In [4]:

```
# setup
from IPython.core.display import display,HTML
display(HTML('<style>.prompt{width: 0px; min-width: 0px; visibility: collapse}</style>'))
display(HTML(open('rise.css').read()))

# imports
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
sns.set(style="whitegrid", font_scale=1.5, rc={'figure.figsize':(12, 6)})
```

CMPS 2200

Introduction to Algorithms

Randomized Algorithms

Today's agenda:

- Random variables and expectations
- Making random choices in algorithms

Suppose you could flip a coin to make a decision in an algorithm, rather than relying on inputs. So:

```
 if (random([0,1]) > 0):
    print('heads')
else:
    print('tails')
```

Why would we ever do this?

When we don't know exactly how to proceed in solving a problem, we can make a random choice and hope we made the right one.

Alternately, we can view randomization as helping us avoid always making the wrong choice.

Additionally for instances where we might not have a good way to proceed, we can "guess" an answer and "hope" it's a good choice.

What does "hope" mean, computationally? A proof that at least most of the time, we are correct/efficient.

Randomized algorithms can often be:

1- easier to understand

2- faster

3- more robust to adversarial input (cryptography)

Basic Definitions

A probability space consists of a **sample space** Ω representing the set of possible *outcomes*, and a **probability measure** $\mathbf{P} : \Omega \to \mathbb{R}$. The probability measure $\mathbf{P}$ must satisfy: </p>

- **Nonnegativity**: $\mathbf{P}\left[A\right] \in [0, 1]$
- **Additivity**: For any two disjoint events A and B (i.e., $A \cap B = \emptyset$): $\mathbf{P}\left[A \cup B\right] = \mathbf{P}\left[A\right] + \mathbf{P}\left[B\right]$
- **Normalization**: $\mathbf{P}\left[\Omega\right] = 1$

For example, let Ω be the set of all outcomes of a pair of 6-sided dice.

$\Omega = \{(1,1), (1,2), \ldots, (2,1), \ldots, (6,6)\}$

How many outcomes are there if the dice sum to 3? To 4?

The dice sum to 3 if we roll any of $\{(1,2), (2,1)\}$, and sum to 4 if we roll any of $\{(1,3), (2,2), (3,1)\}$.

Every outcome is disjoint and has probability 1/36. So the probability of rolling a 3 is 2/36 = 1/18 and the probability of rolling a 4 is 3/36 = 1/12.

If we want to consider multiple, possibly overlapping events. Then, the **union bound** is helpful:

$$\mathbf{P}\left[\bigcup_{0 \leq i < n} A_i\right] \leq \sum_{i=0}^{n-1} \mathbf{P}\left[A_i\right]$$

Conditional Probabilities

We may want to know the probability of an event A with "partial knowledge." The **conditional probability** of an event A *given* that B occurs ($\mathbf{P}\left[B\right] > 0$) is

$$\mathbf{P}\left[A \mid B\right] = \frac{\mathbf{P}\left[A \cap B\right]}{\mathbf{P}\left[B\right]}$$

For example, what is the probability that the first die comes up 1 given that the sum of a pair of dice is 4?

$A = \{(1,1), (1,2), (1,3), (1,4), (1,5), (1,6)\}$

$B = \{(1,3), (2,2), (3,1)\}$

$A \cap B = \{(1,3)\}$

$\mathbf{P}\left[{A \mid B}\right] = \frac{\mathbf{P}\left[{A \cap B}\right]}{\mathbf{P}\left[{B}\right]}$

\frac{|A \cap B|}{|B|} = \frac{1}{3}.$

A useful fact that relates conditional probabilities to non-conditional probabilities is the **Law of Total Probability**:

Let Ω be a sample space and let $A_0, \ldots, A_{n-1}$ be a set of events that partition Ω such that $\mathbf{P}[A_i] > 0$ for all $0 \leq i < n$. Then, for any event B we have that:

$$\begin{aligned}\mathbf{P}[B] &= \sum_{i=0}^{n-1} \mathbf{P}[B \cap A_i] \\ &= \textstyle\sum_{i=0}^{n-1} \mathbf{P}[A_i]\,\mathbf{P}[B \mid A_i]\end{aligned}$$

An intuitive way to think about this is that the probability of an event B is just the probability over different possible (disjoint) situations.

There is a great example in the book about this. See Vol 1, Ch 18.

Another useful concept is **independence**. We say that two events A and B are independent when they are disjoint. So if A and B are independent, $\mathbf{P}[A \cap B] = \mathbf{P}[A] \cdot \mathbf{P}[B]$.

Random Variables and Expectations

A **random variable** X is a real-valued function on the outcomes of an experiment, i.e. $X : \Omega \to \mathbb{R}$. We will consider only discrete random variables.

Even more simply, we will most often use **indicator random variables**, which map outcomes to 0 or 1.

$$Y(d_1, d_2) = \begin{cases} 1 \text{ if } d_1 = d_2 \\ 0 \text{ if } d_1 \neq d_2 \end{cases}.$$

Note: The term "random variable" is somewhat odd. Really it's just a function, but we use this terminology since it's output it dependent on a random process and it is useful to look at functions of outcomes (which have probabilities under a given probability measure).

