**Programming Assignment Unit 8**

University of the People

CS 1101: Programming Fundamentals

Mahmoud Mohanna

29 March 2024

**Problem Statement**

Write a program to read dictionary items from a file and then write the inverted dictionary to a file. Ensure the program includes the following components:

1. The input file for your original dictionary (with at least six items).
2. The Python program you used to read from a file, invert the dictionary, and write to a different file. (You need to create a dictionary file and invert it into another file).
3. The output file for your inverted dictionary.
4. Provide a technical explanation for the code and its output in a minimum of 200 words.

Sample Input File (Not specific)

{

apple: red

banana: yellow

cherry: red

mango: yellow

grapes: black, green

}

Sample Output File:

{

red: apple, cherry

yellow: banana, mango

black: grapes

blue: grapes

}

**Programming Instructions:**

The code and its output must be explained technically. The explanation can be provided before or after the code, or in the form of comments within the code.

**Solution**

In this assignment I decided to use the data from the assignment for unit 7, only this time the data is taken from a text file instead of hard coded into the program itself.

The Python script facilitates the transformation of the student’s data between a file format and an inverted dictionary representation. The script comprises of three main functions: **invert\_dictionary**, **read\_students\_file**, and **write\_inverted\_dict\_to\_file**. The **read\_students\_file** function reads student information from the specified file “students.txt” , organizing it into a dictionary where student names are keys and their respective courses are values. Subsequently, the **invert\_dictionary** function converts this dictionary, turning course names into keys and students enrolled in each course into corresponding values.

Finally, the **write\_inverted\_dict\_to\_file** function writes this inverted dictionary to a designated output file. The script incorporates exception handling to manage file-related errors, ensuring robustness during file operations. Upon execution, the script processes the provided student data file, generates an inverted representation, and stores it in a new file. Additionally, it displays the original and inverted data files for inspection.

**:: Code ::**

def invert\_dictionary(original\_dict):

"""

Function to invert a dictionary where keys become values and values become keys.

Args:

original\_dict (dict): The original dictionary to be inverted.

Returns:

dict: The inverted dictionary.

"""

inverted\_dict = {}

for key, values in original\_dict.items():

for value in values:

if value in inverted\_dict:

inverted\_dict[value].append(key)

else:

inverted\_dict[value] = [key]

return inverted\_dict

def read\_students\_file(file\_name):

"""

Function to read student data from a file and return it as a dictionary.

Args:

file\_name (str): The name of the file to read from.

Returns:

dict: A dictionary containing student names as keys and their courses as values.

"""

students\_dict = {}

try:

with open(file\_name, 'r') as file:

for line in file:

parts = line.strip().split(':')

student = parts[0].strip()

courses = [course.strip() for course in parts[1].split(',')]

students\_dict[student] = courses

except FileNotFoundError:

print(f"Error: File '{file\_name}' not found.")

except Exception as e:

print(f"An error occurred while reading the file '{file\_name}': {e}")

return students\_dict

def write\_inverted\_dict\_to\_file(inverted\_dict, file\_name):

"""

Function to write an inverted dictionary to a file.

Args:

inverted\_dict (dict): The inverted dictionary to be written to the file.

file\_name (str): The name of the file to write to.

"""

try:

with open(file\_name, 'w') as file:

for key, values in inverted\_dict.items():

file.write(key + ': ' + ', '.join(values) + '\n')

except Exception as e:

print(f"An error occurred while writing to the file '{file\_name}': {e}")

# Read from students.txt

students\_dict = read\_students\_file('students.txt')

# Invert the dictionary

inverted\_dict = invert\_dictionary(students\_dict)

# Write the inverted dictionary to another file

write\_inverted\_dict\_to\_file(inverted\_dict, 'inverted\_students.txt')

# Display original and inverted files to the screen

print("Original students.txt:")

try:

with open('students.txt', 'r') as file:

print(file.read())

except FileNotFoundError:

print("Error: Original students file not found.")

print("\nInverted inverted\_students.txt:")

