1 Shift and Affine Ciphers

- If you did not get the functions `toWords` and `toNumbers` working yesterday, do that now; These will be indespensible tools for what we do next.
- Working with lists is going to feature prominently today. Copy-paste the example below into the Sage worksheet for a quick reminder of some essential ways in which we work with lists,

```
A=[]
for x in range(0,20):
    if x%3==0:
         A.append(x)
print A

print A[5]

print len(A)

for x in range(0,len(A)):
    A[x]=A[x]+1

print A

for x in range(0,len(A)):
    if x%2==0:
        A[x]=A[x]+1
    else:
        A[x]=A[x]-1
print A
```

- Write a function `Caesar(message,shift)` that takes in a string and a number and implement a Caesar cipher by shifting each letter in the string by the number given. You will definitely want to use the `toWords` and `toNumbers` functions to do this. . Putting in `Caesar("AUTONOMOUSZONE",3)` should give you 'DXWRQRPRXVCRQH'. What shift should be used to decrypt that message?
- Write a function `affine(message,a,b)` that takes a message and two integers, a and b, and encrypts the message using the affine cipher $x \mapsto a \cdot x + b$. One way to do this is to copy-paste your code for the Caesar cipher and just add in the extra step of multiplication. Try encoding the message 'ALLWORKANDNOPLAY' using a = 3 and b = 5. You should get 'FMMTVEJFSOSVYMFZ'.
- Now it is time to try to break these. One way to break the Caesar cipher is by brute force; there are only 26 possible shifts and it is a small task for a computer to try them all. Write a function `bCaesar(message)` that takes in a message and outputs all 26 shifted messages. This can be accomplished with a single `for` loop that calls the Caesar function you've written for each shift value 1 though 25. Use your code to decrypt the message 'OVTOEBGUREYBIRFLBH'.
- Breaking the affine cipher using brute force is not so unreasonable, so we're going to do it. Write a function `baffine(message)` that outputs all possible affine decryptions of a message. You can accomplish this by writing a `for` loop inside of a `for` loop; one loop cycles through possible mulitplications while the other cycles through possible subtractions. If you try multiplying and adding by every number, that gives 26^2 outputs for you to sort through. However, keep in mind that you do not need to consider all 26 multiplications, only those that are invertible mod 26. Use your function to decode the message

  PCNSNNYRLGUFAXNLAHLSHFKVONKXXNLAHSNNYFCNAOINNGBNUFAXFYDHSNNYJ-
  FADPYBRNSVYHFUUUFASLYGHPCNLKJLXHSNNYRLGPDYFJPCNSNNYRLGKPDNOINR-
  FHOFUVHCNAXILAGOFNQMKLPYJIXXFVANRLGNCNYPUXFVANYFORLG

2 Vigenere Cipher

- We want to write a function `Vigenere(message,key)` that implements the Vigenere cipher. The basic idea is the following: change the message and the key strings to lists of numbers and then go through the message list and add the corresponding entry in the key list, cycling back to the beginning of the key list as needed to make it to the end of the message list. There are several ways to approach this. Run your code on the message 'CALLMEISHMAEL' with the keyword 'WHALE' and you should get output 'YHLWQAPSSQWLL'.

3 Bonus Questions

- Write a program that will take a text encrypted with a Caesar cipher and make a guess as to what the original message was (instead of outputting all possibilities). You will probably want to use frequency analysis. Try decoding the following messages:
  - SDFZX KFRUAKFMGAKFZUFSKFGFVGXZXOJMKFOTFGFVKGXFZXKK
  - GVSESFNBASNFBHARNGVOGNABNBASNYABJFNJVOGNRBSFNGVSNTBKNFOL
  - HKJ KJWYNE CAWEOWBXHHEJCW KSJWIUWBXENWHX U
  - P PGOAMO CJTOMCIOSCOATOKFCBVOHCORPGHOATOCUUOSXGRCIFHTCIG M

  Something that you may find useful is that Sage can count how many times a letter appears in a message. That is, you can type "ILL ALWAYS LOVE YOU MOM".count("L") and Sage will return 4. You can also find a table of letter frequency by typing "letter frequency" into Google.

- If you square the first ten natural numbers and add you get $1^2 + 2^2 + ... + 10^2 = 385$. If you instead add the numbers and square, you get $(1+2+3+...+10)^2 = 3025$. The difference between these two is 3025-385 = 2640. Figure out what the difference is if you did this with the first 1000 numbers instead of the first 10.

- Yesterday a bonus question asked you to write a program that generates the Fibonacci numbers. If you haven't done this then you should give it a try (or try again); it's a pretty good exercise in basic programming. Once you've got this working, determine what the 1000th Fibonacci number is.

- Triangle numbers are numbers that are formed by adding the first consecutive natural numbers. So the first triangle number is 1, the second is 1+2=3, the third is 1+2+3=6, the fourth is 1+2+3+4=10, and so on. Determine which triangle number is the first to be evenly divisible by at least five different numbers. Determine which triangle number is the first to be evenly divisible by at least 100 different numbers.