Week 4 - Wednesday

COMP 1800

Last time

- What did we talk about last time?
- Looping over strings
- Cryptography
- Transposition cipher
- Shift cipher
- **`ord()`** and **`chr()`** functions

Questions?

Review of `ord()` and `chr()`

- We can convert a string with a single character in it into the integer that represents it with the **`ord()`** function

```
number = ord('a') # number contains 97
```

- If you know the numerical value of a character, you can convert that number back into a string using the **`chr()`** function

```
letter = chr(100) # letter contains 'd'
```

Shift Cipher

Definition

- A shift cipher encrypts a message by shifting all of the letters down in the alphabet
- Using the Latin alphabet, there are 26 (well, 25) possible shift ciphers
- We can model a shift cipher by thinking of the letters A, B, C, … Z as 0, 1, 2, … 25
- Then, we let the key ***k*** be the shift
- For a given letter with value ***x***:

encrypt (***x***) = (***x*** + ***k***) mod 26

Example: Caesar Cipher

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

- ***E***("KILL EDWARD") = "NLOO HGZDUG"
- What is ***E***("I DRINK YOUR MILKSHAKE")?
- What is ***D***("EUHDNLWGRZQ")?
- This code was actually used by Julius Caesar who used it to send messages to his generals

Shift encryption in Python

- Algorithm:
  - Loop over all characters
    - Convert character to ASCII value
    - Convert ASCII value to a value from 0-25 by subtracting the value of `'A'`
    - Add the key to the result
    - Compute the result modulus 26 (which makes numbers bigger than 25 wrap around)
    - Add back the value of `'A'` to turn a value from 0-25 back into an ASCII value
    - Turn the ASCII value back into a character and concatenate it onto the ciphertext
  - Return the ciphertext

```
def shiftEncrypt(plaintext, key):
```

Shift decryption in Python

- Reversing the process to decrypt the ciphertext is simple
- All we need to do is "encrypt" the ciphertext with the negation of the key we used to encrypt
- For example, if we encrypted with a key of 7, we can decrypt by encrypting with a key of -7
- Our decrypt function should simply call the encrypt function with a negative key

```
def shiftDecrypt(ciphertext, key):
```

Quick note

- Our implementation expects all input characters to be from `'A'` up to `'Z'`
- That's why subtracting `ord('A')` will make the values be between 0 and 25
- Inputting strings that contain characters other than uppercase letters (e.g. digits, lowercase letters, punctuation) will cause strange results

Substitution Ciphers

Substitution ciphers

- **Substitution ciphers** cover a wide range of possible ciphers, including the shift cipher
- In a substitution cipher, each element of the plaintext is substituted for some corresponding element of the ciphertext
- **Monoalphabetic** substitution ciphers always use the same substitutions for a letter (or given sequence of letters)
- **Polyalphabetic** substitution ciphers use different substitutions throughout the encryption process

Example: Simple Monoalphabetic Substitution Cipher

- We can map to a random permutation of letters
- For example:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
I	N	O	V	Z	H	A	P	T	R	G	E	U	F	D	W	S	B	Q	Y	L	K	M	J	C	X

- E("MATH IS GREAT") = "UIYP TQ ABZIY"
- 26! possible permutations
- Hard to check every one

Example continued

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
I	N	O	V	Z	H	A	P	T	R	G	E	U	F	D	W	S	B	Q	Y	L	K	M	J	C	X

- Using the same mapping, perform the following encryption:
- ***E***("HELP ME") =
- Perform the following decryption:
- ***D***("VD CDL QZZ YPZ HFDBV") =

Substitution encryption in Python

- Algorithm:
  - Loop over all characters
    - Convert character to ASCII value
    - Convert ASCII value to a value from 0-25 by subtracting the value of **`'A'`**
    - Use this value as an index into the scrambled **`key`** alphabet
    - Concatenate the character at this location onto the ciphertext
  - Return the ciphertext

```
def substitutionEncrypt(plaintext, key):
```

Quick note

- Our version of the substitution cipher is different from the book's
- We only use uppercase letters
- We don't have a space
- We don't have to use the **`find()`** function
  - But it's a good idea to learn about **`find()`** on your own

Making a key

- One of the annoying things about using a substitution cipher is that you have to come up with a permutation (a scrambling) of the alphabet
- But Python can help us do that!

A useful function

- First, we need a function that:
  - Takes a string and an index
  - Returns a new string with that index removed
  - In other words, we return the string with all the characters before the index concatenated to all the characters after the index
- We can do that in a single line of Python

```
def removeChar(string, index):
```

Making a key

- Algorithm:
  - Create a string that holds all the characters of the alphabet
  - Loop as many times as the length of the alphabet:
    - Pick a random integer between 0 and the remaining length of the alphabet
    - Put the character at that location at the end of the key we're making
    - Use **`removeChar()`** to remove the character at that location from the alphabet
  - Return the key

```
def makeKey():
```

Quiz

Upcoming

Next time…

- Vigenère cipher
- Work time for Assignment 3

Reminders

- Read Section 3.7 of the textbook
- Work on Assignment 3