CSC 401 – Introduction to Programming

Assignment 4

Your assignment is to create a file encrypter & decrypter. You must use lists, sets, and dictionaries as indicated in the description.

Encryption

1) In a file called "assign4.py", create a function called "encrypt_file" that accepts *one string parameter* –the filename to read and encrypt. This function should do the following:

   a) Load the file (specified by the function parameter) into a string. Close the file when done.

   b) Create a set that contains the complete unique set of characters from the file you loaded.

   c) Create a list from with the contents of your set, then shuffle it using the random shuffle method.

   d) Create a new empty dictionary - remember the type is "dict". This will hold out encryption keys & values

   e) For each character in your shuffled list (from step "c"), add the following key-value pairs to the dictionary you just created.

      - Key: the current character (from your shuffled list)
      - Value: the next "popped" element from your set of characters (from step "b").

   - Open an output file named with the same as the file name parameter passed in, plus ".enc" added to the name. *For example, the file name sample.txt would generate the file name "sample.txt.enc".*

   - For each character in your original file content string (from step "a"), use the character as a key to your dictionary and get the associated value - write the value to the output file. Close the file once you are done.

   - Open a 2nd output file named the same as the file name parameter passed in, plus ".json" added to the name. *For example, the file name sample.txt would generate the file name "sample.txt.dict".* Write your entire dictionary to the output file: *output_file.write(json.dumps(encrypt_dict)).* You need to "import json" in order to do this.

   - Close the file once you are done. *(Nothing is returned from this function)*

Executing this function should generate a text file with the ".enc" extension that contains an unreadable encrypted version of the original file, and a file with a ".dict" extension that contains the json version of your encryption dictionary.

If done properly the method should contain less than 20 lines of code. That is not a "hard" number, just a general guideline.

Decrypt

2) In the same file (after the first function), create a function called “decrypt_file” that accepts two parameters – a dictionary file name (.json), and the filename you want to read and decrypt (.enc). This function should do the following:

   a) Open the file json dictionary file sing the json file name passed in.

   b) Load the json file content into a new dictionary: *orig_dict = json.load(your_file_variable).* Close the input file when done.

   c) Next, create a new dictionary – this will be used to hold a reverse version of the dictionary you just loaded (this will allow us to decrypt the encrypted file that as passed in.

   d) For each key (i.e., each character) in the dictionary you loaded in step “b” (to iterate over the keys, just use ‘for x in orig_dict:’), add the following air to your new dictionary (decrypt_dict):

      - Key: the value associated with the current key character (the dictionary value associated with the current key character).

      - Value: the current key character (your loop variable)

Once done, this is your decryption character mapping – how each character will be mapped from an alternate character back to its original character. Make sure this is working before you move on – you should have approximately 99 key-value pairs in your dictionary.

   e) Now open and then load the entire encrypted file (specified by the 2nd function parameter) into a string. Be sure to close the file when done.

   f) Open an output file named the same as the file name parameter passed in, but replace the “.enc” with “.dec”. *For example, the file name sample.txt.enc would generate the file name “sample.txt.dec”.*

   g) For each character in the string containing the encrypted file content (loaded in step “e”), write the character associated with the current character (from your dictionary reversed dictionary). Close the file once you are done. Nothing is returned from this function.

This should generate a text file with the “.dec” extension. That file contains a decrypted version of the encrypted file – which should be the same as the original.

If done properly the method should contain approximately 15 lines of code. That is not a “hard” number, just a general target.

3) After your 2 functions, add the following code. This will work with your functions:

```
source_file = "MobyDick.txt"

print("Encrypting input file " + source_file)
encrypt_file(source_file)

print("-"*20)

print("Decrypting input file " + source_file)
decrypt_file(source_file + ".dict", source_file + ".enc")
```

NOTE: There is no output generated from this program, except the 2 statements printed in the main.py file:

```
Encrypting input file Dracula.txt

--------------------

Decrypting input file Dracula.txt
```

The actual output will be the ".enc", the ".json", and ".dec" files.

Extra Credit (up to 20 points)

Note - this problem is unrelated to the encryption and decryption functions.

Create a function called "top_word" that accepts one string parameter – a filename. This function should load the specified file and return the most-used word in that file, and the number of times it was used. The returned data should be returned as a tuple: (*top_word, num_instances*). Your solution method should contain approximately 20 lines of python code.

Note – upon reading the file content, you should remove the following characters: `, $ * ; . & ! ( - )`

You should add this function to your python file after the 2 functions previously discussed.

Add the following line to the end of your "main.py" file to execute this function:

```
print(top_word("MobyDick.txt "))
```

Expected output: `Top Word: ('the', 13,704)`

Submission

- Your submission should consist of your Python ".py" file, submitted via D2L.
- This assignment is due (submitted via D2L) before the start of class in *1 week*
- NO LATE ASSIGNMENTS CAN BE ACCEPTED.
- You may email me with any questions on this assignment at any time between now and the due date at chield@depaul.edu or christopher.hield@gmail.com.