This code pipeline attempts to mimic the core functionality of previous StickWRLD code in a more efficient, clear, and flexible fashion. Of particular note:

* The only non-standard Python libraries used are MatPlotLib and SciPy, which is used to display and view 3-D sequence correlations
* The code is operating system independent – e.g., can be run on Windows, Linux, and Mac without issue
* The code is biological symbol independent – e.g., no prior knowledge is known about the symbols making up a sequence (like amino acids or nucleotides)
* Is capable of processing 283 amino acid sequences of length 16 and generating 5 separate p-value threshold plots in <5 seconds

We calculate p-values for every possible combination of symbols between position i and positions (0 to i and i to n), where i is in the initial position of interest and n is the total number of positions in the sequence. P-values are generated via SciPy’s binomial distributions, where cdf is a function of x, n, and p. Take for example calculating the p-value of symbol1 occurring at position1 given symbol2 occurring at position2.

* x is the number of observed symbol1 in the downselected sequences that contain symbol2 at position2
* n is the number of downselected sequences
* p is the expected probability of observing symbol1 at position1 with no priors (e.g., in the entire sequence dataset)

\*Note: When x > n \* p, the survival function (1 – cdf) is calculated instead of the cdf.

Further areas for future development of the code pipeline could include:

* Automating the normalization of data for removing uninteresting relationships (e.g., Watson creek, amino acid electric charges, etc.)
* Add auto-scaling plot features for increasing plot size, etc.
* Perform machine learning on correlations for… classification model? Predict core sequence features?

All code steps are driven from **master.py**, which gets necessary variables & paths from **config.py**.

Computation steps (rough):

* Sequences are read into the program from a FASTA file (**load\_in\_seqs.py)**
  + Currently, we assume that sequences are in the same frame and are the same length. We could add an option later to process unnormalized sequences
* Sequences are next sent to **get\_position\_distributions.py**, where i) all unique symbols (sans gaps) in the sequence are found, and ii) the symbol-distribution at each position on the sequence are found
* A list of every possible symbol-symbol relationship between every position is analyzed for its p-value (per the description above) in **get\_correlation\_matrices.py**
  + This correlation list is what each vector – e.g., “edge” – is built from
* A list of each symbol’s fractional representation at each position is created
  + This list is what each symbol – e.g., “ball” is built from
* All symbols and the vectors in between them are plotted on a cylinder with a fixed circle circumference and cylinder height in **plotting.py**
  + Symbols are arbitrarily assigned a height plane and color
  + Each of the symbol balls are scaled by their fractional representation (larger balls indicate that the symbol occurs more often at that position)
  + Each of the vectors are scaled by their p-value (lower p-values result in a thicker vector)
  + NOTE: There can be 2 vectors that exist between two symbols – 1 describing the relationship of each symbol to the other one.
    - E.g., “what is the likelihood of an A at position 1 given a G at position 5” versus “what is the likelihood of a G at position 5 given an A at position 1”
    - At the moment, the plot does not visualize this directional relationship. Adding arrow heads could accomplish this, but muddy the data viewing
  + For each vector, we evaluate whether that vector’s p-value falls into the the p-value threshold range (inclusive). If so, plot the vector.