# Analyzing a Simple Dice Game

Help Center

## Overview

This week's mini-project will be to build a program that determines which dice to discard when trying to maximize your score on a Yahtzee hand. To help illustrate several of the concepts that you will use in the mini-project, we will consider a simpler dice game as follows:

Walking on a busy street, you happen upon a stranger sitting at a small folding table and holding three dice. He is pitching the following game to the surrounding crowd: give him $10 and he will let you roll the dice. If you roll doubles (two of the dice are paired), he will return your $10. If you roll triples (all three dice have the same value), he will give you $200. However, if you roll neither (the dice are unpaired), he keeps your $10. Although the lure of $200 is tempting, you are wise and move on. Later that evening your curiosity gets the best of you and you decide to use your math/Python skills to analyze the game and determine whether the game (if it is legitimate) is in your favor.

Our task in analyzing this game is to compute the expected value of a single round of the game. To compute this value, we need to determine how likely rolling doubles and triples are using three dice. After paying our initial $10, our expected return will be the probability of rolling doubles times $10 plus the probability of rolling triples times $200. If the resulting expected return is greater than $10, you should have played (as long as you had a bodyguard). Your task for this activity is to build a short Python program that computes this quantity.

## Generating rolls

Our first decision is how to create the set of all rolls of three dice. We already have some code that is available to us here. This program provides us with two possible functions for generating dice rolls. Which function should we use? `gen_all_sequences` or `gen_sorted_sequences`? Your first task is to decide on which function you should use and why. Stop and think about this question for a moment.

We recommend that you use `gen_all_sequences` since the probability for each sequence is easy to compute: $\frac{1}{6} \times \frac{1}{6} \times \frac{1}{6} = \frac{1}{216}$. Using `gen_sorted_sequences` would require some math to account for all of the sequences that correspond to one sorted sequence. With that decision behind us, here is a short template that you can build on.

## Detecting doubles and triples

Using `gen_all_sequences`, we can now generate all possible $6^3 = 216$ sequences of three die values. To compute the expected value for one round of the game, we need to detect whether a sequence corresponds to doubles or triples. Since this task can be easily isolated from the computation of the expected value, we will next implement a helper function `max_repeats(seq)` that takes a sequence and computes the maximum number of times that any individual item occurs in the sequence.

Before you rush off to implement this helper function, I suggest that you spend a few seconds and consider how to implement the function. In particular, you might take a look at the list methods in the Docs and see if there are any built-in Python methods that can help simplify this task. As a hint, our implementation of `max_repeats` consists of two lines of Python code (using a list comprehension). If you are writing more, think more and code less.

## Computing the expected value

With the helper function `max_repeats` built, our last task is to implement a function `compute_expected_value`. This function should return the expected value excluding the initial $10 that you paid. If the computed expected value from your roll is greater than $10, you should play the game as much as possible. To aid in debugging your code, the value returned by `compute_expected_value()` should lie in the range of $9 to $11. This result makes the decision of whether or not to play the game a close one. Please implement `compute_expected_value` now.

Once you have implemented your version of `compute_expected_value`, you are welcome to consult our solution. `compute_expected_value()` returns approximately $9.72 which is consistent with the game being slightly unfavorable. So, resist the urge to play and learn how to count cards in Blackjack instead.

Created Wed 4 Jun 2014 3:18 PM PDT

Last Modified Thu 3 Jul 2014 3:51 PM PDT