|  |  |  |
| --- | --- | --- |
| **Rational Quadratic Zeros** |  | |
| In this lesson, you will extend the code from **Integer Quadratic Zeros**. If you didn’t complete the activity, complete that activity first or obtain the base code from your teacher.  In this lesson, you will create a game that lets you practice finding x-intercepts for equations in the form y = ax2 + bx + c. These solutions will have one rational and one integer solution.  In the challenge, you will apply what you have learned to create a third game. This game will let you practice finding x-intercepts for equations in the form y = ax2 + bx + c where both x-intercepts could be rational numbers. | **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 rational solutions | |
| **Math Course Connections: Algebra 1 or Algebra 2** | | |
| In this lesson, you will create a game that lets you practice finding x-intercepts for equations in the form y = ax2 + bx + c. These solutions will have one rational and one integer solution.  Example 1:  Example 2: | | |
| 1. Open up the python File Manager.  [prgm] Python App  Scroll through your list of projects until the arrow is to the left of QUADZERO.   Select [Manage]    Choose: 1 Replicate Program  Name the new program QUADZER2. | |  |
| 1. The factored equations in this problem will be of the type:  y = ( m\*x – x1 )( x – x2 )  In the first project, the line   x2 = randint(-10,10)  creates and stores random integer value from -10 to 10 in the variable x2   Similarily, we will let m be an integer value from two to seven.   Add a line of code after the x2 = randint(-10,10) to generate and store the   value of m. | |  |
| 1. How does the addition of the cofficient m change the values of b and c in the code?  Use distribution to solve and rewrite the equation in standard form.   y = ( m\*x – x1 )( x – x2 )  b = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  c = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Modify the values for b and c in the code if necessary. | |  |
| 1. Does your code match the code to the right? | |  |
| 1. When distributing m in step 3, your final equation started with mx^2 instead of x^2.  How can you modify the print statements to show mx^2 instead of x^2?  Be careful. You want the value of m to display not the letter m.   Original Modified | | |
| 1. *How does the user input change?*   Let’s look at a sample problem:  4x2 + 25x - 21 = 0  (4x – 3)(x + 7) = 0  4x – 3 = 0 x + 7 = 0  x = 3/4 x = -7  Not all of the answers will be fractions, but some will be fractions.   The original code:  z1 = int(input(“x1 = “))  will not allow the user to enter the division sign.  To preform a calculation then store as a float, use the eval() function.  Modify the two input lines to:  z1 = float(eval(input(“x1 = “))  z2 = float(eval(input(“x2 = “))  Fns > Type > Float Fns > I/O > eval() | |  |
| 1. You have one more modification to make. The original project had the line:  if (x1 == z1 and x2== z2) or (x1 == z2 and z1== x2):   Modify the if statement so it includes the new coefficient m.   *Execute your program. Verify your if statement works.* | |  |
| 1. Did you change the code to:  if (x1/m==z1 and x2==z2) or (x1/m==z2 and x2==z1): | |  |
| 1. Lastly, modify your print statement if the user input is incorrect.   Original:  print(“Sorry sould be”,x1,”and”,x2)   Change To:  print(“Sorry sould be”,x1,”/”,m,”and”,x2) | |  |
| **Challenge:** Create a **QUADZER3** program that generates equations with two fractional x-intercepts.  For example, 6x2 – 11x – 35 = 0 factors to (3x + 5)(2x – 7) = 0.  The x-intercepts would be x = -5/3 and x = 7/2. | | |