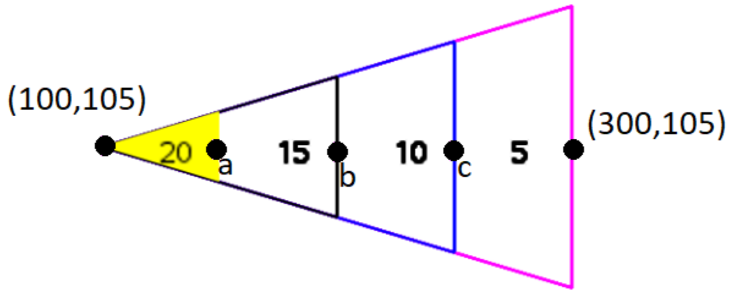
```
    fill_circle(x,y,5)
    sleep(0.1)
  return x

def score(x,y):
  points = 0
  if x>=100 and x<=300 and abs((y−105)/(x−100 and x<=300 and abs((y−105)/(x−100)) <=0.3:
```

if
menu → Built-ins → Control → if

>=, <=, and
[ctrl] [=]

abs
You need the absolute value function to measure the distance from the line of symmetry.

If a disc is within the boundaries to score points, how many points should the user get?



To earn 20 points, the x value should be between _______ and _____.

To earn 15 points, the x value should be between _______ and _____.

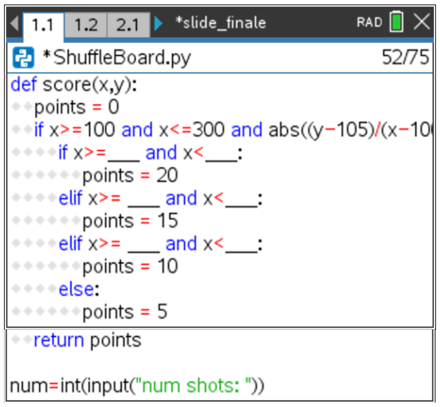
To earn 10 points, the x value should be between _______ and _____.

To earn 5 points, the x value should be between _______ and _____.

Explain or show how you found the boundaries above.

28. Use your answers from step 27 to fill in the Python template:

```
if x >= ___ and x < ___:
    points = 20
elif x >= ___ and x < ___:
    points = 15
elif x >= ___ and x < ___:
    points = 10
else:
    points = 5
```



```
def score(x,y):
  points = 0
  if x>=100 and x<=300 and abs((y−105)/(x−10
    if x>=___ and x<___:
      points = 20
    elif x>= ___ and x<___:
      points = 15
    elif x>= ___ and x<___:
      points = 10
    else:
      points = 5
  return points

num=int(input("num shots: "))
```

(You shouldn't have ___ in your code, you should have the correct values you found in step 27).

Notice the indenting in the picture on the right. The new ***if***, ***elif, else*** statements need indented two spaces from the original if. The ***points =*** lines are indented two additional spaces from the if, elif, else statements.

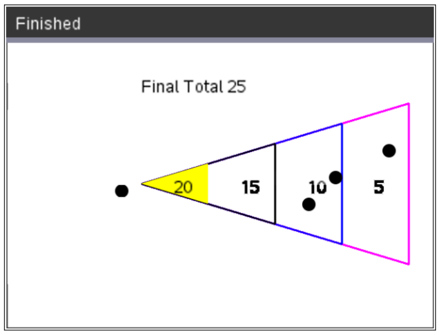
29. Execute your code.

    Play your game ***three*** times.

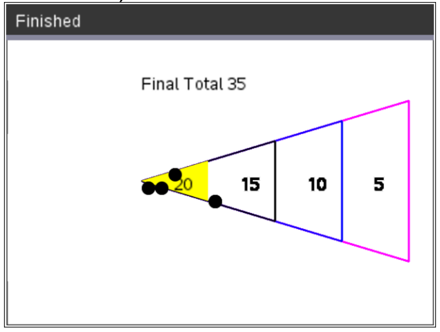
    Each time enter **4** shots.

    Verify your boundaries create the correct score.
    If they are not correct, re-evaluate your work in step 27.

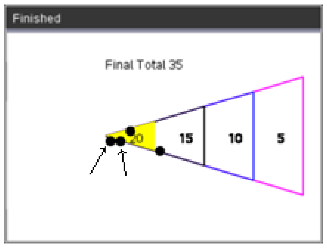
    Two example runs are shown on the right.