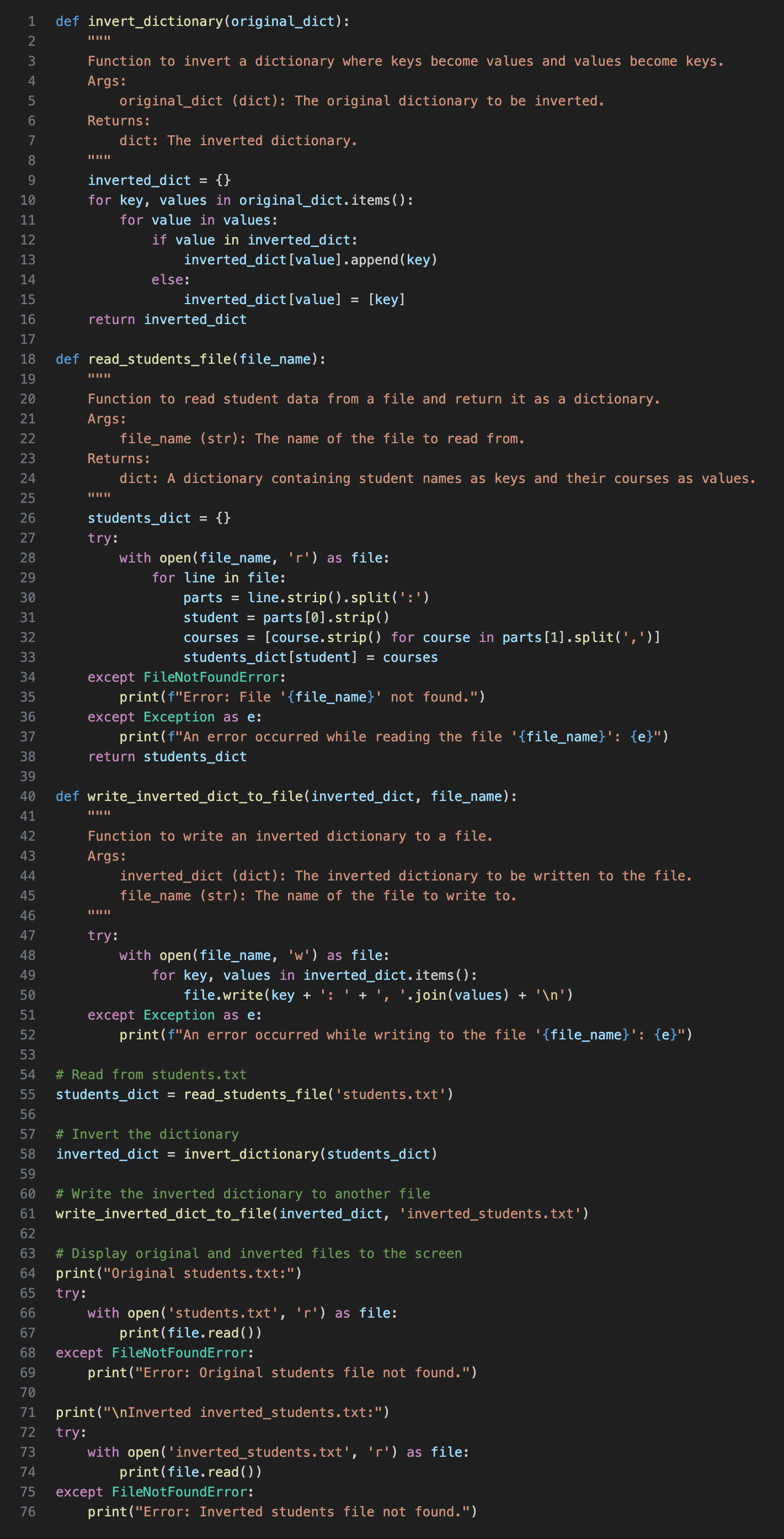
try:

