


TRƯỜNG: ĐH Công thương TP HCM		
KHOA: CÔNG NGHỆ THÔNG TIN	BUỔI 6. LÀM VIỆC VỚI FILE & XỬ LÝ LỖI	
BỘ MÔN: KHD&TTNT		
MH: LẬP TRÌNH PYTHON		

A. MỤC TIÊU

- Làm việc với file trong Python
- Xử lý lỗi trong python

B. DỤNG CỤ - THIẾT BỊ THÍ NGHIỆM CHO MỘT SV

STT	Chủng loại – Quy cách vật tư	Số lượng	Đơn vị	Ghi chú
1	Computer	1	1	

C. NỘI DUNG THỰC HÀNH

I. Tóm tắt lý thuyết

1. Đọc và ghi file text

a. Sử dụng hàm `open()`

Trong Python, để làm việc với file text, chúng ta sử dụng hàm `open()` để mở một file và tạo một đối tượng file.

Hàm `open()`

- Hàm `open()` được sử dụng để mở một file trong Python.
- Cú pháp: `open(file, mode='r', buffering=-1, encoding=None, errors=None, newline=None, closefd=True, opener=None)`

Trong đó:

- `file`: Tên của file cần mở.
- `mode`: Chế độ mở file. Mặc định là 'r' (đọc). Các chế độ khác bao gồm 'w' (ghi), 'a' (nối), 'rb', 'wb', v.v.
- `buffering`: Kiểu buffering (đệm). Mặc định là -1 (điều chỉnh theo hệ thống).
- `encoding`: Mã hóa của file.
- `errors`: Xử lý lỗi khi giải mã hoặc mã hóa file.

- `newline`: Xác định ký tự dùng để phân biệt dòng (`\n`, `\r\n`, v.v.).
- `closefd`: Xác định liệu file descriptor sẽ đóng khi file bị đóng hay không. Mặc định là `True`.
- `opener`: Một hàm tạo file mới.

Ví dụ

```
# Mở file để đọc
with open('example.txt', 'r') as file:
    content = file.read()
    print(content)

# Mở file để ghi (xóa toàn bộ file khi mở)
with open('example.txt', 'w') as file:
    file.write('Hello, world!')

# Mở file để ghi thêm vào file
with open('example.txt', 'a') as file:
    file.write('\nThis is a new line.')

# Mở file ở chế độ binary
with open('example.txt', 'rb') as file:
    content = file.read()
    print(content)
```

Chú ý

- Khi sử dụng `open()`, luôn sử dụng với câu lệnh **with** để đảm bảo rằng file sẽ được đóng tự động sau khi hoàn thành thao tác.
- Nên xác định chế độ mở file một cách cẩn thận để tránh ghi đè hoặc mất dữ liệu không mong muốn.

b. Phương pháp đọc file

Trong Python, có ba phương pháp chính để đọc dữ liệu từ file text:

- `read()`: Phương thức này sẽ đọc toàn bộ nội dung của file và trả về một chuỗi (string) chứa dữ liệu đó.
- `readline()`: Phương thức này sẽ đọc dòng đầu tiên trong file và trả về nó dưới dạng chuỗi. Mỗi lần gọi `readline()` tiếp theo sẽ đọc dòng tiếp theo trong file.

- `readlines()`: Phương thức này sẽ đọc toàn bộ nội dung của file và trả về một danh sách (list) các chuỗi, mỗi chuỗi đại diện cho một dòng trong file.

Ví dụ

```
with open('myfile.txt', 'r') as file_object:
    # Đọc toàn bộ nội dung của file bằng phương thức read()
    content = file_object.read()
    print(content)

with open('myfile.txt', 'r') as file_object:
    # Đọc dòng đầu tiên của file bằng phương thức readline()
    first_line = file_object.readline()
    print(first_line)

    # Đọc dòng tiếp theo của file
    second_line = file_object.readline()
    print(second_line)

with open('myfile.txt', 'r') as file_object:
    # Đọc toàn bộ file và lưu vào danh sách các dòng bằng phương
    # thức readlines()
    lines = file_object.readlines()
    print(lines)
```

c. Phương pháp ghi file

Khi đã mở một file text để ghi bằng hàm `open()` với chế độ 'w' hoặc 'a', chúng ta có thể sử dụng các phương thức sau để ghi dữ liệu vào file:

- `write()`: Phương thức này cho phép ghi một chuỗi (string) vào file. Nếu file đã tồn tại, nội dung cũ sẽ bị ghi đè bởi nội dung mới. Nếu file chưa tồn tại, một file mới sẽ được tạo ra.
- `writelines()`: Phương thức này cho phép ghi một danh sách (list) các chuỗi vào file. Mỗi chuỗi trong danh sách sẽ được ghi là một dòng trong file.

