

# Computer Science



SIMON FRASER  
UNIVERSITY

Student:

Tommy Ju

Teacher:

Ms. Lawrence



# Table of Contents

Cover Page.....	1
Formal Proposal.....	3
Capstone Timeline.....	4
Logs for Vertex and Quadratic Calculators.....	5
Logs for Triangle Solver Program.....	6, 7, 8
Goals.....	9
Resume & Cover letter.....	10
Reference Letters.....	11
Final Reflection.....	12
Core Competencies.....	13

# Formal Proposal

◇ [Formal Proposal Word Document](#)

# Capstone Timeline

Due Date: December 13 2019

Oct. 17 – Research and develop flow charts for Vertex form converter and Quadratic Formula Calculator.

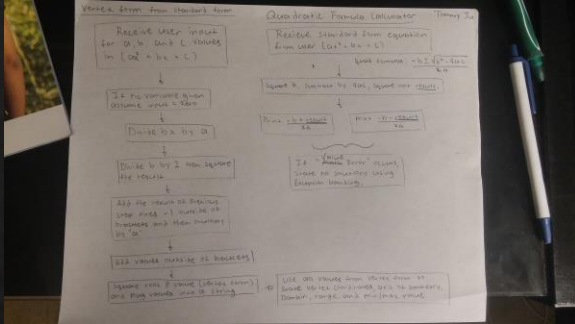
Oct. 30 – Code and debug program using flow charts as references.

Nov. 2 – Research modules that can be used to solve triangles and create a flowchart for the Triangle Solver program.

Dec. 11 – Finish Coding and debugging the Triangle Solver, create a GUI if done planning before Nov. 20



# Logs for Vertex and Quadratic Calculators



Created flow charts for Vertex form converter and Quadratic Calculator. Also reviewed procedures for completing the square.

## Motivation

The classic division operator makes it hard to write numerical expressions that are supposed to give correct results from arbitrary numerical inputs. For all other operators, one can write down a formula such as  $x*y**2 + x$ , and the calculated result will be close to the mathematical result (within the limits of numerical accuracy, of course) for any numerical input type (int, long, float, or complex). But division poses a problem: if the expressions for both arguments happen to have an integral type, it implements floor division rather than true division.

The problem is unique to dynamically typed languages: in a statically typed language like C, the inputs, typically function arguments, would be declared as double or float, and when a call passes an integer argument, it is converted to double or float at the time of the call. Python doesn't have argument type declarations, so integer arguments can easily find their way into an expression.

Had trouble with Python's "Classic division", researched and learned how to use "division" function from `__future__` module to get accurate numbers.

Finished debugging and making sure the Vertex Form Converter works properly. Added extra features to find slope given coordinates, x – intercepts, and y – intercept.

10 Oct.

16 Oct.

20 Oct.

25 Oct.

27 Oct.

Learned about the square root function under the Math module and learned how to use Exception Handling for the Quadratic Calculator.

Finished debugging and coding the Quadratic Formula Calculator

Return  $x$  raised to the power  $y$ . Exceptional cases are possible. In particular, `pow(1.0, x)` and `pow(x, 0.0)` return a NaN. If both  $x$  and  $y$  are finite,  $x$  is negative, and  $y$  is not an integer, `pow` raises `ValueError`.

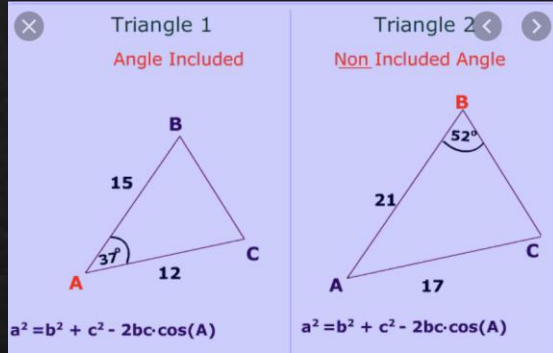
Unlike the built-in `**` operator, `math.pow()` always returns a float, even if the arguments are integers. This is useful for computing exact results like the square root of 2.

