

Integer Quadratic Zeros

In this lesson, you will learn how to find the x-intercepts for equations in the form $y = (x - x_1)(x - x_2)$.

You will explore how factored equations in the form $y = (x - x_1)(x - x_2)$ can be rewritten in a standard form $y = x^2 + bx + c$.

You will create a game that will let you practice finding integer x-intercepts for equations of the form $y = x^2 + bx + c$.

Objectives:

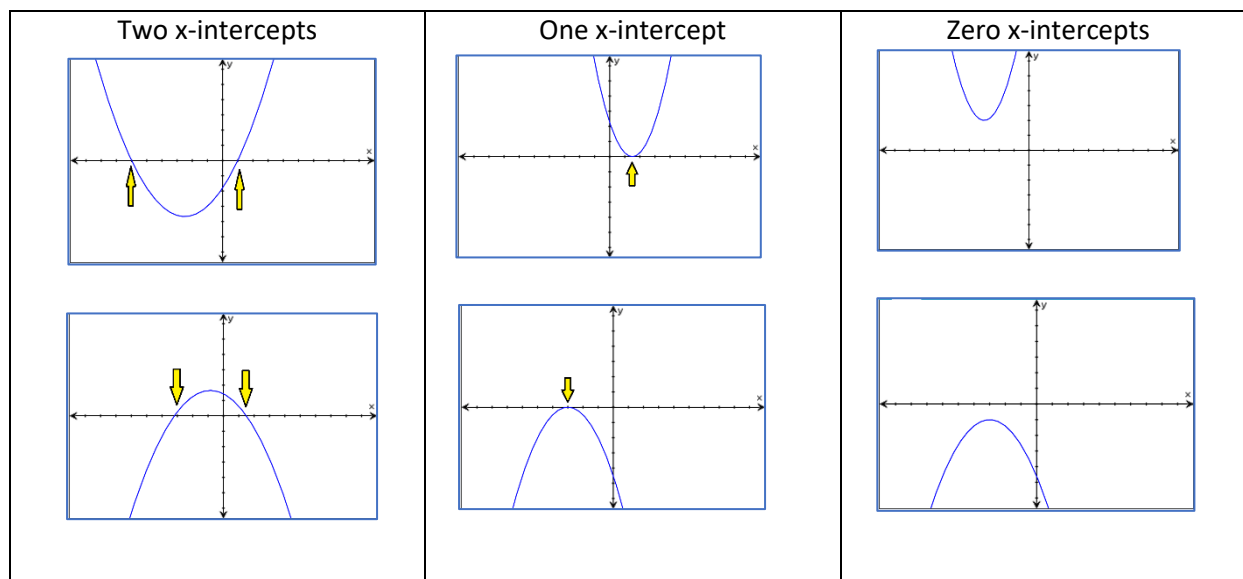
Programming Objectives:

- Use the input function and a variable to collect and store data from a user
- Use the randint() function to generate random integers.
- Use a while loop to repeat code
- Use if..elif..else statements to make decisions.

Math Objectives:

- Explore how x-intercepts are related to factored quadratic equations
- Explore how to factor equation in standard form
- Factor quadratic equations with integer solutions

The standard form of a quadratic equation is $y = ax^2 + bx + c$ where a, b, and c are real numbers. Quadratic equations can cross the x-axis twice, once or not at all. The value of a, b and c determine the number of x-intercepts.



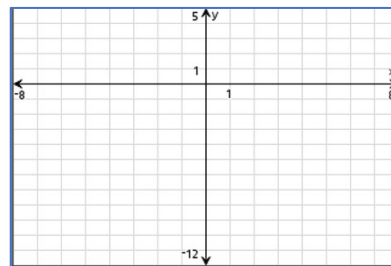
If the graph of the quadratic equation crosses the x-axis, the x values for the x-intercepts can be any integer such as 3, -5 and 0 or any other real number such as 1.3, $\frac{-5}{7}$, or $\sqrt{3}$. This activity will focus on quadratic equations with integer x-intercepts.



Explorations:

1. Graph the equation $y = (x + 4)(x - 2)$ on your calculator.
Copy the graph on the grid to the right.

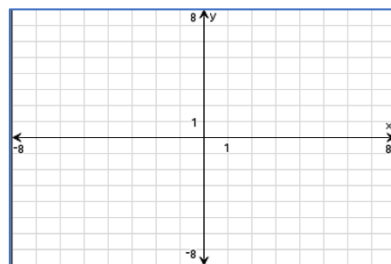
Where are the x-intercepts? _____



2. Graph the equation $y = (x - 1)(x - 3)$ on your calculator.

Copy the graph on the grid to the right.