Dưới đây là một ví dụ minh họa:

```

# Mở hoặc tạo file để ghi, chế độ 'w' sẽ ghi đè lên nếu file đã
tồn tại
with open('myfile.txt', 'w') as file_object:
    file_object.write("Hello, world!\n") # Ghi một chuỗi vào
file
    file_object.write("This is a test.\n") # Ghi một chuỗi khác
vào file

# Mở file để ghi tiếp, chế độ 'a' sẽ ghi thêm vào cuối file nếu
file đã tồn tại
with open('myfile.txt', 'a') as file_object:
    file_object.write("Appending new line.\n") # Ghi thêm một
chuỗi vào cuối file

# Ghi danh sách các chuỗi vào file sử dụng phương thức
writelines()
lines_to_write = ["Line 1\n", "Line 2\n", "Line 3\n"]
with open('myfile.txt', 'a') as file_object:
    file_object.writelines(lines_to_write)

# Ghi danh sách các chuỗi vào file với ký tự ngăn cách giữa các
dòng
lines_to_write = ["Line 4", "Line 5", "Line 6"]
with open('myfile.txt', 'a') as file_object:
    file_object.write("\n".join(lines_to_write))

```

2. Xử lý file CSV (Comma Separated Values)

a. Giới thiệu về file CSV

File CSV (Comma Separated Values) là một định dạng lưu trữ dữ liệu đơn giản và phổ biến được sử dụng để trao đổi dữ liệu giữa các ứng dụng và các hệ thống khác nhau. Trong file CSV, dữ liệu được tổ chức dưới dạng các hàng và cột, trong đó mỗi hàng đại diện cho một bản ghi và các giá trị trong hàng được phân tách bằng dấu phẩy (,).

Dưới đây là một số đặc điểm quan trọng của file CSV:

- Dễ đọc và hiểu: Với định dạng đơn giản, file CSV rất dễ đọc và hiểu, cả cho con người và cho các chương trình máy tính.

- Hỗ trợ nhiều loại dữ liệu: File CSV có thể lưu trữ nhiều loại dữ liệu, bao gồm số, chuỗi ký tự, ngày tháng, và các kiểu dữ liệu khác.
- Tương thích với nhiều ứng dụng và ngôn ngữ lập trình: File CSV được hỗ trợ rộng rãi và có thể được sử dụng trong nhiều loại ứng dụng và ngôn ngữ lập trình khác nhau như Python, Excel, MySQL, và nhiều ứng dụng khác.
- Dung lượng nhỏ gọn: Do định dạng đơn giản và việc sử dụng ký tự phân tách dữ liệu là dấu phẩy, file CSV thường có dung lượng nhỏ gọn, giúp tiết kiệm không gian lưu trữ.
- Dễ dàng xử lý và thao tác: Các thao tác như đọc, ghi, và xử lý dữ liệu trong file CSV thường rất dễ dàng và đơn giản.

File CSV thường được sử dụng để lưu trữ dữ liệu từ các bảng dữ liệu trong cơ sở dữ liệu, kết quả của các báo cáo, hoặc dữ liệu thu thập từ các tập tin log. Trong Python, thư viện csv cung cấp các công cụ mạnh mẽ để làm việc với file CSV, bao gồm đọc, ghi, và xử lý dữ liệu.

b. Sử dụng thư viện csv của Python

Để làm việc với file CSV trong Python, chúng ta có thể sử dụng thư viện csv tích hợp sẵn trong Python. Thư viện này cung cấp các công cụ tiện ích để đọc và ghi dữ liệu từ và vào file CSV một cách dễ dàng và hiệu quả.

