Partial Permutations Documentation

What are partial permutations?

A permutation is a re-arrangement of a given sequence in a different order. For example, if you have a sequence: 1, 2, 3. A permutation of that sequence would be 2, 3, 1. In some cases, it might appear as though the term 'combination' is the same as permutation, however, these terms are mutually exclusive and cannot be used interchangeably.

A partial permutation is therefore re-ordering the sequence, just in a particular region instead of the whole sequence. As an example, if your sequence was 1, 2, 3, 4, 5; you would only re-arrange 1, 2, and 3. The initial sequence is known as a **collection of objects**, and is assigned the letter **n**. Whereas the partial sequence (the latter sequence) is assigned the letter **k**.

It is often important to find the maximum number of different partial permutations possible given a sequence. This can be solved using the following equation:

*Maximum number of partial permutations =*

*n!/(n-k)!*

The value quickly becomes very large, and depending on the use-case scenario, the **modulo** becomes very useful. In Python, the modulo sign is '***%***'. It is usually used as follows: *a % b* where 'a' is the value being divided, and 'b' is the value by which you are dividing. The value returned from the function is the remainder from the division. For example, if you were dividing 5 by 3, 2 will be returned.

Implementation

In the application, first the user is asked to input the values for 'n', and 'k'. The *permutation* class is then called which plugs the values into the permutation equation outlined earlier. However, Python does not have a dedicated command to calculate factorials; instead, a separate algorithm is required to process such a request. Therefore, the *permutation* class calls on a new class – *factorial* to calculate the value for the numerator, and denominator in the equation.