# **Pythagorean Approximations Assignment**

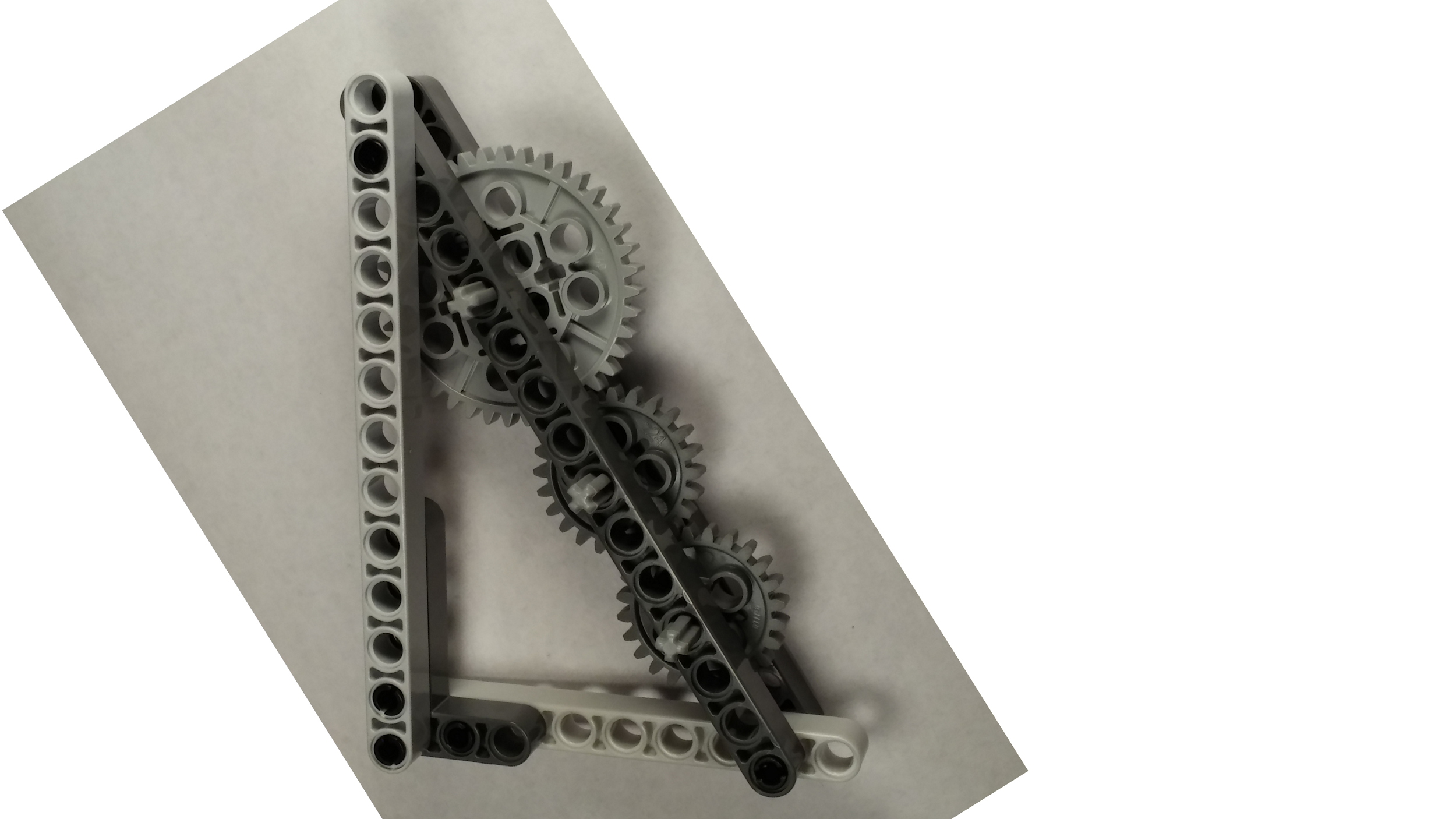
# **(Java programming and spreadsheet analysis for LEGO robot building)**

## **Background**

In LEGO kits for building robots, there is generally a substantial supply of "beams" or "liftarms" in various lengths as well as pins to hold them together.

|  |  |
| --- | --- |
| beam image | image of pins |

While LEGO beams come in varying lengths, they all have holes spaced at a regular interval that we will refer to as one LEGO unit (about 8mm). In most LEGO constructions, beams are generally placed horizontally or vertically along an underlying virtual grid where the holes fall. But it may also be desirable to place beams along a **diagonal** where most of the beams are horizontal and vertical but a set of gears is sandwiched between beams running diagonally.



In this example, the beams almost form a right triangle with sides of length 7, 11, and 13 in LEGO units, but these lengths do not exactly satisfy the Pythagorean Theorem.