Đọc dữ liệu từ file CSV

```
import csv

# Mở file CSV để đọc
with open('file.csv', 'r', encoding='utf-8') as file:
    # Tạo đối tượng reader để đọc file CSV
    reader = csv.reader(file)

    # Đọc dữ liệu từ file và in ra màn hình
    for row in reader:
        print(row)
```

Ghi dữ liệu vào file CSV

```
import csv

# Dữ liệu cần ghi vào file CSV
```

```

data = [
    ['Tên', 'Tuổi', 'Địa chỉ'],
    ['John', '25', 'New York'],
    ['Alice', '30', 'San Francisco'],
    ['Bob', '28', 'Los Angeles']
]

# Mở file CSV để ghi
with open('output.csv', 'w', newline='', encoding='utf-8') as file:
    # Tạo đối tượng writer để ghi vào file CSV
    writer = csv.writer(file)

    # Ghi dữ liệu vào file CSV
    writer.writerows(data)

```

Sử dụng DictReader và DictWriter (đọc và ghi dữ liệu dưới dạng từ điển)

```

import csv

# Đọc dữ liệu từ file CSV và chuyển đổi thành từ điển
with open('file.csv', 'r') as file:
    # Tạo đối tượng DictReader để đọc file CSV
    reader = csv.DictReader(file)

    # In dữ liệu từ file CSV dưới dạng từ điển
    for row in reader:
        print(row)

# Ghi dữ liệu từ từ điển vào file CSV
data = [
    {'Tên': 'John', 'Tuổi': '25', 'Địa chỉ': 'New York'},
    {'Tên': 'Alice', 'Tuổi': '30', 'Địa chỉ': 'San Francisco'},
    {'Tên': 'Bob', 'Tuổi': '28', 'Địa chỉ': 'Los Angeles'}
]

# Mở file CSV để ghi
with open('output.csv', 'w', newline='', encoding='utf-8') as file:
    # Tạo đối tượng DictWriter để ghi vào file CSV

```

```
fieldnames = ['Tên', 'Tuổi', 'Địa chỉ']  
writer = csv.DictWriter(file, fieldnames=fieldnames)  
  
# Ghi dữ liệu từ từ điển vào file CSV  
writer.writeheader() # Viết header  
writer.writerows(data)
```

3. Xử lý file JSON (JavaScript Object Notation)

a. Giới thiệu về file JSON

File JSON (JavaScript Object Notation) là một định dạng lưu trữ dữ liệu phổ biến được sử dụng trong truyền tải và lưu trữ dữ liệu cấu trúc. JSON là một định dạng đơn giản, dễ đọc và dễ hiểu, đặc biệt phù hợp cho việc trao đổi dữ liệu giữa các ứng dụng web hoặc giữa máy chủ và máy khách.

Một số đặc điểm quan trọng của file JSON:

- Đơn giản và dễ đọc: JSON được viết dưới dạng văn bản và có cấu trúc gần giống với các đối tượng và mảng trong JavaScript. Do đó, nó rất dễ đọc và hiểu cho con người.
- Cấu trúc phân cấp: JSON hỗ trợ cấu trúc phân cấp, có nghĩa là bạn có thể lồng các đối tượng (object) và mảng (array) trong nhau để tạo ra các dữ liệu phức tạp và có cấu trúc.
- Hỗ trợ nhiều kiểu dữ liệu: JSON hỗ trợ nhiều loại dữ liệu, bao gồm số, chuỗi ký tự, đối tượng, mảng, boolean và null.
- Tương thích đa nền tảng: JSON là một định dạng độc lập với nền tảng, nghĩa là nó có thể được sử dụng trên nhiều loại ứng dụng và hệ điều hành khác nhau.
- Hỗ trợ trong lập trình: JSON được hỗ trợ trong nhiều ngôn ngữ lập trình, không chỉ là JavaScript mà còn trong Python, Java, PHP và nhiều ngôn ngữ khác.

