Information theory provides us with a formula for determining the number of bits required in an optimal code even when we don't know the code. Let's first consider uniform probability distributions where the number of possible outcomes is not a power of two.
Suppose we had a conventional die with six faces.
The number of bits required to transmit one throw of a fair six-sided die is:$log6 = 2.58$. Once again,we can't really transmit a single throw in less than 3 bits, but a sequence of such throws can be transmitted using 2.58 bits on average.

The optimal code in this case is complicated, but here's an approach that's fairly simple and yet does better than 3 bits/throw. Instead of treating throws individually, consider them three at a time. The number of possible three-throw sequences is $6^3 = 216$.
Using 8 bits we can encode a number between 0 and 255, so a three-throw sequence can be encoded in 8 bits with a little to spare; this is better than the 9 bits we'd need if we encoded each of the three throws seperately.

In probability terms, each possible value of the six-sided die occurs with equal probability $P = 1/6$. Information theory tells us that the minmum number of bits required to encode a throw is $-logP = 2.58$. If you look back at the eight-sided die example,you'll see that in the optimal code that was described, every message had a length exactly equal to -log P bits.

Now let’s look at how to apply the formula to biased (non-uniform) probability distributions.
Let the variable x range over the values to be encoded,and let P(x) denote the probability of that value occurring.
The expected number of bits required to encode one value is the weighted average of the number of bits required to encode each possible value,where the weight is the probability of that value: