1 Basic Questions

- Log into the computer system using the name and password from your paper.
- The first thing to do is work through a tutorial that has been prepared to introduce you to the basics of programming in Sage. You can retrieve this tutorial at

  `http://www.math.utah.edu/~carapezz/tutorial.pdf`

- Now that you have seen the basics, it is time to build a working familiarity with Sage. Try to following exercises, and don't hesitate to refer back to the tutorial.

  1. We write "$n!$" to denote the number $n \cdot (n-1) \cdot (n-2) \cdot ... \cdot 3 \cdot 2 \cdot 1$. This is pronounced "en factorial." Write a function `factorial(n)` so that when given a positive integer n it returns $n!$. Test it by putting in `factorial(5)` and `factorial(6)`; you should get 120 and 720, respectively.
  2. If we want to know how many ways we can choose a set of k objects from a set of n objects, where repeats are not allowed and order doesn't matter, we use the combination formula

     $$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

     Write a function `choose(n,k)` to compute this. This should incorporate the `factorial(n)` function you wrote. Check it by computing `choose(7,3)` and `choose(15,7)` which should give 35 and 6435.
  3. Write a function `ListFactorial(list)` that takes a list and returns a new list where each new entry is the factorial of the old entry. For example, this function should take the list [1,2,3,4,5] to the list [1,2,6,24,120].

2 Words to numbers and back

To do what we want to do with cryptography we will need to be able to translate between letters and numbers. Sage makes this relatively painless through the use of the `ord` and `chr` functions. Basically, Sage has already assigned a number to all capital and lowercase letters, as well as punctuation marks, and these two functions provide the translation. To get an idea for this, evaluate the commands `ord('A'), ord('B'), ord('C'), chr(65), chr(66), chr(67)`.

- Write a function `toNumbers(message)` that takes in a string of capital letters to a list of numbers. Do this so that A corresponds to 0, B corresponds to 1 and so on. For example, evaluating `toNumbers('HELLO')` should return `[7,4,11,11,14]`. It may help to think about building your funcition in steps:

  1. Convert the string of letters to a list of letters.
  2. Convert the list of letters to a list of numbers using `ord`.
  3. Shift the numbers down so that A is 0, B is 1 and so on.

- Write a function `toWords(list)` that takes a list of numbers and outputs a word, thinking of A as 0 and so on. You can think of doing this in steps that reverse the process that changes from words to numbers,

  1. Shift the numbers up so that A is 65, B is 66 and so on.
  2. Convert the list of numbers to a list of letters using `chr`
  3. Convert the list of letters to a string of letters by creating a blank string and adding letters to it one at a time from the list.

If you get these working you may want to try modifying them to include spaces and lowercase letters (maybe even punctuation). You can use `ord` to determine what numbers these characters correspond to.

3 Bonus Questions

The following questions are taken from the website `https://projecteuler.net`. This is a great website that presents questions that require mathematical insight as well as some programming ability. Feel free to do these problems in any order.

- Determine how many numbers below 1000 are multiples of both 3 and 5.
- The Fibonacci sequence is $1, 1, 2, 3, 5, 8, 13, 21, ...$ The structure is, start with 1,1 and then the next number is the sum of the two previous numbers. Compute the sum of all even Fibonacci numbers smaller than one-million.
- A Pythagorean triple is a triple of integers (a, b, c) where $a^2 + b^2 = c^2$. It is a fact that there is only one Pythagorean triple where $a + b + c = 1000$; find it.
- If one sums the digits in $10! = 3628800$ one gets $3 + 6 + 2 + 8 + 8 + 0 + 0 = 27$. Determine the sum of the digits in 100!.