File JSON thường được sử dụng để lưu trữ cấu trúc dữ liệu phức tạp như cài đặt của ứng dụng, cấu trúc dữ liệu của trang web, cấu trúc dữ liệu từ API, và nhiều hơn nữa. Trong Python, thư viện json cung cấp các công cụ để làm việc với file JSON, bao gồm đọc, ghi và xử lý dữ liệu JSON. Thư viện này giúp cho việc làm việc với dữ liệu JSON trở nên dễ dàng và hiệu quả.

b. Sử dụng thư viện json của Python

Để làm việc với dữ liệu JSON trong Python, chúng ta có thể sử dụng thư viện tích hợp sẵn json. Thư viện này cung cấp các phương pháp để đọc dữ liệu từ một chuỗi JSON và chuyển

đổi nó thành các đối tượng Python, cũng như để chuyển đổi các đối tượng Python thành chuỗi JSON và ghi vào file.

Đọc dữ liệu từ chuỗi JSON

```
import json

# Chuỗi JSON
json_string = '{"name": "John", "age": 30, "city": "New York"}'

# Chuyển đổi chuỗi JSON thành đối tượng Python (dictionary)
data = json.loads(json_string)

# In dữ liệu
print(data)
```

Ghi dữ liệu vào file JSON

```
import json

# Dữ liệu cần ghi vào file JSON
data = {
    "name": "John",
    "age": 30,
    "city": "New York"
}

# Ghi dữ liệu vào file JSON
with open('output.json', 'w') as file:
    json.dump(data, file)
```

Đọc dữ liệu từ file JSON

```
import json

# Mở file JSON để đọc
with open('input.json', 'r') as file:
    # Đọc dữ liệu từ file và chuyển đổi thành đối tượng Python
    (dictionary)
    data = json.load(file)

# In dữ liệu
```



```
print(data)
```

Chuyển đổi đối tượng Python thành chuỗi JSON

```
import json

# Dữ liệu Python (dictionary)
data = {
    "name": "John",
    "age": 30,
    "city": "New York"
}

# Chuyển đổi đối tượng Python thành chuỗi JSON
json_string = json.dumps(data)

# In chuỗi JSON
print(json_string)
```

4. Xử lý lỗi với Python

a. Ý nghĩa

Xử lý lỗi là một khía cạnh quan trọng trong lập trình, với ý nghĩa và tầm quan trọng như sau:

Ý nghĩa của xử lý lỗi:

- Bảo vệ ứng dụng: Xử lý lỗi giúp bảo vệ ứng dụng của bạn trước những tình huống không mong muốn, như nhập liệu không hợp lệ hoặc lỗi trong quá trình thực thi chương trình.
- Cải thiện trải nghiệm người dùng: Xử lý lỗi một cách tự nhiên và thông minh giúp cải thiện trải nghiệm người dùng bằng cách cung cấp thông báo lỗi thân thiện và hữu ích.
- Giúp gỡ lỗi: Xử lý lỗi cung cấp thông tin cần thiết để gỡ lỗi và sửa chữa lỗi, giúp các lập trình viên dễ dàng tìm ra nguyên nhân của sự cố và khắc phục nó.
- Đảm bảo tính ổn định của ứng dụng: Xử lý lỗi giúp đảm bảo tính ổn định của ứng dụng bằng cách ngăn chặn việc xuất hiện các tình huống không xác định có thể làm cho ứng dụng gặp sự cố hoặc crash.

Tầm quan trọng của xử lý lỗi:

- Bảo mật: Xử lý lỗi giúp bảo vệ ứng dụng khỏi các cuộc tấn công bảo mật, như các lỗ hổng nhập liệu hoặc tấn công dựa trên lỗi.
- Chất lượng phần mềm: Xử lý lỗi là một phần quan trọng của quy trình kiểm thử phần mềm, giúp đảm bảo chất lượng và tính ổn định của ứng dụng.
- Tăng cường tin cậy và uy tín: Việc xử lý lỗi một cách chuyên nghiệp giúp tăng cường tin cậy và uy tín của ứng dụng trong mắt người dùng và các bên liên quan.
- Tiết kiệm thời gian và chi phí: Xử lý lỗi đúng cách từ ban đầu giúp tiết kiệm thời gian và chi phí so với việc phải sửa lỗi sau khi ứng dụng đã triển khai.

