CIS 1152 - Lab #4 Functions and Files

S. Ruegsegger
Modified with permission by Peter Chapin

Objective

This lab will focus on defining and calling functions and reading data from *flat* files.

Task 1: Cipher Decryption

Student skill: integrate a provided function into a PHP script.

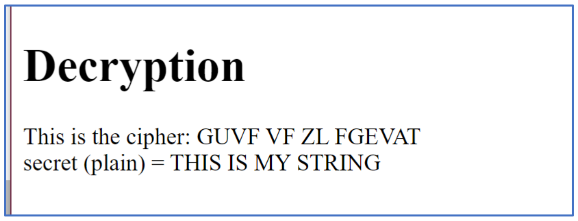
In this task, you will write a decryption script to return the plaintext message from a given ciphertext message. We are going to use the *Caesar Rotation* cipher system, where each character is rotated by a fixed amount through the character set to another character. In this system the cipher key is the rotation distance, and we will hard-code that value. This encryption scheme is also sometimes called ROT-nn where nn is the rotation distance. The character set will be restricted to the 26 uppercase letters of the alphabet plus the space and line breaks. See: https://en.wikipedia.org/wiki/Caesar_cipher.

Be aware that this is not a serious encryption scheme by today's standards (was it ever?). With only 26 possible rotation distances (in our case) there are only 26 possible keys. The method is easily broken by "brute force" whereby the attacker simply tries every key until finding one that produces a meaningful decoded message. We are using this method only as a vehicle for exploring functions and reading text from files.

Requirements:

- Define 2 global vars:
    - `$letters`, which is a string containing all 26 letters of the alphabet (upper case)
    - `$rotation` which is the integer 13. This is the cipher key we will use.
- Read the `cipher.txt` file into a string variable. Hint: the `file_get_contents()` library function is perfect for this situation
- Loop over each char in the string (see *"Looping over chars in a string"* in the Lecture #4 slides)
- For each char, call the `cipher2plain()` function provided. See the class home page in the Samples section. Append the character returned by the function to the plaintext message.
- Outside the loop, print the ciphertext message and the decrypted plaintext message.

For example:



Decryption

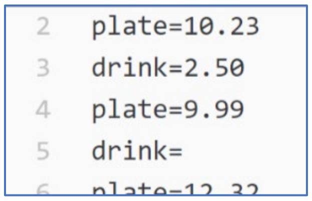
This is the cipher: GUVF VF ZL FGEVAT
secret (plain) = THIS IS MY STRING

You may find several string functions useful when doing this task (or later tasks). The PHP Manual contains a section that lists various string functions of interest.

Task 2: The Restaurant

Student skill: read *key-value pair* data from a file, and put that data into a set of parallel arrays.

Go back to Lab 3 and get your HTML and PHP scripts for the restaurant bill application. You had *hardcoded arrays* for the bill of 3 customers. Now, have your PHP script *read* the data from a provided file. The PHP Manual contains some information on available file handling functions. The file to use is on the class website in the Samples section.



```
plate=10.23
drink=2.50
plate=9.99
drink=
plate=12.32
```

The text file has a *var=value* format, as shown above. Read each line, then *split* the line on the =. The *left* word is the variable name and the *right* is the value. The value could be numeric or string. Watch out: the value or entire line could also be 'blank'.

Notes:

- In the Lab Report writeup, you don't have re-document all the features from Lab 2. Just document the new features for Lab 3.
- The data file has 1 line with *spaces* on either side of the equals. That's not the default, but write the code to deal with it.
- There might be empty lines. You, the programmer, need to deal with it (in your code). This happens all the time.
- We need to "match" the plates to the drinks for each customer. Therefore, the protocol in the datafile is that a plate and drink order will be *paired* together – even if one is null (empty). The plate bill will come first, then the drink bill for that *same* customer.