Recall that the *Pythagorean Theorem* provides a relationship between the side lengths in a right triangle, a triangle that contains a right (90⚬) angle. It tells us that

a2 + b2 = c2

where c is the length of the hypotenuse (the longest side, which is opposite the right angle) and a and b are the lengths of other sides (legs). (A nice site with graphical illustrations is <https://www.mathsisfun.com/pythagoras.html>.)

In the previously referenced LEGO example, we see that we do not have a right triangle because the sides violate the Pythagorean Theorem:

|  |  |
| --- | --- |
|  | 72 + 112 = 49 + 121 = 170 ≠ 169 = 132 |

The reality is that because the numbers are close enough to working out (170 is approximately 169), it is possible to slightly deform the LEGO pieces so that they fit together into what is essentially a right triangle. The goal of the exercise below is to find triples of side lengths, such as 7-11-13, that are close enough to satisfying the Pythagorean Theorem to be constructible in practice with LEGO.

## **Java programming**

Your first task is to create a Java program that performs computations on potential Pythagorean triples with side lengths up to 14 (the longest length between the end holes in standard LEGO beams). Use a loop to consider all possible lengths from 1 to 14 for the shortest leg, and incorporate another loop inside to consider possible lengths for the long leg. For the long leg, consider lengths from the current length of the shortest leg up to 14.

For each combination of leg lengths as described above, compute:

* the **actual hypotenuse**, the square root of shortleg\*shortleg+longleg\*longleg
* the **approximate hypotenuse** by rounding the hypotenuse to the nearest whole number
* the **error** (the approximate hypotenuse minus the actual hypotenuse)
* the **absolute error** (the absolute value of the error)
* the **slope** of the hypotenuse (long leg divided by short leg)

You may find the following methods of the Math class useful for these operations.

|  |  |
| --- | --- |
| static double | [**sqrt**](https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html#sqrt-double-)(double a)  Returns the positive square root of a double value. |
| static double | [**abs**](https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html#abs-double-)(double a)  Returns the absolute value of a double value. |

For rounding a positive decimal number to the nearest whole number, it will be useful to use the following command:

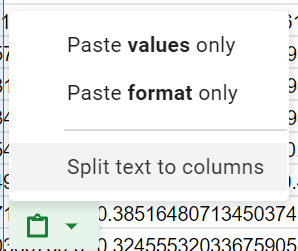
int rounded = (int)(value + 0.5); //rounds value to the nearest positive whole number

You are to print all your data to the console. The first thing your program should print is just the line "Short Leg,Long Leg,Hypotenuse,Approx Hyp,Error,Abs Error,Slope". As you consider each combination of leg lengths, print more lines, with the leg lengths and computed values in the correct order and separated by commas. Before printing data, check that the hypotenuse and slope are in the range of interest to us: approximate hypotenuse at most 14 (so it can be realized with a single LEGO beam) and slope at most 5 (so that it is substantially more interesting than something that runs horizontally or vertically).

Copy all the output from your program. Click on <http://sheet.new> to create a new google sheet. Make sure the cell in the upper left hand corner is **selected** but not in **edit mode**.



Paste the output data from your program (for example, using the "Edit" menu), and then you should see a little clipboard at the lower corner. Click on the clipboard and select the option **Split text to columns**.

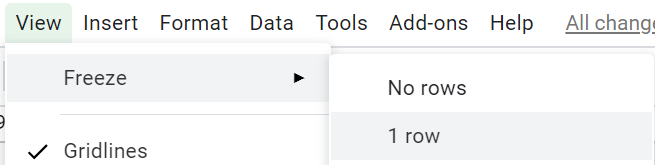


After your data is split into columns, your sheet should look like the screen below (without all the data blurred out!)

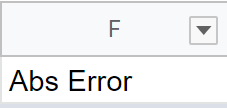


## **Spreadsheet analysis**

Your next task is to manipulate the spreadsheet in such a way that you can quickly read off the triple of three integral side lengths that produces the least absolute error for a specific slope. To make your data easier to sort, go to the **View** menu, select the **Freeze** submenu, and choose the **1 row** option.



To put your spreadsheet in the desired form, sort all the rows in ascending order of the absolute error values, and also sort in ascending order of the slope values. Think about which of these two columns you want to sort on first so that all the rows with the same slope will end up together in the spreadsheet. You can sort according to the numbers in a column by clicking on the arrow next to the single-letter header inserted by the spreadsheet program at the top of the column, and then choosing **Sort sheet A→**Z or **Sort sheet Z→A**.



## **Data Questions**

1) When the slope is 1.25, which short leg and long leg pair will have the smallest absolute error?

2) Which slope is the most common in your data?

## **Deliverables**

Turn in your Java program, your sorted spreadsheet, and answers to the data questions using the mechanisms specified by your teacher.