Tóm lại, xử lý lỗi không chỉ là một phần quan trọng của quy trình phát triển phần mềm mà còn là một yếu tố không thể thiếu trong việc đảm bảo tính ổn định, an toàn và hiệu quả của ứng dụng.

b. Cách xử lý lỗi trong Python

Trong Python, bạn có thể sử dụng câu lệnh **try**, **except**, **else**, và **finally** để xử lý lỗi. Dưới đây là cú pháp và mô tả cách xử lý lỗi trong Python:

```
try:
    # Mã lệnh có thể gây ra lỗi
    # Ví dụ: chia cho 0, mở một file không tồn tại, gọi một hàm
    không tồn tại, v.v.
except ExceptionType1:
    # Xử lý lỗi loại ExceptionType1
except ExceptionType2:
    # Xử lý lỗi loại ExceptionType2
else:
    # Mã lệnh chạy khi không có lỗi xảy ra
finally:
    # Mã lệnh chạy sau cùng, bất kể có lỗi hay không
```

Mô tả:

- try: Khối lệnh được bao bọc trong try là nơi bạn đặt mã lệnh mà bạn nghi ngờ có thể gây ra lỗi.
- except: Bạn có thể sử dụng một hoặc nhiều khối except để xử lý các loại lỗi cụ thể. Bất kỳ lỗi nào xảy ra trong khối try và phù hợp với loại được xác định trong except sẽ được xử lý ở đó.

- **else:** Khối lệnh trong **else** sẽ được thực thi nếu không có lỗi nào xảy ra trong khối **try**. Điều này thường được sử dụng để chạy mã lệnh phụ thuộc vào việc có lỗi hay không.
- **finally:** Mã lệnh trong khối **finally** sẽ được thực thi bất kể có lỗi xảy ra hay không. Thường được sử dụng để giải phóng tài nguyên, như đóng file hoặc đóng kết nối cơ sở dữ liệu.

Ví dụ

```
try:
    result = 10 / 0 # Chia cho 0 - gây ra ZeroDivisionError
except ZeroDivisionError:
    print("Không thể chia cho 0!")
except Exception as e: # Bắt tất cả các loại lỗi khác
    print("Lỗi:", e)
else:
    print("Không có lỗi nào xảy ra.")
finally:
    print("Kết thúc xử lý lỗi.")
```

c. Một số loại lỗi trong python

Trong Python, có nhiều loại lỗi khác nhau có thể xảy ra trong quá trình thực thi chương trình. Dưới đây là một số loại lỗi phổ biến:

- **SyntaxError:** Xảy ra khi có lỗi cú pháp trong mã Python, chẳng hạn như việc sử dụng từ khóa không đúng cách hoặc quên ký tự đóng ngoặc.
- **IndentationError:** Xảy ra khi có lỗi về cách thụt lề (indentation) trong mã Python, chẳng hạn như việc sử dụng các cấp thụt lề không đồng nhất.
- **NameError:** Xảy ra khi một biến hoặc tên không được định nghĩa trong phạm vi hiện tại của chương trình.
- **TypeError:** Xảy ra khi một phép toán hoặc hàm được gọi với một loại dữ liệu không phù hợp, chẳng hạn như cố gắng cộng hai chuỗi và một số.
- **ZeroDivisionError:** Xảy ra khi một phép chia được thực hiện với mẫu số bằng 0.
- **IndexError:** Xảy ra khi bạn truy cập vào một phần tử trong một danh sách hoặc chuỗi với một chỉ mục không hợp lệ.
- **KeyError:** Xảy ra khi bạn truy cập vào một khóa không tồn tại trong một từ điển.
- **FileNotFoundError:** Xảy ra khi một file không được tìm thấy trong hệ thống tệp.

- **ValueError:** Xảy ra khi một hàm được gọi với một đối số có kiểu đúng nhưng giá trị không hợp lệ.
- **AttributeError:** Xảy ra khi một đối tượng không có thuộc tính hoặc phương thức mà bạn cố gắng truy cập.

II. Bài tập hướng dẫn mẫu

Bài 1. Đọc dữ liệu từ file và tính trung bình.