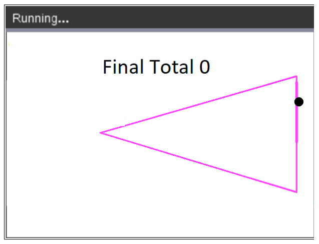


(We will discuss this case in the next step, why might this score be an issue?)

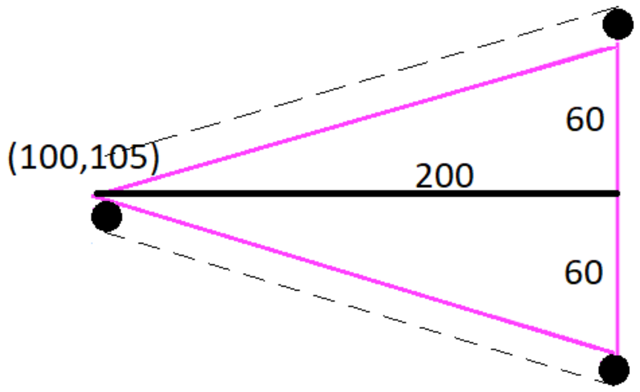


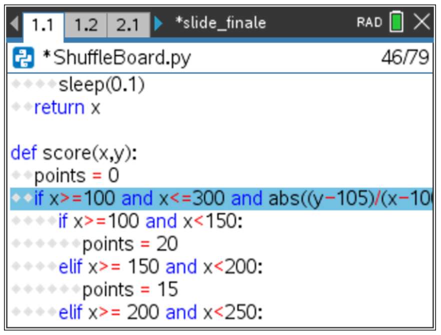
30. The code was written using the **CENTER** of the disk and the assumption it would land **inside** the triangle. The picture to the right has two discs with centers *outisde* the triangle, however, parts of the circles overlap the triange. These two discs are not counted in the score.The same issue happens if the center goes past the large triangle. How might we adjust the code to fix this?





31. The disc radius is 5 pixels.
That means, instead of using the ratio $\frac{60}{200} = 0.3$, we should use $\frac{65}{200} = 0.325$.



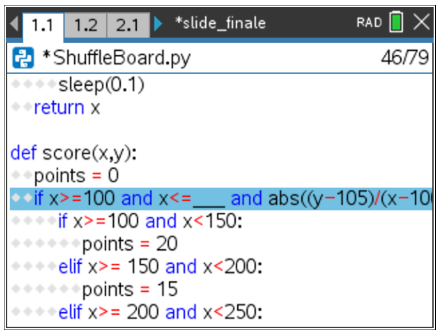


```
1.1  1.2  2.1  *slide_finale                RAD
*ShuffleBoard.py                           46/79
    sleep(0.1)
  return x

def score(x,y):
  points = 0
  if x>=100 and x<=300 and abs((y-105)/(x-10
    if x>=100 and x<150:
      points = 20
    elif x>= 150 and x<200:
      points = 15
    elif x>= 200 and x<250:
```

Modify the first if statement in the score function to accommodate this adjustment.

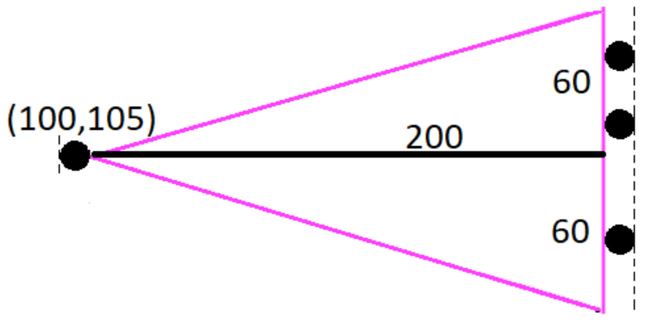
32. Instead of making 300 the maximum x value, what should it be?
Change the maximum x value from 300 to the new x value.

Instead of making 100 the minimum x value, what should it be?
Change the minimum x value from 100 to the new x value.



```
1.1  1.2  2.1  *slide_finale                RAD
*ShuffleBoard.py                           46/79
    sleep(0.1)
  return x

def score(x,y):
  points = 0
  if x>=100 and x<=___ and abs((y-105)/(x-10
    if x>=100 and x<150:
      points = 20
    elif x>= 150 and x<200:
      points = 15
    elif x>= 200 and x<250:
```

(You should have the new value where the ___ is in the highlighted line)



33. Play your game several times. Make sure the modifications we made to the code work correctly.