



南通大学人工智能与计算机学院

School of Artificial Intelligence and Computer Science, Nantong University

(阿里云大数据学院)

(人工智能研究院合署)

数据结构课程设计，算法设计与分析

**Curriculum Design for Data Structure and Algorithms,
Algorithm Design and Analysis**

REPORT ON

Student Transcript Management System

Supervisor

Dr. 王皓晨 (Wang Haochen)

School of Artificial Intelligence and Computer Science
Nantong University

Submitted By

Vaskar Chakma

2130130204

2025 年 01 月 01 日

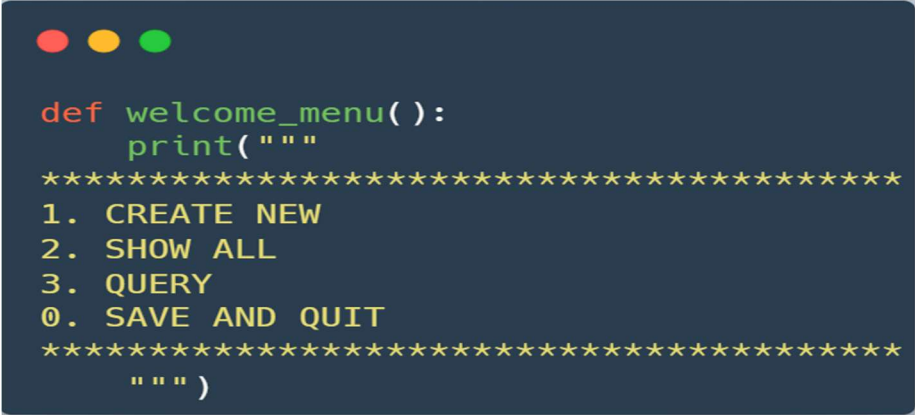
Introduction

The Student Transcript Management System is a simple and efficient tool for managing student records. It helps users store, update, and save information about students, such as their names, IDs, majors, subjects, and scores. The program is written in Python and uses an easy-to-navigate menu, making it user-friendly for everyone. The system has several features, such as *adding new students*, *viewing all records*, *searching for specific students*, and *saving the data to a file*. These features work together to make managing student data faster and more accurate. By saving data in a (`import csv`) CSV file, the system ensures that no information is lost when the program closes. This makes it useful for schools, colleges, or any organization that needs to handle student information in an organized way.

Welcome Menu

The *Welcome Menu* is the starting interface of the Student Transcript Management System. It displays a simple and structured list of options for the user to interact with. These options represent the main functionalities of the system and are assigned specific numbers to make navigation intuitive and efficient. The menu includes the following options:

1. **CREATE NEW:** Allows the user to add a new student record to the system.
2. **SHOW ALL:** Displays all the student records stored in the database, with sorting options available.
3. **QUERY:** Enables the user to search for a specific student and perform operations such as modifying or deleting their details.
0. **SAVE AND QUIT:** Saves the current student data to a CSV file and exits the program.



```
def welcome_menu():
    print("""
*****
1. CREATE NEW
2. SHOW ALL
3. QUERY
0. SAVE AND QUIT
*****
""")
```

Output

```
PS I:\Classes and Assignments\4th 1st\Algorithm Design> & "C:/Users/MR. VASKAR CHAKMA/anscript Management System.py"
```

```
*****
1. CREATE NEW
2. SHOW ALL
3. QUERY
0. SAVE AND QUIT
*****
```

Create New Module

The Create New Module is a vital part of the Student Transcript Management System, allowing users to add new student records. This feature is designed to gather essential details about each student and store them in a structured format for future use. It ensures that all necessary information is captured accurately while providing flexibility for continuous data entry.

Functionality

When the user selects the '1' for "CREATE NEW" option, this module is executed. The program prompts the user to input specific details for the student, including:

Major: The student's field of study.

ID: A unique identifier for the student.

Name: The student's full name.

Gender: The gender of the student.

Subject: The subject the student is enrolled in.

Score: The student's score for the respective subject.