Where are the x-intercepts? _____



3. Look at your x-intercepts and the equations for the previous two problems. Do you notice similarities between the x-intercepts and the equations?

4. Let $y = (x + 5)(x - 1)$. Use your results from above to predict where the x-intercepts will be on the graph.

Why does this relationship exist between the factored form of an equation and the x-intercepts? The answer lies with the number 0.

5. List two numbers that multiply to zero. ____ and ____

List another pair of numbers that multiply to zero. ____ and ____

List another pair of numbers that multiply to zero. ____ and ____

Look at your answers above.

Did you list zero in each pair?

6. If two numbers multiply to zero, at least one of those two numbers must be a zero.
You will use this fact to find the x-intercepts for quadratics.



The x-intercepts have a y-value of 0.

Therefore, the x-intercepts occur at $0 = (x + 5)(x - 1)$.

The value of $(x + 5)$ times $(x - 1)$ must equal 0.

Using this fact, either $x + 5 = 0$ or $x - 1 = 0$.

Solve each equation for x. $x = \underline{\hspace{2cm}}$ $x = \underline{\hspace{2cm}}$

Compare these values with your predictions from #4.

7. If the x-intercepts are at $x = -2$ and $x = 7$, what is the factored form for the quadratic equation?

$$y = (\quad) (\quad)$$

Graph your equation and verify it crosses the x-axis at $x = -2$ and $x = 7$.

How do you find the x-intercept for a quadratic if it is not in factored form?

8. You know that if a graph has x-intercepts at $x = 6$ and $x = -3$, you can write it as

$$y = (x - 6)(x + 3)$$

Use distribution to write the equation in standard form.

standard form $y = \underline{\hspace{4cm}}$

Graph your standard form equation on your calculator.

Does it have x-intercepts at $x = 6$ and $x = -3$?

Compare the standard form with the factored form.

How does the coefficient for x in the standard form relate to the x-intercepts?

How does the constant in the standard form compare to the x-intercepts?

After completing the activity above, you know:

Given two real x-intercepts, x_1 and x_2 , you can write the factored form of a quadratic as

$$y = (x - x_1)(x - x_2)$$

You can distribute the values in this equation to get the standard form $y = x^2 + bx + c$.

How do the x-intercepts relate to the constant b ?

How do the x-intercepts relate to the constant c ?

You will use this to create a quadratic factoring game.

The game will generate a quadratic equation with integer x-intercepts. It will display the quadratic equation and ask for the x-intercepts.

```
Python Shell 5/5
>>>#Running QuadraticZeros.py
>>>from QuadraticZeros import *
Find the zeros for
x^2 - 1 x - 12 = 0
x1 = |
```

The user will

*factor the given equation

$$y = (x - 4)(x + 3)$$

*Find and enter the x-intercepts.

$$x = 4 \text{ and } x = -3$$

If the user is correct, say correct.

```
Python Shell 11/11
>>>#Running QuadraticZeros.py
>>>from QuadraticZeros import *
Find the zeros for
x^2 - 1 x - 12 = 0
x1 = 4
x2 = -3
Correct!
```

If the user is incorrect, tell the user the correct values for the zeros.

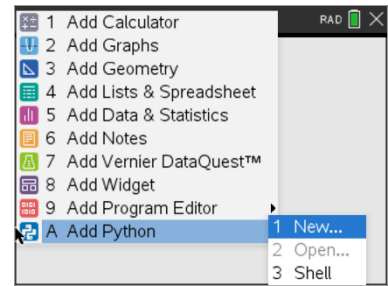
$$y = (x + 8)(x - 1)$$

$$x = -8 \text{ and } x = 1$$

The user forgot the negative on the 8.

```
Python Shell 11/11
>>>#Running QuadraticZeros.py
>>>from QuadraticZeros import *
Find the zeros for
x^2 + 7 x - 8 = 0
x1 = 8
x2 = 1
Sorry should be 1 and -8
```

1. Create a new Python project.

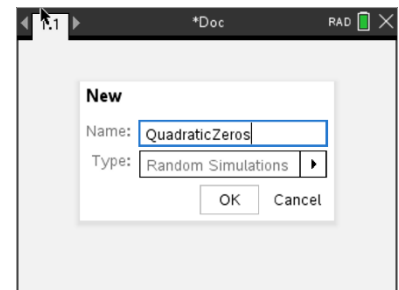


Name the project “**QuadraticZeros**”.