The **expectation** of a random variable X defined for a probability space $(\Omega, \mathbf{P})$ is denoted:

$$\mathbf{E}_{\Omega,\mathbf{P}}[X] = \sum_{y \in \Omega} X(y) \cdot \mathbf{P}[\{y\}].$$

We usually write this more conveniently as:

$$\mathbf{E}[X] = \sum_{x} x \cdot \mathbf{P}[X = x].$$

Let X be the value of a pair of dice. X can take on values between 2 and 12, where each has a certain probability based on the possible ways the dice can sum to that value. What is the expected value of the value of a pair of unbiased dice? It is:

$$\begin{aligned}\mathbf{E}[X] &= \sum_{x=2}^{12} x \cdot \mathbf{P}[X = x] \\ &= 2 \cdot \frac{1}{36} + 3 \cdot \frac{2}{36} + 4 \cdot \frac{3}{36} + \cdots + 11 \cdot \frac{2}{36} + 12 \cdot \frac{1}{36}\end{aligned}$$

Intuitively, the expectation is really just a weighted average of the values of the random variable.

Let's look at coin flips where we have equal probability of heads/tails. Let X be the number of flips until we get heads. What is $E[X]$? Observe that:

$$E[X] = \frac{1}{2} \cdot 1 + \frac{1}{2}(1 + E[X])$$

Solving for $E[X]$, we get that $E[X] = 2$. More generally, the expected number of trials to get an outcome of probability p is $1/p$. (We'll see this later.)

In [5]:
```
expectation = 0
for d1 in range(1,7):
    for d2 in range(1,7):
        expectation += (d1+d2)/36
print(expectation)
```

```
7.0
```

We will make repeated use of some basic facts about the expectations of random variables.

First, we can easily treat the expected value of functions of random variables:

$$\mathbf{E}\left[f(X)\right] = \sum_{x} f(x) \cdot \mathbf{P}\left[X = x\right].$$

We also have **linearity of expectations** for random variables:

$$\mathbf{E}\left[X + Y\right] = \mathbf{E}\left[X\right] + E\left[Y\right].$$

This identity will help us simplify the calcuation of the *expected* work and span of an algorithm.

How does all this relate to algorithms?

The sample space we must consider for an algorithm is the set of decisions made by the algorithm. We typically use random variables that map outcomes to a way to measure work/span.

However it is usually difficult to reason directly about the overall outcome (e.g. correctness or work/span), so we try to identify when choices are independent or conditional upon one another.

Ultimately if we choose to make random choices in our algorithm, we must relax our notion of correctness and our definitions of work/span.

For correctness, some executions may produce the wrong output while others are correct. We usually seek to have a very high *probability of correctness*.

For work/span, some executions may take longer than others. We seek to provide asymptotic bounds on the *expected* work or span of a randomized algorithm. Intuitively, this is the weighted average of the running times over all possible sets of random choices.

When using randomization we usually have one of two types of algorithms.

A **Monte Carlo** randomized algorithm has a **deterministic** worst-case runtime but a randomized output that is correct with some probability.

A **Las Vegas** randomized algorithm always produces a correct solution, but has an **expected** runtime.

We will focus on Las Vegas algorithms.

Randomly splitting a list

Suppose we are given a list L of length n. We choose a random index i of the list and return $L[i:]$ as output.

What is the expected size of $L[i:]$?

Let X be a random variable that is the size of the output list. By the definition of expectation we have:

$$\begin{aligned}\mathbf{E}[X] &= \sum_{x=1}^{n} x \cdot \mathbf{P}[X = x] \\ &= \sum_{x=1}^{n} x \cdot \frac{1}{n} \\ &= \frac{1}{n} \cdot \sum_{x=1}^{n} x \\ &= \frac{1}{n} \cdot \frac{n(n+1)}{2} \\ &= \frac{n+1}{2}\end{aligned}$$

This might follow your intuition that, since all indices are equally likely, the chosen index averages out to roughly half of the list size.

A related question is, what is the probability of getting a list with more than half of the elements? That is, what is $\mathbf{P}\left[X \geq n/2\right]$?

$$\begin{aligned}\mathbf{P}\left[X \geq n/2\right] &= \sum_{x=n/2}^{n} \mathbf{P}[X = x] \\ &= \sum_{x=n/2}^{n} \frac{1}{n} \\ &= \frac{1}{n} \cdot \sum_{x=n/2}^{n} 1 \\ &= \frac{n/2}{n} \\ &= \frac{1}{2}\end{aligned}$$

Randomly filtering elements

Suppose we are given a list of length L of length n.

For each element we flip an unbiased coin x_i. Return a list R with $L[i]$ such that x_i is heads.

What is the expected size of R?

Let X_i be an indicator random variable that is 1 if element i is chosen, and 0 otherwise. Let X be the size of R. We can see that $X = \sum_{i=0}^{n-1} X_i$ by definition. So by linearity of expectation we can compute:

$$\mathbf{E}[X] = \sum_{i=0}^{n} \mathbf{E}[X_i].$$

Now notice that by the definition of expectation

$$E[X_i] = 0 \cdot \mathbf{P}[X_i = 0] + 1 \cdot \mathbf{P}[X_i = 1] = \mathbf{P}[X_i = 1].$$

Since we flip a coin for each element independently, $\mathbf{P}[X_i = 1] = 1/2$. Thus, $\mathbf{E}[X_i] = 1/2$ for all i. So we have that

$$\mathbf{E}[X] = \sum_{i=0}^{n} \frac{1}{2} = \frac{n}{2}.$$

So the expected size of R is $n/2$.

These simplistic algorithms illustrate the kind of analysis we will be doing to analyze the expected work and span of algorithms for manipulating lists.

We will look at algorithms for selection and sorting. For both problems we will analyze recursive algorithms that use randomization to select the input for the recursive calls.

This creates a complicated situation in which each successive call depends on the previous one, and thereby requires some clever accounting to derive the expected work and span.