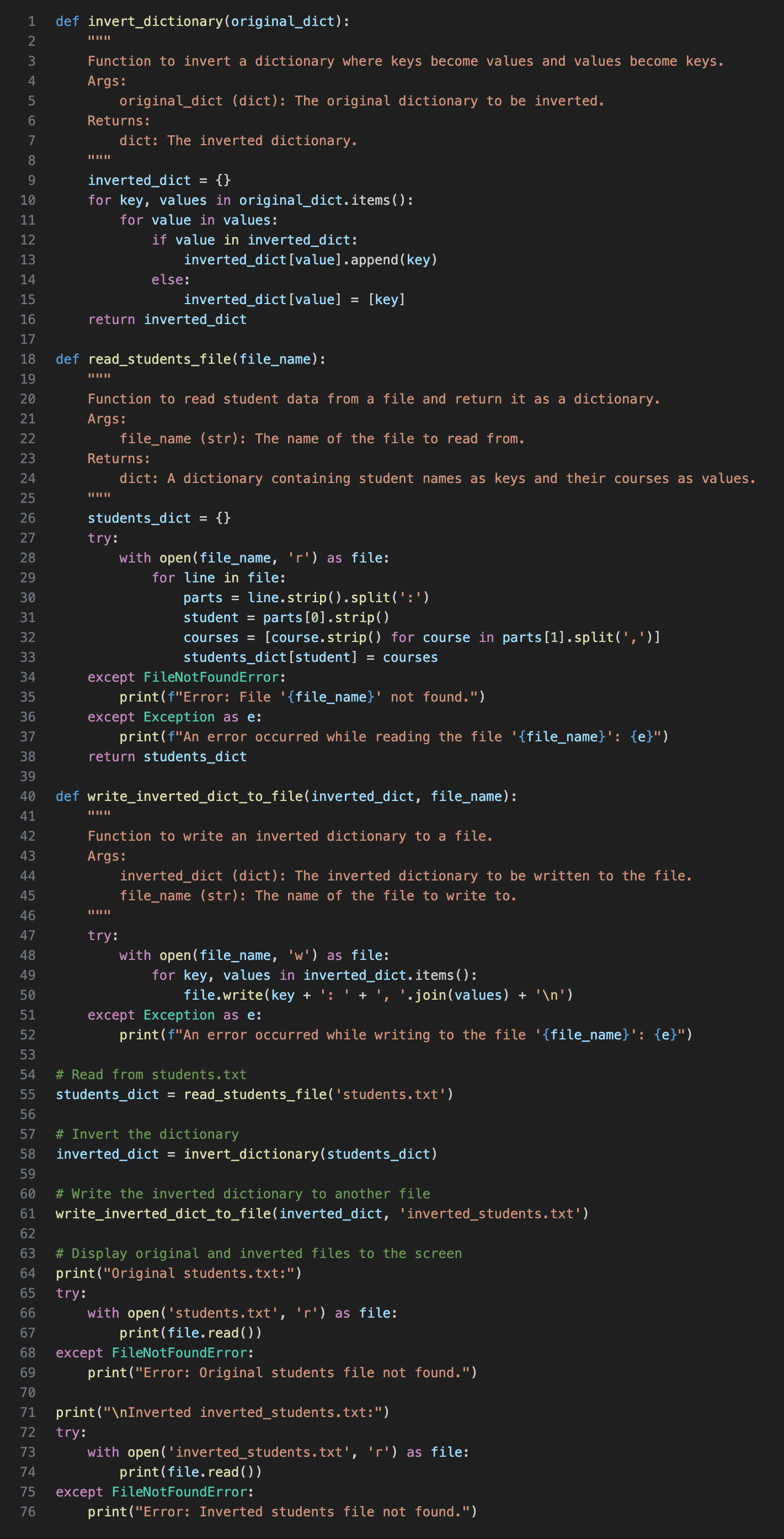
with open('inverted\_students.txt', 'r') as file:

print(file.read())

except FileNotFoundError:

print("Error: Inverted students file not found.")

**:: Snapshot of the code from VSCode ::**



**Explanation**

The code is designed to read data from a file, invert the data using dictionaries, and write the inverted data to another file.

**invert\_dictionary(original\_dict)**

def invert\_dictionary(original\_dict):

inverted\_dict = {}

for key, values in original\_dict.items():

for value in values:

if value in inverted\_dict:

inverted\_dict[value].append(key)

else:

inverted\_dict[value] = [key]

return inverted\_dict

First we define a function called `invert\_dictionary` that takes in a dictionary `original\_dict` as an argument. The purpose of this function is to invert the dictionary, where the keys of the original dictionary become the values in the inverted dictionary, and the values of the original dictionary become the keys in the inverted dictionary. The function iterates over the items of the original dictionary and creates a new dictionary called `inverted\_dict`. It checks if each value is already a key in `inverted\_dict`. If it is, it appends the corresponding key from the original dictionary to the value in `inverted\_dict`. If the value is not a key in `inverted\_dict`, it adds the value as a key in `inverted\_dict` and assigns a list with the corresponding key from the original dictionary as the value. Finally, the inverted dictionary `inverted\_dict` is returned.