If you like, you can use all lowercase letters.

Do **not** put a space between the words quadratic and zeros. The name of a python project cannot have a space.

You will need to generate random values for the x-intercepts. Therefore, choose the type “Random Simulations”. This will include the libraries needed for the project.



2. You should have the image to the right. If you didn’t select “Random Simulation” from the menu, you’ll have a blank screen.

The lines with the hashtag, #, are comments.

(Did you know another name for the # symbol is octothorpe?)

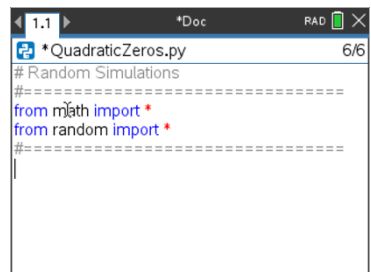
Comments allow the programmer to label sections of the code, making it easier to read. Comments are optional, they do not change the way the program executes.

The two lines that start with the words “from” import two different libraries. Without these libraries, you won’t be able to generate random integers for the random problems.

If you didn’t select Random Simulations, you can import these libraries in two steps.

Menu> Math> from math import *

Menu> Random> from random import *





3. In order to play the game, you will need to generate two random numbers for x-intercepts. The line
- $$x1 = randint(-10,10)$$
- will generate one random integer from -10 to 10 and store it in the variable x1.

You don't need to type all of the code using the keyboard.

The randint() function can be found using

Menu → Random → randint(min,max)

Can you add a line of code to generate another integer value and store that value in x2?

```

1.1 *Doc RAD 6/6
*QuadraticZeros.py
# Random Simulations
#-----
from math import *
from random import *
#-----
x1 = randint(-10,10)

```

4. Frequently checking your code to see if it works properly is a good programming practice. Add a print statement to print the values of x1 and x2 to the screen.

Menu > Built-ins> I/O> print()

After adding the line print(x1,x2) execute your code in the Python shell.

Press **ctrl → r** or **Menu> Run> Run**

```

1.1 *Doc RAD 8/8
*QuadraticZeros.py
# Random Simulations
#-----
from math import *
from random import *
#-----
x1 = randint(-10,10)
x2 = randint(-10,10)
print(x1,x2)

```

Does your code generate two random integers between the values of -10 and 10?

In the example on the right, the numbers -5 and 3 were randomly generated. Press **ctrl → r** a few more times to generate more random number pairs.

```

1.1 1.2 *Doc RAD 4/4
Python Shell
>>>#Running QuadraticZeros.py
>>>from QuadraticZeros import *
-5 3
>>>|

```

5. Now that you have varified the code runs, remove the print(x1,x2) line from your code.
6. Recall the quadratic equation can be written in factored form and standard form.

Factored Form

$$y = (x - x1)(x - x2)$$

Standard Form

$$y = ax^2 + bx + c$$

How is the coefficient “b” related to the factored form?

How is the coefficient “c” related to the factored form?

7. Recall, using distribution you get

$$y = (x - x_1)(x - x_2)$$

$$y = x^2 - x_1x - x_2x + x_1x_2$$

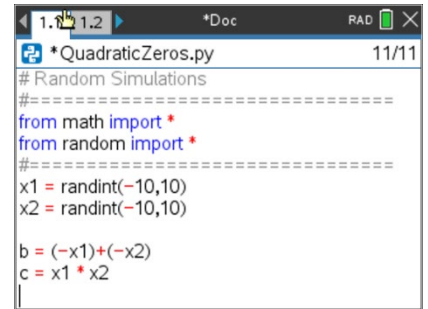
$$y = x^2 + (-x_1 - x_2)x + x_1x_2$$

Therefore, $b = -x_1 - x_2$ and $c = x_1x_2$

Code the two lines:

$$b = -x_1 - x_2$$

$$c = x_1 * x_2$$



```
*QuadraticZeros.py 11/11
# Random Simulations
#-----
from math import *
from random import *
#-----
x1 = randint(-10,10)
x2 = randint(-10,10)


b = (-x1)+(-x2)
c = x1 * x2
```

8. Use the print statement to print the equation.

Recall in step 4, you found print by using the following steps.