- Tạo một file văn bản chứa một số nguyên trên mỗi dòng.
- Tính trung bình mỗi dòng và ghi vào file kết quả.
- Xử lý lỗi: Bắt các ngoại lệ có thể xảy ra khi đọc file và dòng trống.

```
def calculate_average(numbers):  
    total = sum(numbers)  
    return total / len(numbers)  
  
try:  
    # Mở file để đọc  
    with open('input.txt', 'r') as file:  
        lines = file.readlines()  
  
    # Kiểm tra xem file có dòng nào không  
    if len(lines) == 0:  
        raise ValueError("File không có dữ liệu")  
  
    # Tạo file để ghi kết quả  
    with open('output.txt', 'w') as output_file:  
        for line in lines:  
            # Chuyển đổi các chuỗi thành số nguyên  
            numbers = [int(num) for num in  
line.strip().split()]  
  
            # Tính trung bình  
            avg = calculate_average(numbers)  
  
            # Ghi kết quả vào file  
            output_file.write(f"{avg}\n")
```

```
except FileNotFoundError:
    print("Lỗi: File không tồn tại.")
except ValueError as ve:
    print(f"Lỗi: {ve}")
except Exception as e:
    print(f"Lỗi không xác định: {e}")
```

Bài 2. Viết chương trình đọc 2 dãy số nguyên từ 2 dòng của 1 file input.txt. Trộn 2 dãy và sắp xếp theo thứ tự tăng dần rồi ghi vào file output.txt. Xử lý lỗi cần thiết (Không có file, không phải số nguyên)

```
def read_integers(line):
    """
    Chuyển đổi chuỗi thành một danh sách các số nguyên.
    """
    try:
        integers = [int(num) for num in line.strip().split()]
        return integers
    except ValueError:
        raise ValueError("Dòng chứa ký tự không phải là số nguyên")

def merge_and_sort(integers1, integers2):
    """
    Trộn và sắp xếp hai danh sách số nguyên theo thứ tự tăng dần.
    """
    merged_list = integers1 + integers2
    merged_list.sort()
    return merged_list

try:
    # Mở file để đọc
    with open('input.txt', 'r') as file:
        lines = file.readlines()

    # Kiểm tra xem file có ít nhất 2 dòng không
    if len(lines) < 2:
        raise ValueError("File không có đủ dữ liệu")
```

```

# Đọc dãy số nguyên từ hai dòng
integers1 = read_integers(lines[0])
integers2 = read_integers(lines[1])

# Trộn và sắp xếp dãy số nguyên
sorted_integers = merge_and_sort(integers1, integers2)

# Ghi kết quả vào file output.txt
with open('output.txt', 'w') as output_file:
    output_file.write(' '.join(map(str,
sorted_integers)))

except FileNotFoundError:
    print("Lỗi: File không tồn tại.")
except ValueError as ve:
    print(f"Lỗi: {ve}")
except Exception as e:
    print(f"Lỗi không xác định: {e}")

```

Bài 3. Đọc file json là một mảng nhiều phần tử, mỗi phần tử bao gồm chiều rộng (width) và chiều cao (height) của một hình chữ nhật. Viết chương trình đọc file này và in ra màn hình hình chữ nhật có diện tích lớn nhất.

```

import json

def find_max_rectangle(rectangles):
    """
    Tìm hình chữ nhật có diện tích lớn nhất từ một danh sách các
    hình chữ nhật.
    """
    max_area_rectangle = None
    max_area = 0

    for rectangle in rectangles:
        width = rectangle.get('width', 0)
        height = rectangle.get('height', 0)
        area = width * height
        if area > max_area:

```

```

        max_area = area
        max_area_rectangle = rectangle

    return max_area_rectangle

try:
    # Đọc dữ liệu từ file JSON
    with open('input.json', 'r') as file:
        data = json.load(file)

    # Tìm hình chữ nhật có diện tích lớn nhất
    max_rectangle = find_max_rectangle(data)

    # In ra thông tin về hình chữ nhật có diện tích lớn nhất
    if max_rectangle:
        print("Thông tin về hình chữ nhật có diện tích lớn nhất:")
        print(f"Chiều rộng: {max_rectangle.get('width', 0)}")
        print(f"Chiều cao: {max_rectangle.get('height', 0)}")
        print(f"Diện tích: {max_rectangle.get('width', 0) * max_rectangle.get('height', 0)}")
    else:
        print("Không có hình chữ nhật nào trong dữ liệu.")

except FileNotFoundError:
    print("Lỗi: File không tồn tại.")
except json.JSONDecodeError:
    print("Lỗi: Không thể đọc dữ liệu JSON từ file.")
except Exception as e:
    print(f"Lỗi không xác định: {e}")

```

Dạng file input (input.json)

```

[
    {"width": 5, "height": 10},
    {"width": 7, "height": 3},
    {"width": 4, "height": 8},
    {"width": 6, "height": 6}
]

```

III. Bài tập ở lớp

Bài 1. Đọc dữ liệu từ file văn bản, tính tổng các số nguyên và ghi kết quả vào file văn bản.

