

**Integer Darts**

In this project, you will create an integer dart game. Part of the code has already been written in the “Integer Darts Template.tns” file. You will write the code to generate integer addition, subtraction, multiplication and division problems. You will also write the code to determine the score. Each correctly answered integer question will earn a dart.

**Objectives:**

**Programming Objectives:**

- Use variables to store values
- Use the randint() function to generate integers
- Use the print() function to display
- Use a while loop to repeat code.
- Use an if..elif statements to make decisions

**Math Objectives:**

- Add and subtract integers
- Multiply and divide integers.
- Use Pythagorean Theorem to find distance between two points

**Math Course Connections: Middle School Mathematics**

In this project, you will create an integer dart game. Part of the code has already been written in the “Integer Darts Template.tns” file. You will write the code to generate integer addition, subtraction, multiplication and division problems. You will also write the code to determine the score. Each correctly answered integer question will earn a dart.

```

Python Shell 3/3
>>>#Running integerDarts.py
>>>from integerDarts import *
-81 / -9 = |
    
```

Ask an integer addition, subtraction, multiplication or division problem

```

Python Shell 7/7
>>>#Running integerDarts.py
>>>from integerDarts import *
-81 / -9 = 9
correct
--darts-- 1
>>>
-2 + 14 =
    
```

Correct answer earns a dart.

```

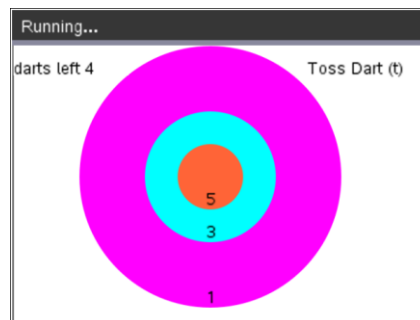
Python Shell 11/11
>>>#Running integerDarts.py
>>>from integerDarts import *
-81 / -9 = 9
correct
--darts-- 1
>>>
-2 + 14 = 12
correct
--darts-- 2
>>>
4 * -3 = |
    
```

Another correct answer, another dart earned.

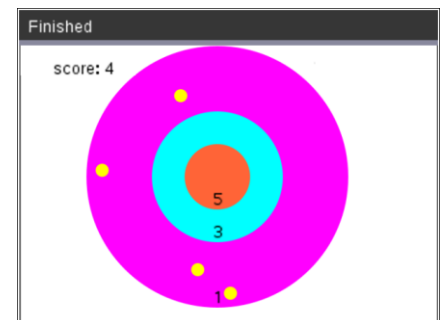
```

Python Shell 15/15
--darts-- 1
>>>
-2 + 14 = 12
correct
--darts-- 2
>>>
4 * -3 = 12
sorry -12
--darts-- 2
>>>
-8 + -8 =
    
```

Incorrect answer. Correct answer displayed. Doesn't earn a dart.



After 5 questions, the dart board appears. Player throws darts.



After all the darts are played, the final score is displayed.



1. Obtain the “Integer Darts Template.tns” from your teacher. Part of the programming code has been coded for you.
2. Let’s examine the code template.

Libraries needed for the project

```

from ti_draw import *
from random import *
from math import *
from time import *
from ti_system import *

```

\*You write code to create two random integers. Randomly determine an operation. Display the question. Check the answer. Add darts if the answer is correct.

```

darts=0
for i in range(5):

```

Display earned darts each round

```

print("--darts--",darts)
print("")
print("You earned", darts,"points")
sleep(4)