Menu → Built-ins → I/O → print()

You will separate string such “x^2” and variables such as b with a comma.

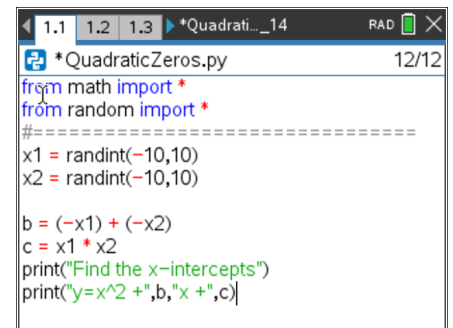
*Hint: The ^ key will add ** to the screen. In Python, ** is the syntax for exponents. To get the ^ symbol, press 

Add the following lines:

```
print("Find the x-intercept")
```

```
print("y=x^2 +",b,"x +",c)
```

*Notice string inside quotes such as “Find the zeros for” and “y=x^2” appear in green. Math operations such as =, *, -, + appear in red.



```
*QuadraticZeros.py 12/12
from math import *
from random import *
#-----
x1 = randint(-10,10)
x2 = randint(-10,10)

b = (-x1) + (-x2)
c = x1 * x2
print("Find the x-intercepts")
print("y=x^2 +",b,"x +",c)
```

9. Run your program a few times. **ctrl** → **r**

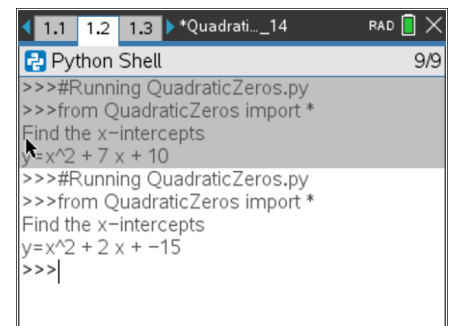
Notice if the b or c coefficient is negative, it displays a plus (+) followed by a negative (-).

The example on the right shows:

$$x^2 + 2x + -15 = 0$$

How could you alter the code so it would display only the minus sign if the operation was add a negative?

$$x^2 + 2x - 15 = 0$$



```
Python Shell 9/9
>>>#Running QuadraticZeros.py
>>>from QuadraticZeros import *
Find the x-intercepts
y=x^2 + 7 x + 10
>>>#Running QuadraticZeros.py
>>>from QuadraticZeros import *
Find the x-intercepts
y=x^2 + 2 x + -15
>>>|
```


10. There are 4 different cases to consider when cleaning up the display code:

- If b and c are both positive:
display doesn't need to change
- If b and c are both negative:
change to + - displays to a -
- If only c is negative:
change the second + - display to a -
- If only b is negative:
change the first + - display to a -

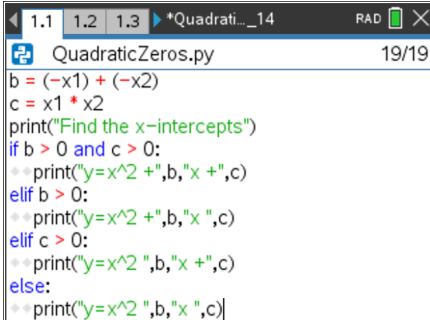
Try fixing the + - print display by writing an if statement in the form:

```
if
elif
elif
else
```

***if Menu -> Built-Ins -> Control**

11. Did you write something similar to:

```
if b > 0 and c > 0:
    print("y=x^2 +",b,"x +",c)
elif b > 0:
    print("y=x^2 +",b,"x ",c)
elif c > 0:
    print("y=x^2 ",b,"x +",c)
else:
    print("y=x^2 ",b,"x ",c)
```

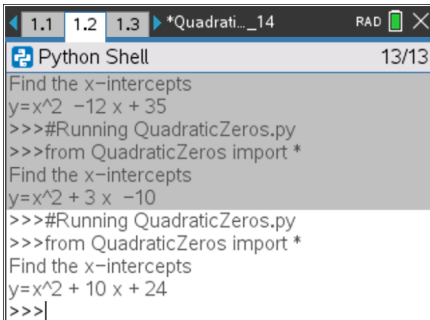


```
1.1 1.2 1.3 *Quadrati..._14 RAD 19/19
QuadraticZeros.py
b = (-x1) + (-x2)
c = x1 * x2
print("Find the x-intercepts")
if b > 0 and c > 0:
    print("y=x^2 +",b,"x +",c)
elif b > 0:
    print("y=x^2 +",b,"x ",c)
elif c > 0:
    print("y=x^2 ",b,"x +",c)
else:
    print("y=x^2 ",b,"x ",c)
```

12. Execute your program at least 5 times. **ctrl** → **r**

Verify the statements printed do not contain + -.