Each record is stored as a dictionary and appended to a list that acts as the database of the program. After successfully creating a record, the user is asked whether they would like to create another record. This ensures that multiple entries can be handled efficiently in one session.

```
def create_new(students):
    while True:
        print("Create a new student. Please input the following details:")
        major = input("Major: ")
        student_id = input("ID: ")
        name = input("Name: ")
        gender = input("Gender: ")
        subject = input("Subject: ")
        score = int(input("Score: "))

        # Add the new student to the list
        students.append({
            "Major": major,
            "ID": student_id,
            "Name": name,
            "Gender": gender,
            "Subject": subject,
            "Score": score
        })
        print("Student info successfully created.")

        # Ask if the user wants to create another student
        continue_choice = input("Would you like to continue to create a new one? Y/N: ").strip().upper()
        if continue_choice != 'Y':
            break
```

Output

```
Please choose a module: 1
Create a new student. Please input the following details:
Major: CST
ID: 2130130204
Name: Vaskar Chakma
Gender: Male
Subject: Software Engineering
Score: 98
Student info successfully created.
Would you like to continue to create a new one? Y/N: Y
```

Show All Module

The Show All Module provides users with a comprehensive view of all student records stored in the system. It is designed to display the data in a structured format, ensuring clarity and readability. This module also includes sorting functionality, allowing users to organize the records based on their preferences.

Functionality

When the user selects the "SHOW ALL" option, the program performs the following steps:

- **Check for Records:** If no records exist, the system displays a message: "No students to display." and returns to the main menu.
- **Sorting Options:** Users can choose how the records are displayed:
- **Sort by ID:** Organizes the records in ascending order based on student IDs.
- **Sort by Score:** Displays the records in descending order of scores, highlighting the highest achievers first.
- **Display Records:** The student data is presented in a table-like format with columns for Major, ID, Name, Gender, Subject, and Score. This layout ensures that users can quickly scan and understand the information.

```
def show_all(students):
    if not students:
        print("No students to display.")
        return

    print("Please choose the mode for display:")
    print("1. Sort by ID\n2. Sort by Score")
    choice = int(input("Your choice: "))

    if choice == 1:
        students.sort(key=lambda x: x["ID"])
    elif choice == 2:
        students.sort(key=lambda x: x["Score"], reverse=True)

    print(f"{'Major':<10}{ 'ID':<10}{ 'Name':<15}{ 'Gender':<10}{ 'Subject':<15}{ 'Score':<10}")
    for student in students:
        print(f"{'Major':<10}{student['Major']:<10}{student['ID']:<10}{student['Name']:<15}{student['Gender']:<10}{student['Subject']:<15}{student['Score']:<10}")
```

Output

```
*****
1. CREATE NEW
2. SHOW ALL
3. QUERY
0. SAVE AND QUIT
*****

Please choose a module: 2
Please choose the mode for display:
1. Sort by ID
2. Sort by Score
Your choice: 1
Major      ID      Name      Gender    Subject    Score
CST        2130130204 Vaskar Chakma Male      Software Engineering 98
```

Query Module

The *query_module()* function helps the user find a student by their name and shows their details, such as major, ID, name, gender, subject, and score. After viewing the details, the user can choose to either modify the student's information, delete the record, or return to the main menu.

If the user selects to modify the student's details, the *modify_student()* function allows them to update specific information like the student's name, major, gender, subject, or score. If the user chooses to delete the record, the student is removed from the list. The function also ensures that the user can cancel the operation or correct any invalid choices.

```
def query_module(students):
    name_to_query = input("Please input the student name for query: ")
    filtered_students = [s for s in students if s['Name'] == name_to_query]

    if not filtered_students:
        print("Student not found.")
        return

    student = filtered_students[0]
    print(f"Major: {student['Major']}\nID: {student['ID']}\nName: {student['Name']}\nGender: {student['Gender']}\nSubject: {student['Subject']}\nScore: {student['Score']}")

    print("Please choose the operation on this student:")
    print("1. Modify\n2. Delete\n0. Back to the menu")
    choice = int(input("Your choice: "))

    if choice == 1:
        modify_student(student)
    elif choice == 2:
        students.remove(student)
        print("Student record deleted successfully.")
```