**read\_students\_file(file\_name)**

def read\_students\_file(file\_name):

students\_dict = {}

try:

with open(file\_name, 'r') as file:

for line in file:

parts = line.strip().split(':')

student = parts[0].strip()

courses = [course.strip() for course in parts[1].split(',')]

students\_dict[student] = courses

except FileNotFoundError:

print(f"Error: File '{file\_name}' not found.")

except Exception as e:

print(f"An error occurred while reading the file '{file\_name}': {e}")

return students\_dict

Next, a function called `read\_students\_file` that takes in a string `file\_name` as an argument. The function reads student data from a file specified by `file\_name` and returns it as a dictionary. It initializes an empty dictionary called `students\_dict`. The function attempts to open the file with the given `file\_name` in read mode using a `with` statement. It then iterates over each line in the file. Each line is split using the colon (':') delimiter, and the first part is considered as the student’s name, while the second part is split using a comma (',') delimiter to obtain a list of courses for that student. The student’s name and the list of courses are added as a key-value pair to the `students\_dict`. If the file is not found, it prints an error message. If any other exception occurs during the file reading process, it catches the exception and prints an error message with the details. Finally, it returns the `students\_dict`.

**write\_inverted\_dict\_to\_file(inverted\_dict, file\_name)**

def write\_inverted\_dict\_to\_file(inverted\_dict, file\_name):

"""

Function to write an inverted dictionary to a file.

Args:

inverted\_dict (dict): The inverted dictionary to be written to the file.

file\_name (str): The name of the file to write to.

"""

try:

with open(file\_name, 'w') as file:

for key, values in inverted\_dict.items():

file.write(key + ': ' + ', '.join(values) + '\n')

except Exception as e:

print(f"An error occurred while writing to the file '{file\_name}': {e}”)

Then a function called `write\_inverted\_dict\_to\_file` that takes in an inverted dictionary `inverted\_dict` and a string `file\_name` as arguments. The function writes the inverted dictionary to a file specified by `file\_name`. It attempts to open the file with the given `file\_name` in write mode using a `with` statement. It iterates over the key-value pairs in the `inverted\_dict`. For each key-value pair, it writes the key, followed by a colon and a space (': '), and then joins the values with commas using the `join` function to create a string representation. The resulting string is written to the file, followed by a newline character. If any exception occurs during the file writing process, it catches the exception and prints an error message with the details.

**Reading and inverting the student data**

# Read from students.txt

students\_dict = read\_students\_file('students.txt')

# Invert the dictionary

inverted\_dict = invert\_dictionary(students\_dict)

Here the program reads the student data from a file called 'students.txt' using the `read\_students\_file` function and stores it in the `students\_dict` dictionary. It then calls the `invert\_dictionary` function, passing the `students\_dict` as an argument, to obtain the inverted dictionary, which is stored in the `inverted\_dict` variable.

**Writing the inverted dictionary to a file**

# Write the inverted dictionary to another file

write\_inverted\_dict\_to\_file(inverted\_dict, 'inverted\_students.txt')

The `write\_inverted\_dict\_to\_file` function, passing the `inverted\_dict` and a file name 'inverted\_students.txt' as arguments. It writes the inverted dictionary to the file.

**Displaying the original and inverted files**

# Display original and inverted files to the screen

print("Original students.txt:")

try:

with open('students.txt', 'r') as file:

print(file.read())

except FileNotFoundError:

print("Error: Original students file not found.")

print("\nInverted inverted\_students.txt:")

try:

with open('inverted\_students.txt', 'r') as file:

print(file.read())

except FileNotFoundError:

print("Error: Inverted students file not found.”)

Lastly, we print the contents of the original file 'students.txt' and the inverted file 'inverted\_students.txt' to the screen. It attempts to open each file using a `with` statement and reads the file contents using the `read` method. If a file is not found, it prints an error message.

