CSE 231, Fall 2009

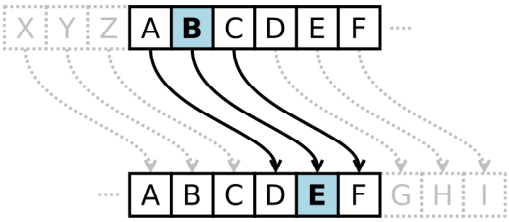
Programming Project 05

Assignment Overview

This assignment is worth 40 points (4.0% of the course grade) and must be completed and turned in before 11:59pm on Monday, October 19, 2007. The purpose of this project is to familiarize you with the use of lists and looping. For this assignment, you will create a decryption program.

Background

The Caesar cipher is named after Julius Caesar who used this type of encryption to keep his military communications secret. A Caesar cipher replaces each plain-text letter with one that is a fixed number of places down the alphabet. The plain-text is your original message; the cipher-text is the encrypted message. The example shown below is a shift of three so that "B" in the plain-text becomes "E" in the cipher-text, a "C" becomes "E", and so on. The mapping wraps around so that "X" maps to "A" and so on.



Here is the complete mapping for a shift of three:

Plain: ABCDEFGHIJKLMNOPQRSTUVWXYZ
Cipher: DEFGHIJKLMNOPQRSTUVWXYZABC

To encrypt a message simply substitute the plain-text letters with the corresponding cipher-text letter. For example, here is an encryption of "the quick brown fox jumps over the lazy dog" using our shift-three cipher (case is ignored):

Plaintext: the quick brown fox jumps over the lazy dog
Ciphertext: WKH TXLFN EURZQ IRA MXPSV RYHU WKH ODCB GRJ

To decrypt the message simply reverse the process.

The encryption can also be represented using modular arithmetic after first transforming the letters into numbers according to the scheme: A = 0, B = 1, etc. (which is the index of the alphabet if it is in a string or a list). A shift-three cipher will take the number of each letter (`plainTextChar`), add 3 (the shift), and then find the remainder after dividing by 26 to get the cipherText:

```
cipherTextChar = (plainTextChar + 3) % 26
```

Program Specifications
Your task is to write a program that can decrypt a message that has been encoded using a Caesar cipher. Stated another way, you need to find the "shift" for the cipher. Once you have the shift, you know the mapping so you can decrypt the message.

Caesar Cipher Cracking
To find the "shift" you need to know about cracking Caesar ciphers using a technique that has been around for over a thousand years. Any language such as English has a known distribution for each letter. For example, the letter "E" is the most common letter in English making up 12.702% of the letters on average (ignoring case). The letter "T" is next (9.056%), followed by "A" (8.17%), and so on. The order "E"-"T"-"A" is what matters, not the percentage, but we provide the complete distribution for the curious (in the file "letter_frequencies").

The procedure begins by finding the most common letter. You can guess that the most common letter maps to "E." You can now find the "shift" from the most common letter in the cipher-text to the expected most common letter "E". For example, if the most common letter in the cipher-text is "H", you know that the shift from "E" to "H" is 3. You should check that the shift for the next most common letter "T", and third most common letter "A" is also 3. Once you know the shift, you can apply the shift to all the letters in the cipher-text and get the original plain-text message.

What about spaces between words and punctuation? In real world cipher-text there are no spaces or punctuation because those are useful clues for deciphering. In the cipher-text we provide for this project there is no punctuation, but we have left spaces in the cipher-text because they will be helpful for you to recognize that your deciphering is correct or not. You will need to ignore spaces when counting letters (if you forget to ignore them, beware that the space will be the most common character).

Your high level algorithm will be:
1. Read the cipher-text (explained below)
2. Get a count of each character in the entire cipher-text (ignore spaces)
3. Find the most common character
4. Find the shift from "E" to that most common character.
5. Check that the shift also works for the next most common.
6. Using the shift, decode each character of the cipher-text and print

Deliverables
You must use handin to turn in a file called **proj05.py** – this is your source code solution; be sure to include your section, the date, the project number and comments describing your code. Please be sure to use the specified file name, and save a copy of your proj05.py file to your H drive as a backup.

Assignment Notes
1. Reading the cipher-text file. You need to import file reading from the os module to get the "file" command. Copy the cipherText file to the same folder as your program and

create this program that will read in the file and print it out. (Be careful that this document does shows fancier double quotes than you need to use in IDLE.)

```
words = ""
for line in file("cipherText"):
    print line   # print the next file line
    words = words + line + " "
print words      # print all the words in the file
```

2. The `ord()` function will be very useful. It generates the ASCII integer value of a character. For example type `ord('a')` into a Python shell and it will return the integer 97. If you create a list of counts—one count for each letter—you can use the `ord()` function to index into that list. For example, if `letter = 'c'` you can find it's index using `index = ord(letter) – ord('a').` Try it in the Python shell. The index for 'a' will be 0, for 'b' it will be 1, for 'c' it will be 2, and so on.

   The inverse to the `ord()` function is the `chr()` function. You may or may not need the chr() function for this project—it depends on how you implement it. Try it out in the Python shell.

3. The `ord()` function is also useful for applying the shift to letters. For example, using the formula from above for a shift of "`shift`" begins with `plainTextChar` as a number.

   ```
   cipherTextChar = (plainTextChar + shift) % 26
   ```

   Deciphering works the same way:

   ```
   plainTextChar = (cipherTextChar + shift) % 26
   ```

   results in `plainTextChar` being a number which you can then must convert to a letter.

4. We provide three files. There is a pair for you to use for testing: both cipherText (cipher1) and plainText (plain1) are provided. The final file (cipherText) is the one your program will be tested on.

5. The `max(List)` function finds the maximum value in a list. The `List.index(value)` method will return the index of the `value` in the `List`. Try them out in the Python shell. For example:

   ```
   List = [4, 2, 12, 5]
   value = max(List)
   index = List.index(value)
   ```

   If `List` had 26 values, one for each letter, can you use the `ord()` function to find the

letter associated with a particular `index`? (See suggestion #2 above.)

6. Do the Algorithm steps separately—use the Python shell for testing.
    a. Step 1 to read in is provided above. Test it separately.
    b. Once you have things in a string "word" you can write a small loop that counts occurrences of each character. You can test out indexing into counters using `ord()` within the Python shell before you try to write the loops. If you run Step 1, you will have the "word" string available for testing in the Python shell. Try out some loops on it.
    c. For step 3 use the `max()` function described above to find the character with the maximum count.
    d. Use the `ord()` function and some arithmetic to calculate the shift from "E".
    e. Leave the check of Step 5 until the rest of the program is done.
    f. Use the shift from step 4 and the formula for encryption/decryption (the one with the % operator) to decode each character.