PROGRAMMING USING PYTHON

Lecturer: Doctor Bui Thanh Hung
Data Science Laboratory
Faculty of Information Technology
Industrial University of Ho Chi Minh city
Email: hung.buithanhcs@gmail.com (buithanhhung@iuh.edu.vn)
Website: https://sites.google.com/site/hungthanhbui1980/

Phần 1: Hoàn thành các bài tập trên lớp

1. Giải và biện luận Phương trình bậc 2 ax2 + bx+ c=0 (Vào/Ra, Sử dụng thư viện, Cấu trúc if và Giải thuật biện luận PT bậc 2)
2. Nhập vào số n, kiểm tra n có phải là số nguyên tố hay không (Sử dụng For)
3. Chuyển đổi 2 thành hàm (Function)
4. In ra 10 số nguyên tố lớn hơn n, sử dụng hàm ở 3 (vòng lặp while)
5. Chuyển đổi 3 thành thư viện giống math.sqrt và giải quyết bài toán: nhập vào 2 số a,b in ra các số nguyên tố trong khoảng [a,b]
6. Sử dụng Advanced Function để sắp xếp các cạnh cho đồ thị có trọng số với dữ liệu đầu vào là Đồ thị là các mảng nhiều chiều [đỉnh nguồn, đỉnh đích, Trọng số] ra là các cạnh được sắp xếp theo thứ tự giảm dần. Yêu cầu chỉ sử dụng 3 dòng lệnh: 1- Vào dữ liệu, 2- Sử dụng Lambda + Sort, 3- In ra kết quả
7. Xây dựng cấu trúc dữ liệu Stack và sử dụng cấu trúc dữ liệu này để check 1 biểu thức có đủ dấu hay không (giả sử có tối đa 3 loại ngoặc). In giá trị phép toán ra màn hình
   Ví dụ: (4+3 : thiếu dấu
   7+[(8+9)-5]) đủ dấu – Kết quả: 19
   (4+3))]: dư dấu

Phần 2: Hoàn thành các bài tập sau:

09. Đảo ngược xâu ký tự

Hãy đảo ngược xâu ký tự “stressed” (theo thứ tự từ cuối xâu đến đầu xâu ký tự).

10. Trích xuất ký tự từ xâu ký tự

Từ xâu ký tự “MpyaktQrBoilk RCSahr”, hãy trích xuất các ký tự ở vị trí 2,4,6,8,10,12,14,16,18,20 và kết hợp theo thứ tự đó để tạo thành 1 xâu ký tự mới (ký tự space cũng được tính, các ký tự được đánh số từ 1).

02. Kết hợp hai xâu ký tự

Hãy kết hợp hai xâu ký tự “Partrol” và “Car” để tạo thành xâu mới “PatrolCar”.

03. Tokenize và thống kê số lượng ký tự của mỗi từ

1. Tokenize câu sau: “Now I need a drink, alcoholic of course, after the heavy lectures involving quantum mechanics.”
2. Đưa ra danh sách gồm số ký tự alphabet trong mỗi từ theo thứ tự xuất hiện của từ đó trong câu.

04. Ký tự thành phần

1. Tokenize câu sau: “Hi He Lied Because Boron Could Not Oxidize Fluorine. New Nations Might Also Sign Peace Security Clause. Arthur King Can.”
2. Lấy ra ký tự đầu tiên của các từ ở vị trí 1, 5, 6, 7, 8, 9, 15, 16, 19; với các từ còn lại lấy ra 2 ký tự đầu tiên. Tạo ra một map từ các xâu ký tự được trích ra tới vị trí của từ trong câu.

05. n-gram

1. Viết hàm sinh ra tất cả các n-gram từ một dãy cho trước (xâu ký tự hoặc danh sách).
2. Sử dụng hàm đã viết, sinh ra word bi-gram và character bi-gram từ câu sau: “I am an NLPer”

06. Tập hợp

1. Sinh ra tập X và Y tương ứng là tập các character bi-gram từ hai xâu ký tự "paraparaparadise" và "paragraph".
2. Sinh ra các tập hợp union, intersection và difference của X và Y
3. Kiểm tra xem bi-gram 'se' có thuộc tập X (Y) hay không?

