Academic Dishonesty Disclaimer

All of the work you submit must be done by you, and your work must not be submitted by someone else. Plagiarism is academic fraud and is taken very seriously. The department uses software that compares programs for evidence of similar code. Please read the Rules and Regulations from the U of T Governing Council (especially the Code of Behaviour on Academic Matters).

Please don't copy. We want you to succeed and are here to help. Here are a couple of general guidelines to help you avoid plagiarism:

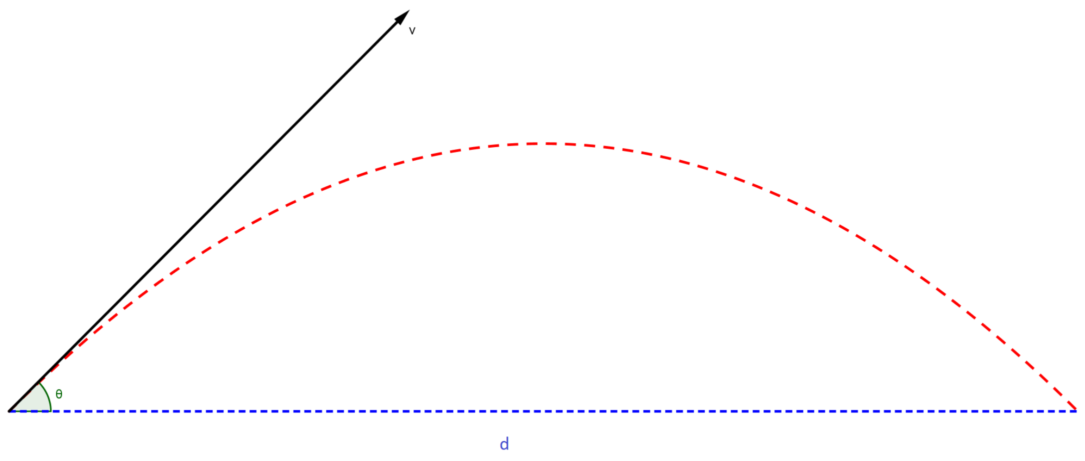
Never look at another assignment solution, whether it is on paper or on the computer screen. Never show another student your assignment solution. This applies to all drafts of a solution and to incomplete solutions. If you find code on the web that solves part or all of an assignment, do not use or submit any part of it! A large percentage of the academic offenses in CS involve students who have never met, and who just happened to find the same solution online. If you find a solution, someone else will too. The easiest way to avoid plagiarism is to only discuss a piece of work on the course discussion board, with the CSC108H TAs in the open labs, or the CSC108H instructors in office hours.

Assignment 1

Please read this document very carefully. Follow instructions exactly. If you have any questions please post them to the course Piazza page.

This assignment is due October 15th, by 10:00pm

Consider a game where you control a cannon which is trying to hit a target some distance d away. You're allowed to choose the velocity of the projectile, and the angle at which the cannon is shot. If you miss your target (outside of some tolerance), the game informs you if you shot short of or past the target, and gives you a chance to try again. See the Figure below for a visual aid.



You are given a file, `assignment1.py`, with six incomplete functions as well as the framework needed to play the game. For this assignment, **you're required to complete these six functions**. A description regarding the intended behaviour of these functions is given later in this documents. Further documentation and examples for these functions are given in the docstrings within `assignment1.py`.

We will assume perfect physics (no wind, constant gravity, etc.) then

$$d = \frac{v^2 \sin(2\theta)}{g}. \tag{1}$$

We will also assume that π equals python's built in value for π, `math.pi`. Note the value is declared at the top of `assignment1.py`. For the purposes of this assignment we will use the following definitions.

1. A valid velocity is a string denoting a positive number, potentially with a decimal point. For example, the following are valid velocities: "15", "10.12", "60.", while the following are **not** valid velocities: "10m/s", "-17", "10.1.4".

2. A valid angle is a string denoting a number, potentially with a decimal point where the last character of the string is either a "d" or "D" (denoting the given angle is in degrees), or a "r" or "R" (denoting the given angle is in radians). Moreover, if in degrees the numeric portion of the string must be less than 90 and greater than 0, and if in radians the numeric portion of the string must be less than $\pi/2$ and greater than 0. For example, "23d", "1.5R, and "77.D" are valid angles, while '-23d", "3.14r, and "7.7R" are **not** valid angles.

Function Name	**Description**
`get_distance(float, float) -> float`	The first parameter is the velocity, the second parameter is the angle of fire. Returns the horizontal distance traveled by the projectile as given by equation (1).
`degrees_to_radians(float) -> float`	Given some number of degrees, it returns the corresponding converted amount of radians.
`get_radian_of(str) -> float`	Given a *valid angle*, it takes the numerical value of the string and returns that angle in radians.
`is_a_number(str) -> bool`	Given a string, returns True if and only if that string represents a positive number (potentially with a decimal).
`is_valid_angle(str) -> bool`	Given a string, returns True if that string is of the form described earlier in this document.
`approx_equal(float, float, float) -> bool`	Returns True if and only if the first two parameters are within some value of each other, where said value equals the third parameter.

Submitting and Grading

This assignment will be submitted electronically via MarkUs. Please find the MarkUs link on the course website. Note, to avoid potential confusion and submitting to the wrong location, there will be nowhere on MarkUs to submit Assignment 1 until Assignment 0 is past due.

This assignment is worth 4% of your final grade. Grading is done completely automatically. That is, a program calls your function, passes it certain arguments, and checks to see if it returns the expected output. Each function is worth 20% of assignment grade, with the exception of `get_distance` and `degrees_to_radians` which are each worth 10% of the assignments grade. For any one function, if you pass n of the m tests we run on that function, your grade for that function will be n/m.

Shortly after the deadline, you will receive your grade. If you are not content with this grade, you may resubmit your assignment up to 48 hours after the original deadline with a 20% penalty. If you choose to resubmit, your final grade on the assignment will be the higher of the two grades (the original submission, and the re-submission with a 20% penalty). Good luck!

Final Notes

After Week 3's material (if statements), you will have all the knowledge needed to complete this assignment. I'd suggest looking up some of Python's String methods as there are a few which you may find useful to complete the functions. This assignment is 100% doable without loops. I strongly suggest you implement these functions without using loops.