```

Create the dart board.

```

set_color(255,0,255)
fill_circle(150,100,100)
set_color(0,255,255)
fill_circle(150,100,50)
set_color(255,100,55)
fill_circle(150,100,25)
set_color(0,0,0)
draw_text(148,200,"1")
draw_text(148,150,"3")
draw_text(148,125,"5")

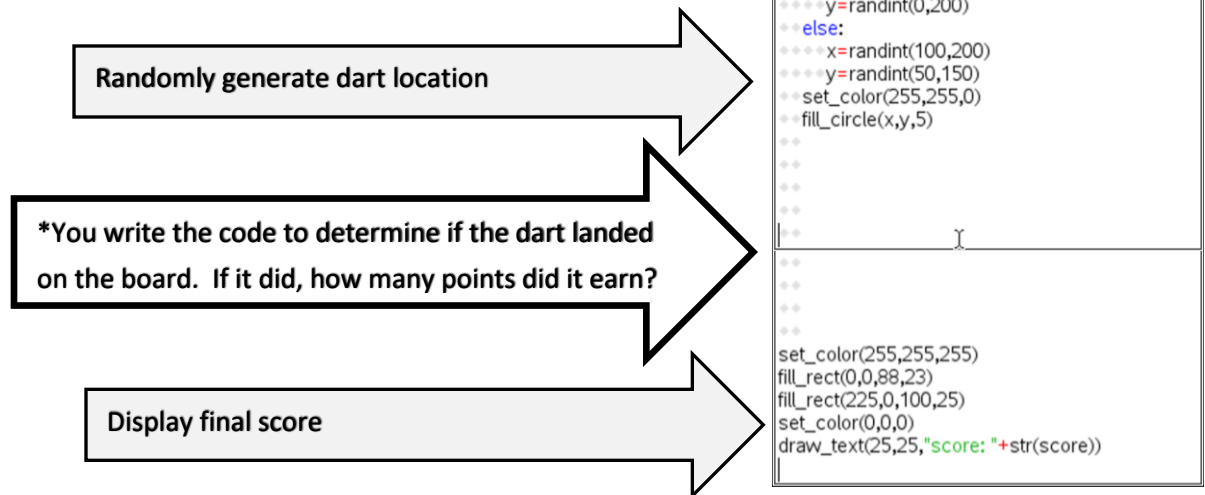
```

For each round, display the darts left, wait for the player to press “t” to toss

```

score = 0
draw_text(225, 25, "Toss Dart (t)")
for i in range(darts):
    set_color(255,255,255)
    fill_rect(0,0,88,23)
    set_color(0,0,0)
    draw_text(0,25,"darts left " + str(darts-i))
    while get_key() != "t":
        continue
    if darts%2==0:
        x=randint(50,250)

```



3. Below are examples of all the types of equations your game will create.

- |               |              |              |               |               |
|---------------|--------------|--------------|---------------|---------------|
| a. $-20 / -5$ | b. $13 - -8$ | c. $-3 * -7$ | d. $-7 - 15$  | e. $8 * -3$   |
| f. $8 * 3$    | g. $9 + -8$  | h. $-48 / 6$ | i. $42 / 6$   | j. $-1 + -18$ |
| k. $3 - 17$   | l. $-10 * 4$ | m. $-8 + 6$  | n. $-6 - -10$ | o. $72 / -8$  |

4. Evaluate equation expression above without a calculator. Use your calculator to verify your answers.

5. The first two lines of code will generate two random integers n1 and n2. Initially, they will be any integer between -10 and 10.

Add the lines:

```

n1 = randint(-10,10)
n2 = randint(-10,10)

```

\*\*randint -- You can type randint or you can find it in the menu menu → random → randint

```

IntegerDarts.py 11/76
from random import *
from math import *
from time import *
from ti_system import *

darts=0

for i in range(5):
n1 = randint(-10,10)
n2 = randint(-10,10)

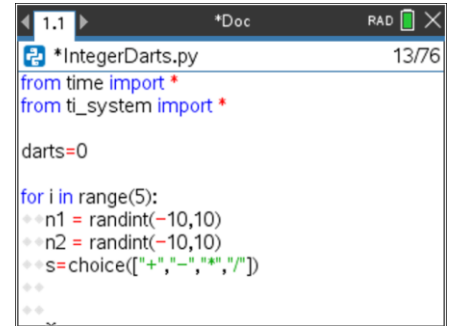
```

6. Next, randomly select the operation. The function `choice()` lets you enter a list of possibilities. It then selects one item from the list. Notice the function has parenthesis `()`, the list starts and ends with square brackets `[]`.