Output

```
*****
1. CREATE NEW
2. SHOW ALL
3. QUERY
0. SAVE AND QUIT
*****
```

```
Please choose a module: 3
Please input the student name for query: Vaskar Chakma
Major: CST
ID: 2130130204
Name: Vaskar Chakma
Gender: Male
Subject: Software Engineering
Score: 98
```

Modification

The `modify_student()` function allows the user to update a student's information. It prompts the user to modify the student's major, subject, and score. For each field, the current value is displayed, and the user can either press Enter to keep the existing value or input a new one. If the user provides a new score, it is converted to an integer; otherwise, the score remains unchanged. After all changes are made, the function confirms that the modifications were successful. This function provides a simple and efficient way to edit a student's details without requiring the user to re-enter values they wish to keep.

```
def modify_student(student):
    print("Modify the info of the student. Press Enter to keep the current value.")

    student['Major'] = input(f"Major ({student['Major']}): ") or student['Major']
    student['Subject'] = input(f"Subject ({student['Subject']}): ") or student['Subject']
    score_input = input(f"Score ({student['Score']}): ")
    student['Score'] = int(score_input) if score_input else student['Score']

    print("Modification is successful!")
```

Output

```
*****
1. CREATE NEW
2. SHOW ALL
3. QUERY
0. SAVE AND QUIT
*****

Please choose a module: 3
Please input the student name for query: Vaskar Chakma
Major: CST
ID: 2130130204
Name: Vaskar Chakma
Gender: Male
Subject: Software Engineering
Score: 98
Please choose the operation on this student:
1. Modify
2. Delete
0. Back to the menu
Your choice: 1
Modify the info of the student. Press Enter to keep the current value.
Major (CST): C.S.T.
Subject (Software Engineering): Soft. Engineering
Score (98): 99
Modification is successful!
```

Delete

In the delete section, the user is given the option to remove a student's record from the list. After querying a student by name and viewing their details, the user can choose the "Delete" option. If the user selects to delete the student's record, the `query_module()` function removes the student from the list of students using the `remove()` method. Once the student is deleted, a confirmation message is displayed, informing the user that the student's record has been successfully deleted. This option allows for easy removal of outdated or incorrect student records from the system.

```
def query_module(students):
    name_to_query = input("Please input the student name for query: ")
    filtered_students = [s for s in students if s['Name'] == name_to_query]

    if not filtered_students:
        print("Student not found.")
        return

    student = filtered_students[0]
    print(f"Major: {student['Major']}\nID: {student['ID']}\nName: {student['Name']}\nGender: {student['Gender']}\nSubject: {student['Subject']}\nScore: {student['Score']}")

    print("Please choose the operation on this student:")
    print("1. Modify\n2. Delete\n0. Back to the menu")
    choice = int(input("Your choice: "))

    if choice == 1:
        modify_student(student)
    elif choice == 2:
        students.remove(student)
        print("Student record deleted successfully.")
```

Output

```
Please choose the operation on this student:
1. Modify
2. Delete
0. Back to the menu
Your choice: 2
Student record deleted successfully.
```

Quit

In the quit section, the `save_and_quit()` function is responsible for saving all student data to a CSV file before exiting the program. When the user chooses to quit, this function is called. It opens the `student.csv` file in write mode and uses the `csv.DictWriter` to write the data. The `fieldnames` are defined, which represent the columns in the CSV file, and the function first writes the header row using `writer.writeheader()`. Then, it writes all the student records from the `students` list to the file using `writer.writerows()`. Once the data is saved, a confirmation message is displayed, letting the user know that the file has been successfully saved and the system is ready for the next use. This ensures that any changes made to the student data are preserved for future use.

```

def save_and_quit(students):
    with open('student.csv', 'w', newline='') as csvfile:
        fieldnames = ['Major', 'ID', 'Name', 'Gender', 'Subject', 'Score']
        writer = csv.DictWriter(csvfile, fieldnames=fieldnames)

        writer.writeheader()
        writer.writerows(students)

    print("File successfully saved. Welcome next use!")