Yêu cầu:

1. Tạo một file văn bản chứa các số nguyên trên mỗi dòng.
2. Viết một chương trình Python để đọc dữ liệu từ file, tính tổng các số nguyên và ghi kết quả vào file khác.
3. Xử lý lỗi: Bắt các ngoại lệ có thể xảy ra khi đọc file và chuyển đổi chuỗi sang số nguyên.

Bài 2. Đọc dữ liệu từ file văn bản, đảo ngược các từ trong mỗi dòng và ghi kết quả vào file văn bản.

Yêu cầu:

1. Tạo một file văn bản chứa các câu, mỗi câu nằm trên một dòng và có thể chứa nhiều từ, các từ cách nhau bởi dấu cách.
2. Viết một chương trình Python để đọc dữ liệu từ file, đảo ngược các từ trong mỗi câu và ghi kết quả vào file khác.
3. Xử lý lỗi: Bắt các ngoại lệ có thể xảy ra khi đọc file và xử lý chuỗi.

Bài 3. Đọc dữ liệu từ file văn bản, tính tổng số từ trong mỗi dòng và ghi kết quả vào file văn bản.

Yêu cầu:

1. Tạo một file văn bản chứa các câu, mỗi câu có thể chứa nhiều từ, các từ cách nhau bởi dấu cách.
2. Viết một chương trình Python để đọc dữ liệu từ file, tính tổng số từ trong mỗi câu và ghi kết quả vào file khác.
3. Xử lý lỗi: Bắt các ngoại lệ có thể xảy ra khi đọc file và xử lý chuỗi.

Bài 4. Đọc file văn bản chứa các số nguyên. Tính tổng, tích, hiệu, thương và ghi vào file csv

Yêu cầu:

1. Tạo một file văn bản chứa 2 số nguyên trên mỗi dòng.
2. Đọc file và ứng với mỗi dòng tính tổng, hiệu, tích, thương.
3. Ghi kết quả ra file csv.

Bài 5. Thực hiện lại bài trên nhưng kết quả ghi ra là file json.

Bài 6. Đọc file json gồm mảng các tam giác, gồm độ dài 3 cạnh [{‘a’: 100, ‘b’: 150, ‘c’: 125}, ...]. Hãy đọc file và xác định loại tam giác (thường, vuông, cân, đều) và chu vi, diện tích hình tam giác. Ghi kết quả vào file csv. Xử lý lỗi: file input không tồn tại, kiểu dữ liệu không chính xác, không phải tam giác, ...

IV. Bài tập về nhà

Câu 1. Xây dựng một lớp người (Person). Bao gồm các thuộc tính như tên, quốc gia và ngày sinh và các phương thức:

- Xác định tuổi của người đó.
- In ra thông tin của người đó:

Person 1:

Name: Tom

Country: Việt nam

Date of Birth: 1964-07-12

Age: 60

- Xây dựng file input.json bao gồm mảng thông tin của nhiều người. Mỗi người gồm các thuộc tính: name, country, birthday.
- Đọc file này thành một list của các đối tượng Person.
- Xây dựng chức năng: Thêm một người vào danh sách, Xóa người dựa trên tên, Xuất ra danh sách người có tuổi cho trước.
- Xử lý lỗi khi cần thiết.

Câu 2. Xây dựng một lớp Sách (Book) có các thuộc tính: name, author, date.

- Xây dựng file input.json bao gồm mảng thông tin của nhiều quyển sách.
- Đọc file thành một list các đối tượng Book.

- Xây dựng các chức năng CRUD: Thêm một quyền sách mới, Truy vấn danh sách sách theo tác giả, Chính sửa thông tin sách (Cho người dùng chọn từ danh sách quyền sách cần sửa) và Xóa một quyền sách (Cho người dùng chọn từ danh sách quyền sách cần xóa).
- Xây dựng chức năng lưu thông tin lại file json.
- Xử lý lỗi khi cần thiết.