# ISTA 311 Programming assignment 1

Due: Thursday, September 26 (11:59 PM)

## Problem Statement

A probability distribution can be represented in Python as a dictionary, where the keys are the possible outcomes and the values are their probabilities. For example, the dictionary

```
{"heads": 0.5, "tails": 0.5}
```

can be used to represent the probability distribution of a fair coin flip.

For our purposes, we will define a class called `Distribution` that acts as a wrapper for a dictionary, defining several additional methods. A template for the class is on D2L, with the `__init__` method already defined. You do not need to modify the `__init__` method, but you should read it and understand it.

To complete the assignment, implement the following methods:

### Calculating probability (10 points)

Our first task is to calculate the probability of a set of outcomes.

Write a method called `prob`. Your method should take one parameter in addition to `self`: a Python `set` object. It should return the real number between 0 and 1 which is the probability of the event corresponding to the set.

(You may assume that the set does not contain any elements not in the dictionary; that is, you don't need to write code protecting against `KeyError`s.)

### Normalizing a distribution (15 points)

In applying Bayes' theorem later in the course, we will encounter intermediate steps where a distribution does not follow Kolmogorov's laws because the sum of the probabilities is not 1. It will be useful to correct this.

Write a method called `normalize` which takes only the parameter `self`. This method should modify the dictionary in-place by scaling the probabilities so that they add up to 1. You may return `None`.

### Conditioning a distribution (15 points)

We have seen that the process of calculating conditional probabilities $P(A|B)$ is equivalent to eliminating those outcomes inconsistent with the given event $B$, and recalculating probabilities so that they sum to 1 while preserving the relative sizes of those probabilities.

Write a method called `condition` which takes one parameter in addition to `self`: a Python `set` object which is a subset of the keys of the dictionary (i.e., an event). Your method should modify the dictionary in-place by removing key-value pairs corresponding to outcomes not in the parameter set, and recalculating the probabilities so that they sum to 1. The end result is the *conditional distribution*, conditioned on the event defined by the parameter set. You may return `None`.

**Simulating a random process (20 points)**

Our second task is to simulate the experience whose probability distribution is described by a dictionary.

Write a method called `sample`. Your method should take no parameters besides `self`. It should return a randomly chosen outcome (dictionary key) according to the probability distribution described by the dictionary. You may use any function that generates uniformly distributed random numbers from $[0, 1)$ (such as `random.random(), or numpy.random.uniform()`), but no other source of randomness.

You may assume that the dictionary passed as input will follow Kolmogorov's laws: that is, the keys represent mutually exclusive outcomes and the values sum to 1.

## Test cases

A Python module containing several test cases `simulate` will be available on D2L. Note: this test script requires SciPy and matplotlib to run.