07. Sinh ra câu từ template

Viết hàm số nhận vào 3 biến x, y, z và trả về xâu ký tự "y vào lúc x giờ là z" Sinh ra kết quả với các giá trị x, y, z sau đây x="12" y="Nhiệt độ" z=22.4

08. Xâu mật mã

Từ các ký tự của một xâu cho trước, cài đặt hàm có tên cipher để mã hoá xâu như sau:

- Nếu là ký tự tiếng Anh ở dạng thường (lower-case characters) thì chuyển thành ký tự có mã là (219 – mã ký tự).
- Các ký tự khác giữ nguyên.

Sử dụng hàm đã viết để mã hoá và giải mã các xâu ký tự tiếng Anh.

09. Typoglycemia

Cho đầu vào là một câu tiếng Anh bao gồm các word ngăn cách nhau bằng ký tự space. Viết chương trình thực hiện việc sau:

- Với mỗi word, giữ nguyên ký tự đầu và ký tự cuối, đảo thứ tự một cách ngẫu nhiên các ký tự còn lại của (tất nhiên các word có ít hơn 4 ký tự thì không cần làm gì)
- Cho trước một câu tiếng Anh hợp lệ, ví dụ "I couldn't believe that I could actually understand what I was reading : the phenomenal power of the human mind .", chạy chương trình đã viết để đưa ra kết quả.

10. A Markov chain

Upon reading an input text file, construct a dictionary that represents a Markov chain. We will set the number of previous words we observe in this chain to ***two***. Thus, for every word, we record as a key the previous two words in the document, and the value as the next word.

Example:
The quick brown fox jumps over the lazy dog.
markovChain['The quick'] = ['brown']
markovChain['quick brown'] = ['fox']

…
The created dictionary must contain a ***list*** of words for each value. In our program, most dictionary values will only have a single two-word key, but since it is possible that the same key might occur for two different values, we must maintain a list. (For example, in "to be or not to be that is the question" the key 'tobe' has two possible next words: ['or', 'that'])
To create a new text document, we begin by selecting the first two words of the original text file as the first two words of our new document. These first two words are our initial key. The value associated with that key will be the third word in the new text document. If there is more than one word in the dictionary value, one of the words is chosen randomly. A new key is formed from the second and third words, and its value added as the fourth word of the document. The newly chosen word is added to the new text document, the key updated and so on. The algorithm cycles by randomly picking a new word from the dictionary value based on the two-word key, adding the value of the dictionary to the text document, and updating the key.
Replacement of a key word is always in the following form.

Key = Word1 +	'	' + Word2	# put back space that split() removed
Word3 = markovChain[Key] # randomly chosen word in this list			
Key = Word2 +	'	' + Word3	

If a key does not have an entry in the dictionary, the program should end text generation. Of course, based on the input file, we could enter an infinite loop of text generation, so limit output to 500 words.
Once the new text has been generated, display the results, and prompt the user for the name of an output file. The user may choose to discard the text by not entering a file name, at which point the program should end.

Example Output

Enter the text file name: TreasureIsland.txt
The results:

-

From the side of the others, we should want you to help work the vessel home." "Ah," said he, "but I had—remarkable pious. And I was not defenseless, courage glowed again in my heart, and I know it. But where was you, do you call yourself, mate?" "Jim," I told him the whole story of our voyage and the

ringleader, too." He was concealed by this time, behind another tree-trunk, but he must have shown the feeling in my face, for he repeated the statement hotly:
"Rich! rich! I says. And I'll tell you, and no more. I were in Flint's ship when he buried the treasure; he and six along—six strong seamen. They was ashore nigh on a week, and us standing off and on in the most liberal of men. "Ay, but you see," returned Ben Gunn, I am; and I saw a figure leap with great rapidity behind the trunk of a fellow-creature. But at my last words he perked up into a kind of punishment common enough among the buccaneers, in which the offender is put ashore with a little powder and shot and left behind on some desolate and distant island. "Marooned three years agone," he continued, "then you'll up, and you'll say
...

-

Enter file name to write output to <Enter for skip>: lol.txt
Your solution will do the following:

1) Prompt for the file name containing the text to work on.
2) Create a function that takes as input, the file name of the text, and returns a list containing all the words in the file. Only newline characters are to be removed from the text. Keep all punctuation (they will likely be attached to words).
3) Create a function that takes as input, a list containing all words in the text file, and returns a dictionary formatted as specified above.