|  |  |  |
| --- | --- | --- |
| American University of SharjahCollege of Engineering Department of Computer Science & Engineering  P. O. Box 26666, Sharjah, UAE |  | **Lab Instructor:** Praveena Kolli  **Office:** EB2126  **Phone**: 971-6-5152352  **e-mail**: pkolli@aus.edu  **Semester**: Spring 2020 |

**CMP 321 - Programming languages Laboratory**

**Lab 8 – Python Revision**

**Objectives**

* Revising Python concepts and making full use of the language features.

**Please explore and make use of Python features where possible. Code that does not follow this note will be penalized.**

**Exercise 1:**

1. Write a *RandomCipher* function that builds a dictionary where each of the 26 letters of the alphabet is randomly matched to another (all lowercase). Each of the matching letters should be unique (in other words, your function creates a bijective mapping). Example:

{'a': 's', 'b': 'j', 'c': 'w', 'd': 'e', 'e': 'g', 'f': 'i', 'g': 'y', 'h': 'c', 'i': 'p', 'j': 'f', 'k': 'l', 'l': 'm', 'm': 'r', 'n': 'h', 'o': 'o', 'p': 'u', 'q': 'k', 'r': 'b', 's': 'x', 't': 'q', 'u': 'd', 'v': 'z', 'w': 'a', 'x': 't', 'y': 'v', 'z': 'n'}

1. Write an *Encode* function that takes a plain text string as input and returns a new string encrypted using your cypher in part (a). Make sure to replace all letters once only; leave punctuation and other non-alphabetical characters as is. Example: if using the above mapping, the text "hellow, world!" would be encoded as "cgmmoa, aobme!"
2. Write a *Decode* function that takes an encrypted text string as input and returns a new string decrypted using your cypher in part (a). Create any intermediate data structure as required to make your code efficient. Example: "cgmmoa, aobme!" should be decoded as "hellow, world!" (Encoding followed by decoding should always produce the same string.)

**Exercise 2:**

1. Use *urllib* to download the 500 most frequent English words (sorted by frequency) from <https://sketchengine.co.uk/wp-content/uploads/word-list/english/english-word-list-total.csv>. Process the file and store the words (only) in a suitable data structure *W*, keeping them in order. Print out the 20 most frequent English words and their rank, as follows: 1. 'the' - 2. 'and' - 3. 'to' - …
2. Download a story of famous computer bugs from <http://textfiles.com/100/famous.bug> as string *S.*
3. Remove from the text *S* all words that are not in *W.* Make sure your code is efficient.
4. Perform a word frequency count for the words in *S* and store the resulting data. Print the 20 most frequent words in string *S* using the same format shown in part (b).
5. Find and print common words between 20 most frequent words in the text to the 20 most frequent words in the English language.
6. Compare the order of 20 words frequency between two files and print out any major discrepancy. For example, it might be that the most frequent word is the same but the second most frequent word is different, etc. Keep efficiency in mind, as before.