The screen to the right shows three random examples.



```
1.1 1.2 1.3 *Quadrati..._14 RAD 13/13
Python Shell
Find the x-intercepts
y=x^2 -12 x + 35
>>>#Running QuadraticZeros.py
>>>from QuadraticZeros import *
Find the x-intercepts
y=x^2 + 3 x -10
>>>#Running QuadraticZeros.py
>>>from QuadraticZeros import *
Find the x-intercepts
y=x^2 + 10 x + 24
>>>
```



13. Notice the first equation displays

$$x^2 - 12x + 35 = 0$$

Which condition was executed i,ii, iii or iv?

- i) if $b > 0$ and $c > 0$:
 `print("y=x^2 +",b,"x +",c)`
- ii) elif $b > 0$:
 `print("y=x^2 +",b,"x",c)`
- iii) elif $c > 0$:
 `print("x^2",b,"x +",c,"=0 ")`
- iv) else:
 `print("x^2",b,"x",c,"= 0")`

14. Notice the second equation displays

$$x^2 + 3x - 10 = 0$$

Which condition was executed i,ii, iii or iv?

- i) if $b > 0$ and $c > 0$:
 `print("y=x^2 +",b,"x +",c)`
- ii) elif $b > 0$:
 `print("y=x^2 +",b,"x",c)`
- iii) elif $c > 0$:
 `print("x^2",b,"x +",c,"=0 ")`
- iv) else:
 `print("x^2",b,"x",c,"= 0")`

15. Your program generates two x-intercepts, then calculates and displays the standard quadratic equation. Now you need to ask the user to enter the two x-intercepts.

The command `input()` will let you ask a question. By default, the information from an `input()` statement is stored as string. That means the computer thinks the information entered is a list of characters. To store information as integers, you must use `int(input())` to convert the input to an integer.

`int()` can be found using
`input()` can be found using

Menu → Built-Ins → Type
Menu → Built-Ins → I/O

Add the lines:

```
z1 = int(input("x1 = "))
z2 = int(input("x2 = "))
```

```

1.1 1.2 1.3 *Quadrati..._14 RAD 13/13
Python Shell
Find the x-intercepts
y=x^2 -12 x + 35
>>>#Running QuadraticZeros.py
>>>from QuadraticZeros import *
Find the x-intercepts
y=x^2 + 3 x -10
>>>#Running QuadraticZeros.py
>>>from QuadraticZeros import *
Find the x-intercepts
y=x^2 + 10 x + 24
>>>|

```

```

1.1 1.2 1.3 *Quadrati..._14 RAD 13/13
Python Shell
Find the x-intercepts
y=x^2 -12 x + 35
>>>#Running QuadraticZeros.py
>>>from QuadraticZeros import *
Find the x-intercepts
y=x^2 + 3 x -10
>>>#Running QuadraticZeros.py
>>>from QuadraticZeros import *
Find the x-intercepts
y=x^2 + 10 x + 24
>>>|

```

```

1.1 1.2 1.3 *Quadrati..._14 RAD 22/23
*QuadraticZeros.py
if b > 0 and c > 0:
    print("y=x^2 +",b,"x +",c)
elif b > 0:
    print("y=x^2 +",b,"x ",c)
elif c > 0:
    print("y=x^2 ",b,"x +",c)
else:
    print("y=x^2 ",b,"x ",c)

z1 = int(input("x1 = "))
z2 = int(input("x2 = "))

```

16. Recall x_1 and x_2 have the real x -intercepts. The variables z_1 and z_2 have the user's entered x -intercepts. Now you need to check to see if they are equivalent. The order the user types the x -intercepts does not matter. Therefore, the user is correct:

if x_1 equals z_1 and x_2 equals z_2 or x_1 equals z_2 and x_2 equals z_1

In python, the equals sign by itself is used to store a value. To check if two values are equivalent, use two equal signs `==`.

