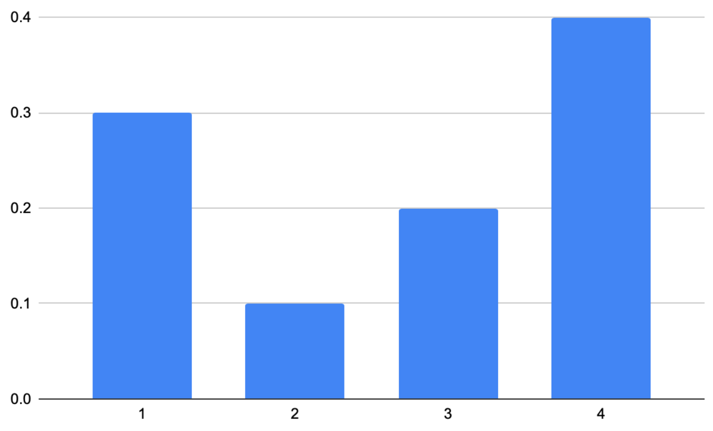
Math and Loops

When presented with an equation, such as the one below for Shannon Entropy, some biology students panic unnecessarily. It's not as complex as it looks and is also easy to implement in a computer program.

$$H = -\sum_{i=1}^{n} p_i log_2(p_i)$$

First, let's read the equation in plain English. Shannon Entropy, H, is the negative sum of p-sub-i times the log base 2 of p-sub-i. That might not have made your life less complex, so let's explore it with an example.

To begin, we need a probability distribution. Let's call the entire distribution P and give it 4 values: P = (0.3, 0.1, 0.2, 0.4). Graphically, this looks like the bar chart below.



If we enumerate each item in P, we note that P_1 = 0.3, P_2 = 0.1 P_3 = 0.2, and P_4 = 0.4. Since this is a probability distribution, all of the values must sum up to 1.0, which they do. We could also call these values a frequency distribution or a histogram. Now let's calculate Shannon Entropy by summing up the product of P * log_2(P).

$$P_1 log_2(P_1) = 0.3\, log_2(0.3) = -0.5210896782498619$$
$$P_2 log_2(P_2) = 0.1\, log_2(0.1) = -0.33219280948873625$$
$$P_3 log_2(P_3) = 0.2\, log_2(0.2) = -0.46438561897747244$$
$$P_4 log_2(P_4) = 0.4\, log_2(0.4) = -0.5287712379549449$$

Summing up all of the terms gives us - 1.8464393446710154. Shannon Entropy is the negative sum, so the final answer is about 1.85 bits.

Let's return to the equation and examine it as a loop instead. The only thing that is changed here is that i starts at 0 and ends at n-1. This is because we start by counting at 0 rather than 1 in Python (and most programming languages).

$$H = -\sum_{i=0}^{n-1} p_i log_2(p_i)$$

Here is the code. On line 1, the probability distribution is defined as lowercase p because we generally use lowercase for variable names. On line 2, we define a variable, h, that will eventually contain the entropy of the distribution. Line 3 is effectively the sigma above. It sets the initial condition implicitly as zero. It ends before the length of the p, which is 4. So it iterates through values: 0, 1, 2, 3. Line 4 pulls out individual elements of p and multiplies them by their log2 transformation. Line 4 uses -= rather than += because we want the negative sum in the end. Alternatively we could multiply the final answer by -1.

```
1     p = [0.3, 0.1, 0.2, 0.4]
2     h = 0
3     for i in range(len(p)):
4           h -= p[i] * math.log2(p[i])
```

Here's another way to write the same code using a "for item in container" strategy. Here, *vals* is plural because lists contain multiple items. Similarly, *val* represents one item of the group.

```
vals = [0.3, 0.1, 0.2, 0.4]
h = 0
for val in vals:
      h -= val * math.log2(val)
```

Now that you know how to convert math equations to Python loops, consider the following two equations that can be used to measure the difference between two probability distributions P and Q. D_{KL} is known as Kullback-Lieberler distance (or divergence). D_T is known as taxicab distance or Manhattan distance.

$$D_{KL}(P||Q) = \sum_{i=1}^{n} p_i log_2\left(\frac{p_i}{q_i}\right)$$

$$D_T(P||Q) = \sum_{i=1}^{n} \left|p_i - q_i\right|$$