Lesson Seventeen

Math 6080 (for the Masters Teaching Program), Summer 2020

17. Ciphers. A cipher encodes a message by replacing each letter of the message with another via a bijective function:

$$f : \{\text{letters of the alphabet} \to \{\text{letters of the alphabet}\}$$

In the cipher, a is replaced by $f(a)$, b is replaced by $f(b)$, etc.

We will ignore cases, and assign a number between 1 and 26 to each letter:

$$\text{a or A} \leftrightarrow 1, \text{b or B} \leftrightarrow 2, \cdots, \text{z or Z} \leftrightarrow 26$$

so that we can reinterpret f as a bijective function on the numbers from 1 to 26:

$$f : \{1, 2, ..., 26\} \to \{1, 2, ..., 26\}$$

Exercise. Complete the following table using Python:

a	b	c	⋯	x	.y	z
1	2	3	⋯	24	25	26

Ciphers Based on Addition and on Multiplication.

(i) **Addition.** Think of 1 to 26 (with $26\%26 = 0$) as the numbers modulo 26. Pick a number $r \in \{1, ..., 25\}$ and let f be the function:

$$f(x) = (x + r)\%26$$

This is a cipher that shifts the letters forward by r units. (It seems Julius Caesar was fond of shifting by 3.) The function f is a bijection, and the inverse to f is the shift by r units backwrds, or (if you prefer shifting forward), the shift forward by $26 - r$. Caesar would thus encode:

'happy birthday' as 'kdssb eluwkgdb'

(if he spoke English, and if he cared to wish anyone a happy birthday).

(ii) **Multiplication.** In this case, we think of the numbers $\{1, ..., 26\}$ as all the nonzero numbers modulo 27. Pick a number r with $\gcd(r, 27) = 1$ (i.e. r is any of the 18 numbers not divisible by three). Then we saw in Lesson Fifteen that:

$$f(x) = (rx)\%27$$

is a bijective function, with inverse function $g(y) = (ay)\%27$ where a comes from the enhanced Euclid's algorithm:

$$ar + b \cdot 27 = 1$$

Exercise. Prompt the user for some text.

(i) Prompt the user for a number between 1 and 25, and then encode the text (leaving anything that is not a letter alone, and reducing all letters to lower case) via the shift cipher. Offer to decode the message for the user.

(ii) Do the same for a number relatively prime to 27 and multiplication.

Remark. If you type ord('a') or ord('A') into Python, you get the "ascii" values of a and A. Note them down and note that the ascii values of b,c,d,e,... and B,C,D,E... progress as you would expect. This, along with the inverse function chr(n), is a time-saver for Python programs.