MAT 305: Lab #10

May 2, 2016

Background

Two applications of algebra and number theory are **coding theory** and **cryptography**:

- The aim of *coding theory* is transmit information *reliably,* detecting errors and, when possible, correcting them.
- The aim of *cryptography* is to transmit information *securely,* so that an eavesdropper can neither understand nor work out the meaning of the transmission.

To apply mathematical ideas, both require a preliminary stage of transforming text into numbers, and vice versa. You will write two or more functions to do help someone this; one will be interactive.

When transforming text into numbers, it is necessary first to group the text into conveniently-sized blocks. For instance, the phrase

```
GET OUT OF DODGE
```

can be groups several different ways. One way is to group it into blocks of two:

```
GE TO UT OF DO DG EX
```

but one could also group it into blocks of four:

```
GETO UTOF DODG EXXX .
```

In both cases, the number of letters in a group does not divide the message's length evenly, so we pad the message with `X`'s. It doesn't have to be `X`'s, of course.

How does one do this? Python has a command that lets you convert each *character* into a number. (If you've forgotten it, it appears in the book, and you used it in a previous assignnment, so it would be good to review that, as I might ask about it on the exam!) In our case, this command would convert our phrase into the numbers

$$71, 69, 84, 79, 85, 84, 79, 70, 68, 79, 68, 71, 69 .$$

You'll notice that I've removed the spaces, and haven't yet created the groups. How do we then group them? First, turn them into the numbers `0`–`25` by subtracting the number corresponding to `A`. That gives us

$$6, 4, 19, 14, 20, 19, 14, 5, 3, 14, 3, 6, 4\,.$$

To group them, we reason the following:

- We don't need lower-case letters or punctuation to get our meaning across. (Related trivia: Ancient written languages typically used all upper-case letters with no punctuation or even spacing.)
- Thus, every value to encode lies between 1 and 26.
- We can use a base-26 number system to encode any group of letters.

So, to encode a group of four letters with numerical values a, b, c, d, we can compute

$$m = a + 26b + 26^2c + 26^3d\,.$$

In general, to encode a group of n letters with numerical values $a_1, a_2, \ldots, a_n$, compute

$$m = a_1 + 26a_2 + 26^2a_3 + \cdots + 26^{n-1}a_n\,.$$

To *decode* an encoded group m of n characters, do the following n times:

- determine the remainder of dividing m by 26, and call it a;
- convert a back into a character; and finally,
- replace m by ${}^{m-a}/_{26}$.

You need to write at least two functions.

1. The first function, `encode()`, takes two inputs: an integer n, and a list L of characters. It converts each character into a number, organizes the numbers into groups of n characters, then converts each group into one number using the formula above. It returns a list M of numbers, one for each group.
2. The second function is interactive. It offers the user a text box and a slider. In the text box, the user inputs a message. In the slider, the user selects from 2 to 6 characters. The function takes these values and sends them to encode, then prints the list numbers that `encode()` returns.

For extra credit, you can also:

3. Implement a third function, `decode()`, takes two inputs: an integer n, and a list M of integers. It converts each integer into a group of n integers, then converts each integer into a character.
4. You can Cythonize either `encode()` or `decode()`. At least one variable's type must be declared.

It is expected that you will need help, and will ask questions.