Add the line:

```
s = choice(["+", "-", "*", "/"])
```

**\*\*choice** -- You can type `choice` or you can find it in the menu  
 menu → random → choice



```
1.1 *IntegerDarts.py 13/76
from time import *
from ti_system import *

darts=0

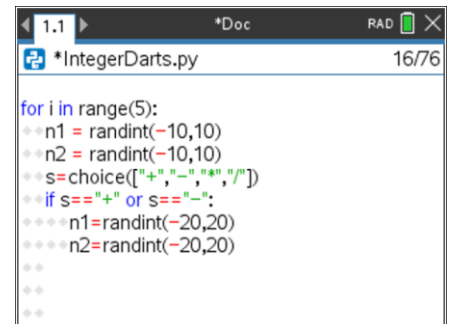
for i in range(5):
  n1 = randint(-10,10)
  n2 = randint(-10,10)
  s=choice(["+", "-", "*", "/"])
  **
```

7. If the choice is an addition or subtraction, we'll make a wider range of integers possible. We'll let `n1` and `n2` be anything from -20 to 20. This will require an `if` statement.

Python uses `==` to check IF two quantities are equivalent. By itself, the `=` sign assigns the variable on the left the value on the right. Notice how `==` and `=` are used in the statements below.

```
if s == "+" or s == "-":
    n1 = randint(-20,20)
    n2 = randint(-20,20)
```

**\*\*if** menu → built-ins → control → if



```
1.1 *IntegerDarts.py 16/76
for i in range(5):
  n1 = randint(-10,10)
  n2 = randint(-10,10)
  s=choice(["+", "-", "*", "/"])
  if s == "+" or s == "-":
    n1 = randint(-20,20)
    n2 = randint(-20,20)
  **
```

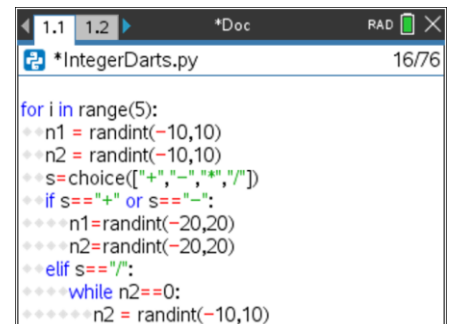
8. If the sign was a division sign, you need to ensure you don't divide by 0. While `n2` is a zero, you will generate a new integer value.

You only need to worry about 0 if the sign is `/`.

Therefore, the code will be

```
elif s == "/":
    while n2 == 0:
        n2 = randint(-10,10)
```

**\*\*while** menu → built-ins → control → while



```
1.1 1.2 *IntegerDarts.py 16/76
for i in range(5):
  n1 = randint(-10,10)
  n2 = randint(-10,10)
  s=choice(["+", "-", "*", "/"])
  if s == "+" or s == "-":
    n1 = randint(-20,20)
    n2 = randint(-20,20)
  elif s == "/":
    while n2 == 0:
      n2 = randint(-10,10)
  **
```

9. If the sign was a division sign, we want to ensure the answer will be an integer. For example, we don't want questions like  $-12 / -7$  or  $-8 / 3$ . We want questions like  $-12 / -4$  or  $42 / -6$  because they result in an integer.

