BÀI TẬP

Cấu trúc dữ liệu cơ bản

Mục tiêu: Làm việc với file và xử lý ngoại lệ.

Bài tập thực hành

Bài 1:

Viết chương trình mô tả một số lệnh tạo tệp và thư mục như sau:

Chương trình nhập vào các lệnh từ bàn phím,

1. Lệnh **mkdir** tạo thư mục
2. Lệnh **mkfile** tạo tệp
3. Lệnh **ls** hiển thị các tệp, thư mục hiện có trong đường dẫn
4. Lệnh **cd** chuyển đường dẫn đến thư mục khác

Ví dụ thư mục hiện tại là /home/student

>> mkdir demo

Sẽ tạo thư mục demo trong /home/student

>>mkfile demo/test.txt

Sẽ tạo file test trong /home/student/demo

>>cd demo

Sẽ chuyển đường dẫn đến /home/student/demo

>> ls

Hiển thị các tệp, thư mục trong thư mục hiện tại

Gợi ý: dùng các hàm trong module os.

Bài 2:

Viết chương trình duyệt, và in ra cây thư mục của 1 đường dẫn.

Bài 3:

Viết chương trình đọc dữ liệu từ tệp input.txt, in dữ liệu đọc được theo thứ tự đảo ngược ra file reverse.txt

Ví dụ input.txt có các dòng

123

456

789

Thì file reverse.txt có các dòng là

987

654

321

Bài 4:

Viết chương trình tạo một từ điển gồm khóa là xâu độ dài 5 được sinh ngẫu nhiên, và giá trị của mỗi khóa là số nguyên cũng được sinh ngẫu nhiên, ghi từ điển ra file dict.dat (ghi đối tượng), đọc đối tượng từ điển từ file dict.dat và in kết quả ra màn hình

Gợi ý: sử dụng module pickle.

Bài 5:

Viết chương trình yêu cầu nhập vào một số nguyên, xử lý ngoại lệ nếu người dùng nhập không đúng định dạng yêu cầu nhập lại, chương trình kết thúc khi người dùng nhập đúng yêu cầu.

Bài 6:

Viết chương trình so sánh nội dung 2 file, nội dung 2 file được coi là giống nhau nếu mỗi dòng trong file này đều có trong file kia và ngược lại.

Bài 7:

Viết chương trình vector hóa các file text. Kho văn bản gồm nhiều file text, chương trình sẽ lọc ra 100 từ xuất hiện nhiều nhất trong kho văn bản làm từ điển, khi đó mỗi file trong kho văn bản có thể được biểu diễn bằng 1 vector 100 chiều.

Ví dụ: từ điển gồm 10 từ:

tôi, làm, anh, đi, ngủ, ăn, học, chơi, tiền, mưa

File a.txt có nội dung như sau:

tôi rất thích **tiền** vì vậy **tôi** cố gắng **học** thật tốt để có thể kiếm được một công việc tốt và sẽ có nhiều **tiền** để vui chơi, ăn **chơi** không còn sợ **mưa** rơi.

Sẽ có vector tương ứng là:

2 0 0 0 0 0 1 1 2 1

Cho kho văn bản trong thư mục **doc**, hãy in ra vector của từng file văn bản trong thư mục **doc** theo mô tả trên.

Bài 8:

Cho kho văn bản doc, xây dựng chương trình in ra các 2-gram trong kho văn bản.

2-gram là các chuỗi gồm 2 tiếng trích từ các câu trong văn bản. Yêu cầu bài toán là in ra các 2-gram và số lượng của mỗi 2 gram có trong kho văn bản.

Ví dụ với xâu: "tôi thích học tôi thích chơi" sẽ có các 2-gram là:

tôi thích: 2

thích học: 1

học tôi: 1

thích chơi: 1

Bài 9:

Cho kho văn bản **doc.**

Viết chương trình tính chỉ số TFI-ID cho mỗi từ trong văn bản. Công thức tính TF-IDF cho mỗi từ trong văn bản như sau:

TF- term frequency – tần số xuất hiện của 1 từ trong 1 văn bản. Cách tính:

$$tf(t,d) = \frac{f(t,d)}{\max\{f(w,d): w \in d\}}$$

- Thương của số lần xuất hiện 1 từ trong văn bản và số lần xuất hiện nhiều nhất của một từ bất kỳ trong văn bản đó. (giá trị sẽ thuộc khoảng [0, 1])
- **f(t,d)** - số lần xuất hiện từ t trong văn bản **d**.
- **max{f(w,d):w∈d}** - số lần xuất hiện nhiều nhất của một từ bất kỳ trong văn bản.

IDF – *inverse document frequency*. Tần số nghịch của 1 từ trong tập văn bản (corpus).

Tính **IDF** để giảm giá trị của những từ phổ biến. Mỗi từ chỉ có 1 giá trị **IDF** duy nhất trong tập văn bản.

$$idf(t, D) = \log\left(\frac{|D|}{|\{d \in D: t \in d\}|}\right)$$

- |D|: - tổng số văn bản trong tập **D** (số văn bản trong kho văn bản)
- $|\{d \in D: t \in d\}|$: - số văn bản chứa từ nhất định, với điều kiện từ **t** xuất hiện trong văn bản **d** (i.e.,). Nếu từ đó không xuất hiện ở bất cứ 1 văn bản nào trong tập thì mẫu số sẽ bằng 0 => phép chia cho không không hợp lệ, vì thế người ta thường thay bằng mẫu thức 1+$|\{d \in D: t \in d\}|$:.

Cơ số logarit trong công thức này không thay đổi giá trị của 1 từ mà chỉ thu hẹp khoảng giá trị của từ đó. Vì thay đổi cơ số sẽ dẫn đến việc giá trị của các từ thay đổi bởi một số nhất định và tỷ lệ giữa các trọng lượng với nhau sẽ không thay đổi. (nói cách khác, thay đổi cơ số sẽ không ảnh hưởng đến tỷ lệ giữa các giá trị IDF). Tuy nhiên việc thay đổi khoảng giá trị sẽ giúp tỷ lệ giữa IDF và TF tương đồng để dùng cho công thức TF-IDF như bên dưới.

Giá trị **TF-IDF**:

$$tfidf(t, d, D) = tf(t, d) \times idf(t, D)$$

Những từ có giá trị TF-IDF cao là những từ xuất hiện nhiều trong văn bản này, và xuất hiện ít trong các văn bản khác. Việc này giúp lọc ra những từ phổ biến và giữ lại những từ có giá trị cao (từ khoá của văn bản đó).

Bài 10:

Đọc và hiển thị dữ liệu theo từng dòng từ file input.txt.

Tại mỗi thời điểm, chương trình hiển thị 1 dòng trong file văn bản, Chương trình nhận phím điều khiển lên (up) để hiển thị dòng phía trên dòng hiện tại và xuống (down) để hiển thị dữ liệu tiếp theo sau dòng hiện tại.

Code mẫu bài 10.

```
import os
import keyboard
# on window platform
clear = lambda: os.system('cls')

#on linux platform
#clear = lambda: os.system('clear')
f = open('demofile.txt', encoding = 'utf-8')

content = f.readlines()
current = 0

def printLineUp():
    clear()
    global current
    current -=1
    if current <= -(len(content)):
        current = 0
    print(content[current])


def printLineDown():
    clear()
    global current
    current +=1
    if(current >= len(content)):
        current =0
    print(content[current%len(content)])



keyboard.add_hotkey('up', printLineUp)
keyboard.add_hotkey('down', printLineDown)
keyboard.wait('esc')
```