**Original Student Source .txt file**

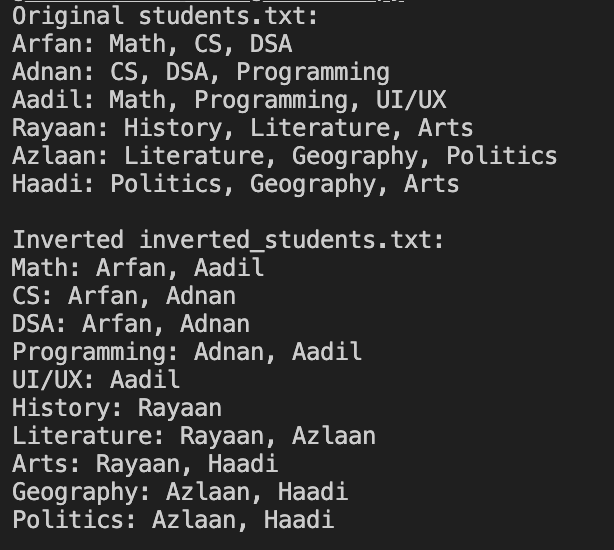
Arfan: Math, CS, DSA

Adnan: CS, DSA, Programming

Aadil: Math, Programming, UI/UX

Rayaan: History, Literature, Arts

Azlaan: Literature, Geography, Politics

Haadi: Politics, Geography, Arts

**Inverted Students .txt file**

Math: Arfan, Aadil

CS: Arfan, Adnan

DSA: Arfan, Adnan

Programming: Adnan, Aadil

UI/UX: Aadil

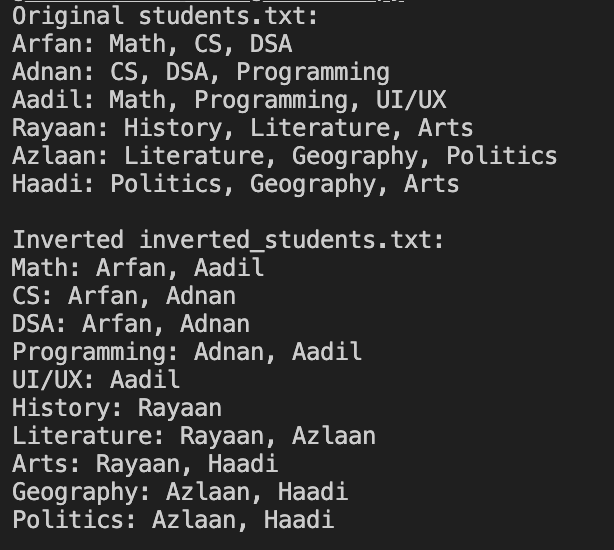
History: Rayaan

Literature: Rayaan, Azlaan

Arts: Rayaan, Haadi

Geography: Azlaan, Haadi

Politics: Azlaan, Haadi

**Output**

**Reference**

Python Official Documentation. (Python 3.12.2). *The official documentation.* Python

[*https://docs.python.org/3/*](https://docs.python.org/3/)

PFB Staff Writer. (2021). *Reading and Writing Files in Python.* Python for Beginners

[*https://www.pythonforbeginners.com/files/reading-and-writing-files-in-python*](https://www.pythonforbeginners.com/files/reading-and-writing-files-in-python)

Said van de Klundert. (2024). *Python Exceptions*. Real Python

[*https://realpython.com/python-exceptions/*](https://realpython.com/python-exceptions/)

John Sturtz. (2024). *Python Dictionaries*. Real Python

[*https://realpython.com/python-dicts/*](https://realpython.com/python-dicts/)

James Timmins. (2024)*. List Comprehensions.* Real Python

[*https://realpython.com/list-comprehension-python/*](https://realpython.com/list-comprehension-python/)

Python Official Documentation. (Python 3.12.2). *Official Python Tutorial.* Python

[*https://docs.python.org/3/tutorial/index.html*](https://docs.python.org/3/tutorial/index.html)

Matthes, Eric. (2023). *Python Crash Course 3ed.* No Starch Press

Lutz, Mark. (2013). *Learning Python 5ed.* O’Reilly