Notice the `n2=randint(-10,10)` is indented from the while. The `n1 = n1*n2` line lines up below the `while`.

```
n1 = n1*n2
```

`**elif` menu → built-ins → control → elif

10. Now to construct the question. We will *concatenate*, put together, the *integers* `n1` and `n2` with the *string* symbol. To put items together, they must all be of the same data type. You will use `str(n1)` and `str(n2)` to convert the *integers* to *string*.

```
prob = str(n1) + s + str(n2)
```

`**str()` menu → built-ins → type → str

11. You are now ready to ask the user for the answer. Python uses the function *input* to get information from the user and store it as a string. You will use `int(input())` to get information and store it as an integer.

```
answer = int(input(prob + "= "))
```

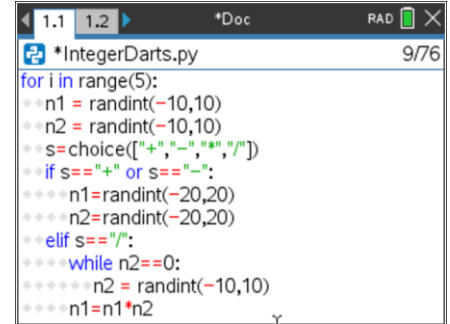
`**int()` menu → built-ins → type → int

`**input()` menu → built-ins → i/o → input

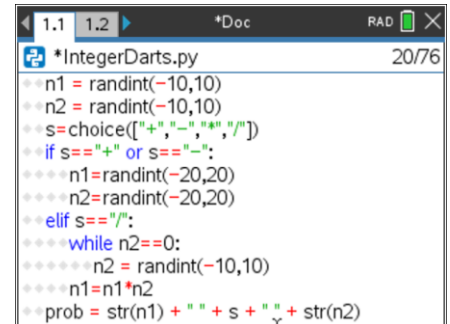
12. If the user's answer matches the evaluated problem, the user will earn a dart. To add one to the darts total you could write `darts = darts + 1`. Python has a shortcut however, that is easier to type `darts += 1`. Print "correct".

```
if answer == eval(prob):
    darts += 1
    print("correct")
```

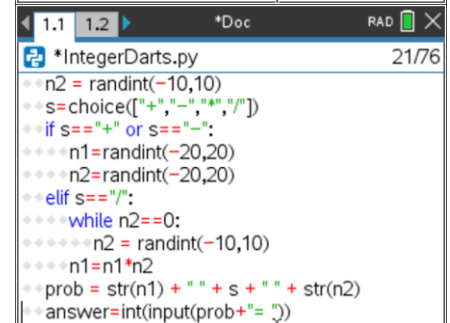
`**eval` menu → built-ins → i/o → eval



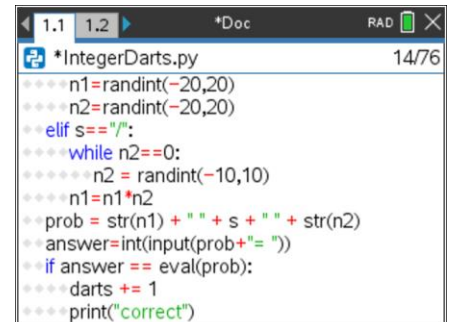
```
*IntegerDarts.py 9/76
for i in range(5):
    n1 = randint(-10,10)
    n2 = randint(-10,10)
    s = choice(["+", "-", "*", "/"])
    if s == "+" or s == "-":
        n1 = randint(-20,20)
        n2 = randint(-20,20)
    elif s == "/":
        while n2 == 0:
            n2 = randint(-10,10)
        n1 = n1*n2
```



```
*IntegerDarts.py 20/76
n1 = randint(-10,10)
n2 = randint(-10,10)
s = choice(["+", "-", "*", "/"])
if s == "+" or s == "-":
    n1 = randint(-20,20)
    n2 = randint(-20,20)
elif s == "/":
    while n2 == 0:
        n2 = randint(-10,10)
    n1 = n1*n2
prob = str(n1) + s + str(n2)
```



```
*IntegerDarts.py 21/76
n2 = randint(-10,10)
s = choice(["+", "-", "*", "/"])
if s == "+" or s == "-":
    n1 = randint(-20,20)
    n2 = randint(-20,20)
elif s == "/":
    while n2 == 0:
        n2 = randint(-10,10)
    n1 = n1*n2
prob = str(n1) + s + str(n2)
answer = int(input(prob + "= "))
```