Add the following lines of code:

```
if (x1 == z1 and x2 == z2) or (x1 == z2 and x2 == z1):
    print("Correct!")
else:
    print("Sorry should be",x1,"and",x2)
```

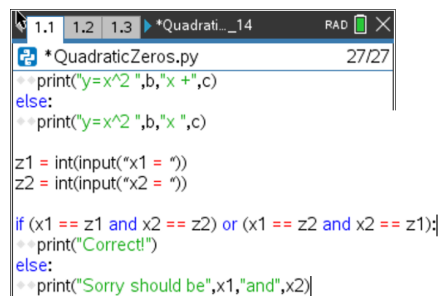
17. Test your program. **ctrl** → **r**

Sample Test:

$$y = x^2 + 14x + 40$$

$$(x + 4)(x + 10) = 0$$

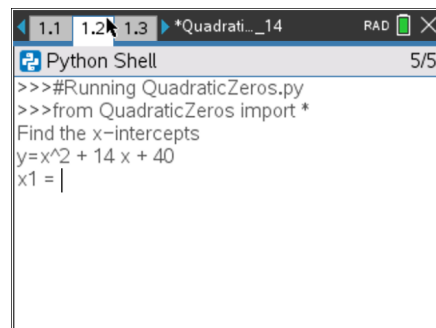
$$x = -4 \quad x = -10$$



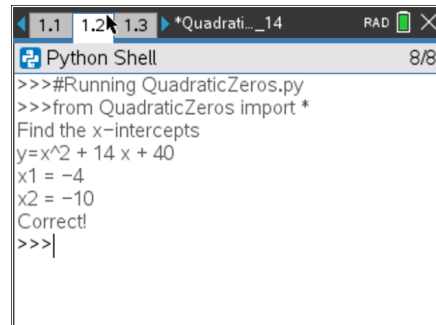
```
*QuadraticZeros.py 27/27
print("y=x^2 ",b,"x +",c)
else:
print("y=x^2 ",b,"x ",c)

z1 = int(input("x1 = "))
z2 = int(input("x2 = "))

if (x1 == z1 and x2 == z2) or (x1 == z2 and x2 == z1):
print("Correct!")
else:
print("Sorry should be",x1,"and",x2)
```



```
Python Shell 5/5
>>>#Running QuadraticZeros.py
>>>from QuadraticZeros import *
Find the x-intercepts
y=x^2 + 14 x + 40
x1 = |
```



```
Python Shell 8/8
>>>#Running QuadraticZeros.py
>>>from QuadraticZeros import *
Find the x-intercepts
y=x^2 + 14 x + 40
x1 = -4
x2 = -10
Correct!
>>>|
```

18. Your program will now let you practice finding the x-intercepts for a quadratic equation. You could stop coding here. However, each time you want to generate a question you have to press

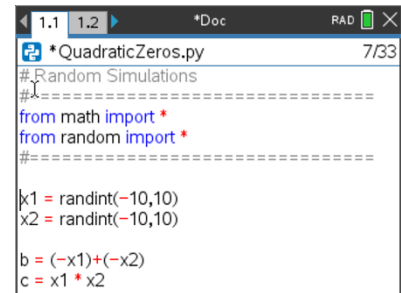
ctrl → r

You can add a loop to your program to repeat the code.

You need to repeat everything except for the two library imports at the top of the code.

Move your cursor to the line that says:

`x1 = randint(-10,10)`



```
*QuadraticZeros.py 7/33
# Random Simulations
#
from math import *
from random import *
#
x1 = randint(-10,10)
x2 = randint(-10,10)
b = (-x1)+(-x2)
c = x1 * x2
```

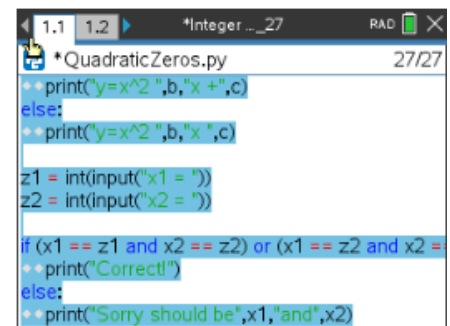
19. Hold down the shift key, then arrow down to the last line of code. This will select all the lines.