`math.sqrt(x)`  
Return the square root of  $x$ .

try statement can have multiple `except` blocks to handle different exceptions. Multiple exceptions can also be put into a single `except` block using parentheses, to have the `except` block handle all of them.

```
try:
    variable = 10
    print(variable + "hello")
    print(variable / 2)
except ZeroDivisionError:
    print("Divided by zero")
except (ValueError, TypeError):
    print("Error occurred")
```

# Logs for Triangle Solver Program (2/3)



Added more detail to program's flowchart and learned about non-included and included angles when solving SSA triangles.

Code didn't work properly, figured out a way to turn a function's input into a list rather than a tuple.

Ran into a very tedious bug, Python's `sin()` and `cos()` functions only work in radians and my code was using degrees.

23 Nov.

1 Dec.

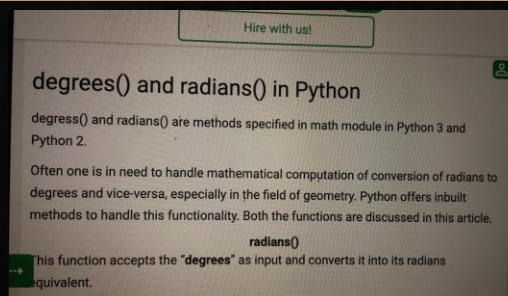
2 Dec.

3 Dec.

5 Dec.

Learned about `degrees()` and `radians()` functions which can be used to convert degrees to radians and vice versa.

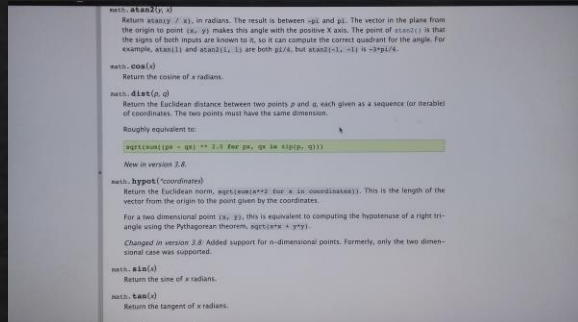
Learned how to use the `round()` function in Python.



## Python `round()` Function

The **`round()`** function returns a floating point number that is a rounded version of the specified number, with the specified number of decimals. The default number of decimals is 0, meaning that the function will return the nearest integer.

# Logs for Triangle Solver Program (1/3)

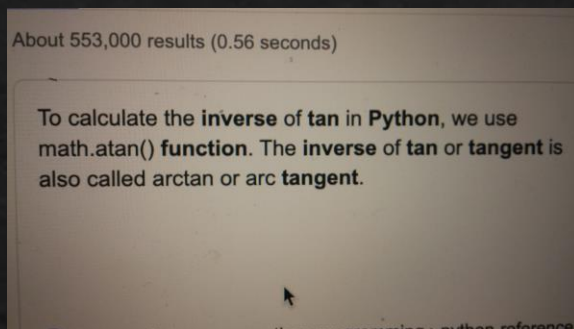
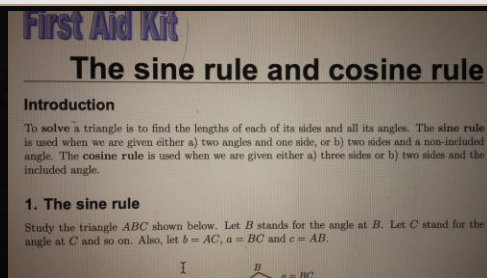


Researched and found trigonometric functions under the Math module that can return an answer in radians on the Python official website.

6 Oct.

2 Nov.

Reviewed how to use cosine and sine laws to solve triangles (SSA, SAA, SSS), and developed a rough flow chart for program.