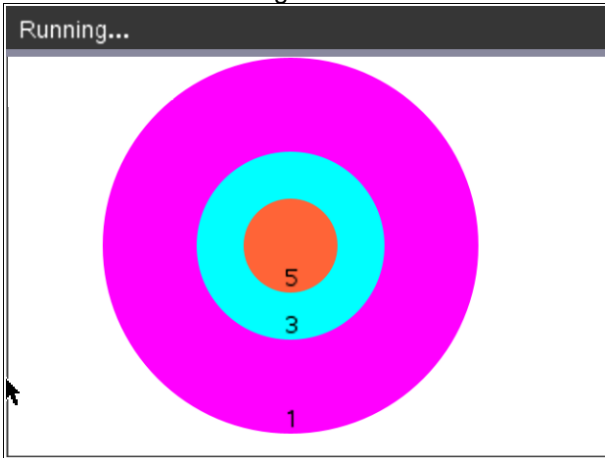
```
*IntegerDarts.py 14/76
n1 = randint(-20,20)
n2 = randint(-20,20)
elif s == "/":
    while n2 == 0:
        n2 = randint(-10,10)
    n1 = n1*n2
prob = str(n1) + s + str(n2)
answer = int(input(prob + "= "))
if answer == eval(prob):
    darts += 1
    print("correct")
```

13. If the answer isn't true, the only option is false. Instead of using an elif like you did a few steps ago, use an else.

```
else:
    print("sorry, ", eval(prob))
```

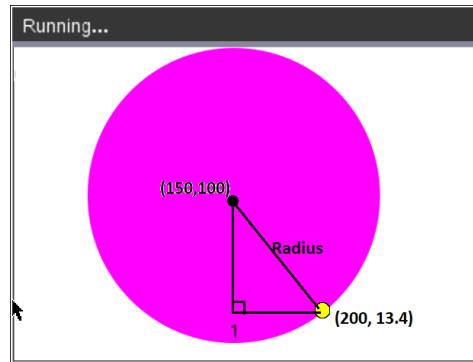
```
*IntegerDarts.py 22/76
while n2==0:
    n2 = randint(-10,10)
    n1=n1*n2
    prob = str(n1) + " * " + str(n2)
    answer=int(input(prob+" = "))
    if answer == eval(prob):
        darts += 1
        print("correct")
    else:
        print("sorry",eval(prob))
```

14. Now to code the scoring section.

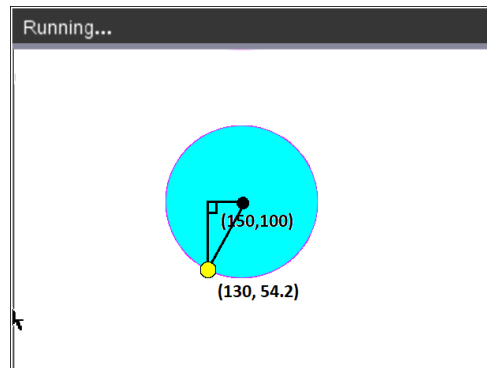


The target is centered at (150,100).

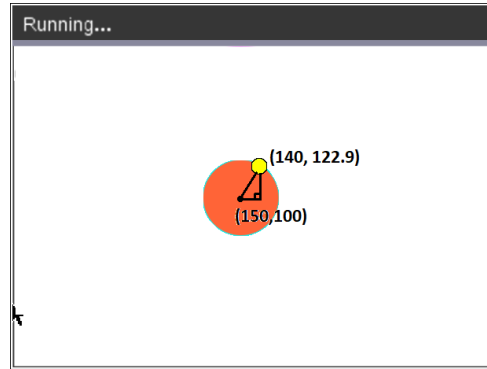
If a dart lands at (200,13.4), it is barely inside the target. Find the radius for the largest region.  
 (You can use the scratchpad for calculations.)



