# CSCA20 - Lab 3

**Loops & Lists**

## Learning Objectives

This lab focuses on looping and lists, and in particular getting comfortable with the 3 types of loops (elemental for, counted for, and while loops)

## Prelab

We've provided you with some starter code that lets you enter runners and times. Make sure you can run this code, and understand how it's adding to the two relevant lists.

## Demonstration & Evaluation

This lab will allow you to demonstrate the skills of Variables and Loops. We've taken care of the user Input/Output for you, but don't worry, you'll get chances to show competency in user I/O again in later labs if you haven't demonstrated it thus far.

This is also the first lab where it is likely that some students may struggle to complete the lab before the demonstration. If you can't complete it: don't panic. You will have the opportunity to demonstrate loops in future labs. So if you can't finish during your tutorial, work on it at home, ask for help on Piazza if you need it, and come back next week ready to show off more skills.

## The Scenario

The Ultramarathon of Tokyo in Season of Cold (UTSC) is the premier winter-time long distance running race in Japan. They have asked you to build them a tool to track and calculate times for runners in the qualifying race. They aren't sure how to decide who qualifies in a way that has the right number of people, but still keeps a good average time.

## Data Input

We have already provided you with code to get the names and finishing times of the runners. You may assume that the data is entered as the runners finish (i.e., the times will always be increasing). Make sure you understand what that code is doing and how it works.

## Total Average

The first thing UTSC wants you to do is to calculate the average time for all the runners in the qualifying race. So you need to add up all the times, and divide by the number of runners.

## Option 1: Number of Qualifiers

UTSC would like to know what the results would look like if they set a specific number of runners who would qualify for the race. So your code should prompt for that number, and print out the names of everyone who would qualify, as well as the average time for qualifiers.

## Option 2: Qualifying Time

Another way of deciding who qualifies would be to set a cutoff time, and only runners whose time is below this mark would qualify. So your code should prompt for that time, and print out the names of everyone who would qualify, as well as the average time for qualifiers

## Hints

Here are a few hints that might help you with this assignment:

- We're testing you on the 3 types of loop. So you're going to have to write one of each loop. Take a moment before you start to think about which task will require an elemental for, a counted for, or a while loop.
- Remember that you can assume that all runner data will be entered in order, so the first runner will have the lowest time, and the times will steadily increase from there
- You are not responsible for bad input. So if there are 5 runners, and the user sets the number of runners to be 10, your code is allowed to crash. You may also assume that the cutoff time is such that at least some runners will qualify and some won't. (we'll worry about building more error-proof code later)
- There are many ways you can go about this, some of which don't actually require loops at all (e.g., there is a way to get the total of a list using a built in function of python), but the goal here isn't just to get working code, it's to practice and learn how to use loops.

## Extra Practice

If you finish early and want to practice more, here are a few things you can try:

- Find different qualification cutoff methods. Can you write code that lets the user input the name of the last qualifying runner? What about code that lets runners qualify until the average time goes above a certain point? What about code that lets the user specify multiple possible cutoffs and stops when it hits the first one? (e.g., I want to qualify runners until the time hits X or the number of runners hits Y)
- You can try to make the code more error-proof, so that if the user enters bad input, the code doesn't crash. If they enter a time greater than any runner took, or less than the fastest runner, it would be nice if the code did something sensible instead of just crashing
- Try to add another field. Maybe name, time, and age, and then lets the user do advanced stuff like setting a maximum number in a specific age category, or printing out the average time for runners over a certain age (this one will likely require you to read ahead to learn about IF statements)