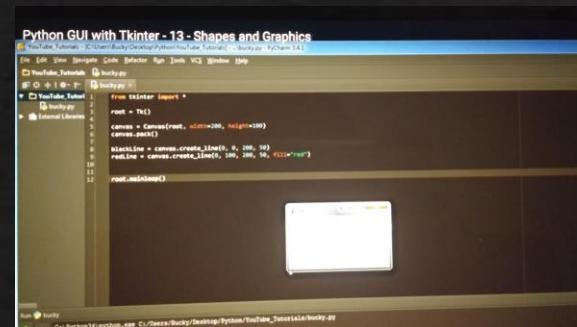


Learned about arc tangent, arc sine, and arc cosine functions that can be used to find the angle of a trigonometric ratio.

5 Nov.

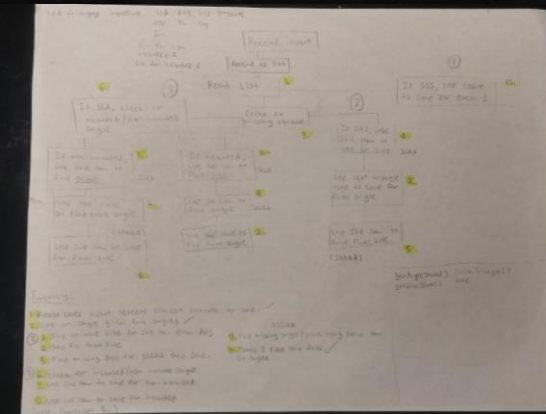
11 Nov.

Expanded on the program design by mapping out functions that the program can use to solve various triangles.



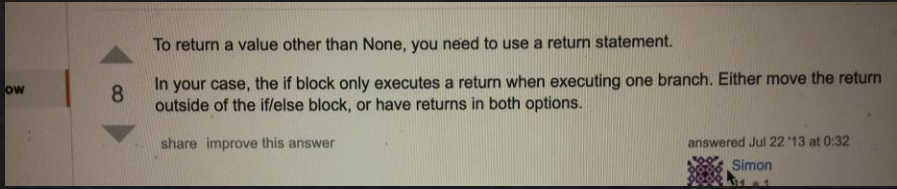
Taught myself how to make a GUI using the Tkinter module from videos on Youtube.

18-23 Nov.





# Logs for Triangle Solver Program (3/3)



Spent over 15 hours coding and debugging over the weekend.

When packing a widget in Tkinter, you have to pack it separately on a new line or else creates a "NoneType".

Finished creating a program that accounts for every possibility to solve any triangle with 3 data points and at least one side.

6-7 Dec.

11 Dec.

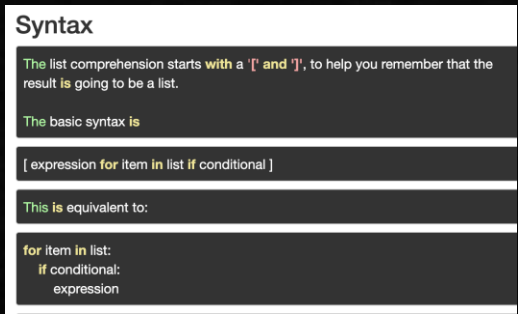
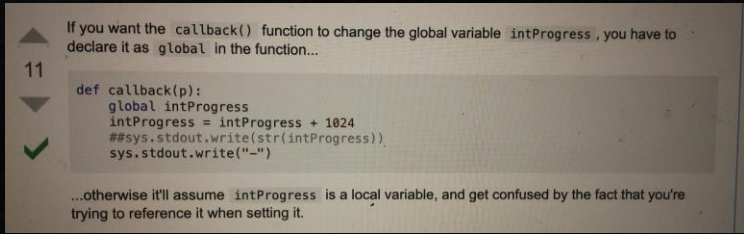
11 Dec.

11 Dec.

12 Dec.

Learned how to reference a global variable inside a function and solved a bug by learning that you always have to return a nested function.

Learned how to use "list comprehension" to iterate through a list and overwrite its elements.





# Goals

◇ [Goals Word Document](#)

# Resume and Cover letter

◇ [Dropbox link](#)

# Reference Letters

◇ [Dropbox link](#)



# Final Reflection

◇ [Capstone Final Reflection](#)

# Core Competencies

- ◆ [Personal Stories Assignment](#)
- ◆ [Self Assessment of Core Competencies](#)