15. If a dart lands at (130, 54.2), It is barely inside the middle target. Find the radius for the middle region.



16. If a dart lands at (140, 122.9). It is barely inside the smallest target. Find the radius for the smallest region.



17. Fill in the blanks below with words or numbers.

To find the distance the dart lands from the center use \_\_\_\_\_

if the distance is less than or equal to \_\_\_\_\_

give 5 points because it is in the **smallest** circle

elif the distance is less than or equal to \_\_\_\_\_

give 3 points because it is in the **middle** circle

elif the distance is less than or equal to \_\_\_\_\_

give 1 point because it is in the **largest** circle

18. Now to put the words called *pseudo code* from step 16 into Python syntax.

Scroll down to the next missing section of code.

```

1.1 *Doc RAD 61/75
IntegerDarts.py
++++ continue
if darts%2==0:
++++ x=randint(50,250)
++++ y=randint(0,200)
else:
++++ x=randint(100,200)
++++ y=randint(50,150)
set_color(255,255,0)
fill_circle(x,y,5)
|

```

19. To find the missing radius for each circle you used the Pythagorean Theorem:

$$\text{leg1}^2 + \text{leg2}^2 = \text{hypotenuse}^2$$

$$(x - 150)^2 + (y - 100)^2 = \text{radius}^2$$

$$\sqrt{(x - 150)^2 + (y - 100)^2} = \text{radius}$$

The distance the dart lands from the center needs to be less than or equal to the radius of the circle.

Python uses `**2` instead of `^2` to square numbers. The function `sqrt()` is used for square root.

Add

$$\text{dist} = \text{sqrt}((x-150)**2+(y-100)**2)$$

```

1.1 *Doc RAD 61/75
IntegerDarts.py
++++ continue
if darts%2==0:
++++ x=randint(50,250)
++++ y=randint(0,200)
else:
++++ x=randint(100,200)
++++ y=randint(50,150)
set_color(255,255,0)
fill_circle(x,y,5)
dist=sqrt((x-150)**2+(y-100)**2)

```

`**sqrt` menu → math → sqrt

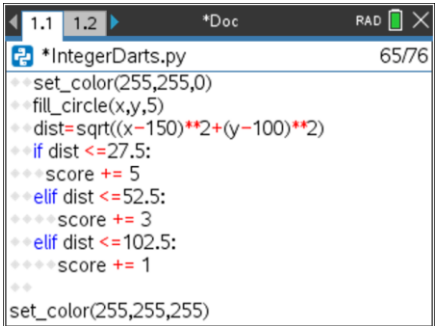
20. Your *pseudo* code found above said:

if the distance is less than or equal to **25**:  
    give 5 points because it is in the ***smallest*** circle  
elif the distance is less than or equal to **50**:  
    give 3 points because it is in the ***middle*** circle  
elif the distance is less than or equal to **100**:  
    give 1 point because it is in the ***largest*** circle

The dart has a width of 2.5 pixels. This gives 2.5 more pixels to the scoring region. Therefore, add the following:

```
if dist <= 27.5:  
    score += 5  
elif dist <= 52.5:  
    score += 3  
elif dist <= 102.5:  
    score += 1
```

21. Congratulations! You have typed all the code. Press [ctrl] [r] to execute the code. If you don't have any errors, you should be able to play the game. If your code has errors, fix the errors, then play the game.



```
1.1 1.2 *Doc RAD 65/76  
*IntegerDarts.py  
set_color(255,255,0)  
fill_circle(x,y,5)  
dist=sqrt((x-150)**2+(y-100)**2)  
if dist <= 27.5:  
    score += 5  
elif dist <= 52.5:  
    score += 3  
elif dist <= 102.5:  
    score += 1  
set_color(255,255,255)
```