```

Output

```

*****
1. CREATE NEW
2. SHOW ALL
3. QUERY
0. SAVE AND QUIT
*****

Please choose a module: 0
File successfully saved. Welcome next use!

```

After Deletion the *student.csv* file is empty.

	A	B	C	D	E	F
1	Major	ID	Name	Gender	Subject	Score
2						
3						
4						
5						
6						
7						
8						
9						
10						

“Student Transcript Management System” Program Code (Python)

```

import csv
def welcome_menu():
    print("""
*****
1. CREATE NEW
2. SHOW ALL
3. QUERY
0. SAVE AND QUIT
*****
""")

```



```
def create_new(students):
    while True:
        print("Create a new student. Please input the following details:")
        major = input("Major: ")
        student_id = input("ID: ")
        name = input("Name: ")
        gender = input("Gender: ")
        subject = input("Subject: ")
        score = int(input("Score: "))

        # Add the new student to the list
        students.append({
            "Major": major,
            "ID": student_id,
            "Name": name,
            "Gender": gender,
            "Subject": subject,
            "Score": score
        })
        print("Student info successfully created.")

        # Ask if the user wants to create another student
        continue_choice = input("Would you like to continue to create a new
one? Y/N: ").strip().upper()
        if continue_choice != 'Y':
            break

def show_all(students):
    if not students:
        print("No students to display.")
        return

    print("Please choose the mode for display:")
    print("1. Sort by ID\n2. Sort by Score")
    choice = int(input("Your choice: "))

    if choice == 1:
        students.sort(key=lambda x: x["ID"])
    elif choice == 2:
        students.sort(key=lambda x: x["Score"], reverse=True)

    print(f"{'Major':<10}{'ID':<10}{'Name':<15}{'Gender':<10}{'Subject':<15}{'Score':<10}")
    for student in students:
        print(f"{student['Major']:<10}{student['ID']:<10}{student['Name']:<15}{student['Gender']:<10}{student['Subject']:<15}{student['Score']:<10}")
```

```

def query_module(students):
    name_to_query = input("Please input the student name for query: ")
    filtered_students = [s for s in students if s['Name'] == name_to_query]

    if not filtered_students:
        print("Student not found.")
        return

    student = filtered_students[0]
    print(f"Major: {student['Major']}\nID: {student['ID']}\nName: {student['Name']}\nGender: {student['Gender']}\nSubject: {student['Subject']}\nScore: {student['Score']}")

    print("Please choose the operation on this student:")
    print("1. Modify\n2. Delete\n0. Back to the menu")
    choice = int(input("Your choice: "))

    if choice == 1:
        modify_student(student)
    elif choice == 2:
        students.remove(student)
        print("Student record deleted successfully.")

def modify_student(student):
    print("Modify the info of the student. Press Enter to keep the current value.")

    student['Major'] = input(f"Major ({student['Major']}): ") or student['Major']
    student['Subject'] = input(f"Subject ({student['Subject']}): ") or student['Subject']
    score_input = input(f"Score ({student['Score']}): ")
    student['Score'] = int(score_input) if score_input else student['Score']

    print("Modification is successful!")

def save_and_quit(students):
    with open('student.csv', 'w', newline='') as csvfile:
        fieldnames = ['Major', 'ID', 'Name', 'Gender', 'Subject', 'Score']
        writer = csv.DictWriter(csvfile, fieldnames=fieldnames)

        writer.writeheader()
        writer.writerows(students)

    print("File successfully saved. Welcome next use!")

def main():
    students = []

```

```
while True:
    welcome_menu()
    choice = int(input("Please choose a module: "))

    if choice == 1:
        create_new(students)
    elif choice == 2:
        show_all(students)
    elif choice == 3:
        query_module(students)
    elif choice == 0:
        save_and_quit(students)
        break
    else:
        print("Invalid choice. Please try again.")

if __name__ == "__main__":
    main()
```

[N.B.: All the code snippet screenshots have been taken using <https://carbon.now.sh/>]

THE END