Press:

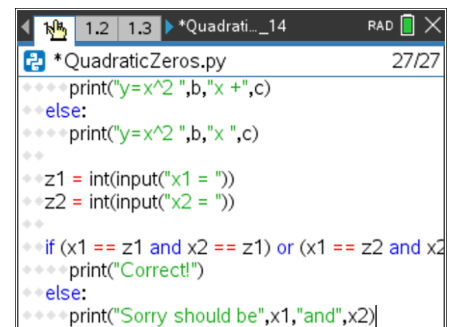
Menu → Edit → Indent

This adds one level of indentation to all of your code.

Notice additional two   diamonds, appear on each line.



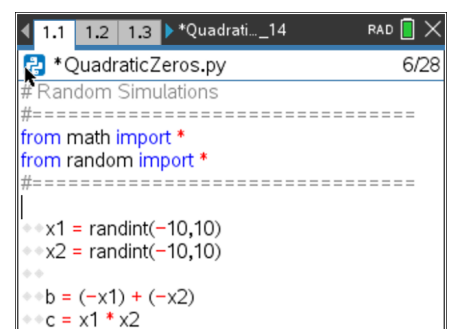
```
*QuadraticZeros.py 27/27
print("y=x^2 ",b,"x +",c)
else:
print("y=x^2 ",b,"x ",c)
z1 = int(input("x1 = "))
z2 = int(input("x2 = "))
if (x1 == z1 and x2 == z2) or (x1 == z2 and x2 == z1):
print("Correct")
else:
print("Sorry should be",x1,"and",x2)
```



```
*QuadraticZeros.py 27/27
print("y=x^2 ",b,"x +",c)
else:
print("y=x^2 ",b,"x ",c)
z1 = int(input("x1 = "))
z2 = int(input("x2 = "))
if (x1 == z1 and x2 == z2) or (x1 == z2 and x2 == z1):
print("Correct")
else:
print("Sorry should be",x1,"and",x2)
```

20. Move your cursor back to the top of your code.

If need be, add a line after the comments before the `x1 = randint(-10,10)` line.



```
*QuadraticZeros.py 6/28
# Random Simulations
#
from math import *
from random import *
#
x1 = randint(-10,10)
x2 = randint(-10,10)
b = (-x1)+(-x2)
c = x1 * x2
```

21. You will add a for loop to repeat the code. The for loop will allow you to determine the number of times the code should be repeated.

To repeat the you will add a four statement.

for can be found using

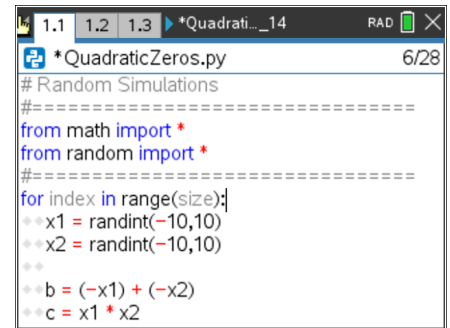
Menu → Built-Ins → Control → for index in range(size):

The index is a variable that will count from zero up to but not including the value for the size.

Set the index to the variable c.

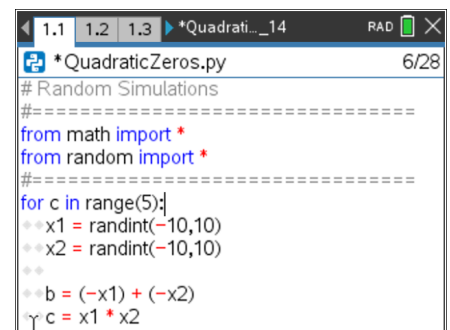
Set the size to 5.

Block is a place holder for the code that goes in the for statement. You've already coded this information. Delete the line with block.



```

1.1 1.2 1.3 ▶ *Quadrati..._14 RAD 6/28
*QuadraticZeros.py
# Random Simulations
#=====
from math import *
from random import *
#=====
for index in range(size):
  x1 = randint(-10,10)
  x2 = randint(-10,10)
  b = (-x1) + (-x2)
  c = x1 * x2
  
```



```

1.1 1.2 1.3 ▶ *Quadrati..._14 RAD 6/28
*QuadraticZeros.py
# Random Simulations
#=====
from math import *
from random import *
#=====
for c in range(5):
  x1 = randint(-10,10)
  x2 = randint(-10,10)
  b = (-x1) + (-x2)
  c = x1 * x2
  
```

22. You have completed the first coding project for factoring quadratic equations. Use your applet to practice factoring. When you have mastered factoring these types of problems, move on to coding activity “Rational Quadratic Zeros” to add another level of factoring to your game.