2024

PROGRAMING FOR AI

FAREENA NOOR(SP23-BAI-012)
FARHEEN ANWAR(SP23-BAI-013)

LAB MID TERM

REPORT

TUPLES:

```
def isValid(subj):
    global data
    try:
       for i in range(10):
            if data[0][i]== subj.name or data[0][i]== subj.abbrev:
                return (subj,i)

    except IndexError:
            return -1
```

CLASS:

```
class Subject:
    def __init__(self,name,abbrev,teacher,credit_hrs):
        self.name = name
        self.abbrev = abbrev
        self.teacher = teacher
        self.credit_hrs = credit_hrs
```

While loop:

```
def marksAddded():
    global data
    subj = getSubject()
    while subj==-1:
        subj = getSubject()

    try:
       for i in range(10):
            if data[0][i]== subj.name or data[0][i]== subj.abbrev:
                return i

    except IndexError:
            print("subject marks not added.")
            return -1
```

CONDITIONAL STATEMENT:

```
    if choice == 1:
        studentDetails()

    elif choice==2:
      index = getRollNumber()
      while index==-1:
        print("INVALID ROLL NUMBER.")
        index = getRollNumber()
      updateMarks(index)

    elif choice==3:
        calGPA()

    elif choice==4:
        calculateAverage()

    elif choice==5:
        deleteRC()

    elif choice==6:
      ranking()
```

FOR LOOP:

```
def ranking():
    global data
    found = False
    for i in range(len(data[0])):
        if data[0][i] == "GPA" or  data[0][i] == "gpa":
            index = i
            found = True
            break
    if not found:
        print("GPA NOT CALCULATED YET.")
    else:
        sorted_data = data[data[:, index].argsort()[::-1]]
        #print(sorted_data)
        #print("TOP THREE",sorted_data[1:4])
        for i in range(1,4):
            print(f"Name:{sorted_data[i][1]},GPA:{sorted_data[i][index]}")
```

LISTS:

```
Subject_list = [LA,DB,PAI,CN,Stats]
```

OPERATORS:

```
    credit_list = []

    for subject in Subject_list:
        credit_list.append(subject.credit_hrs)
        credit_sum += subject.credit_hrs
```

BOOLEAN:

```
def getSubject():
        found = False
        subj = input("Enter subject: ")
        for subject in Subject_list:
            if subject.name== subj or subject.abbrev == subj :
                subj = subject
                found =  True
                return subj
        if not found:
            print("Invalid subject name.")
            return -1
```

STRING AND VARIABLE:

```
global data
choice = int(input("Do you want to delete:\n1) ROW\n2) COLUMN: "))
if choice == 1:
    deleteRow()

elif choice==2:
    deleteColumn()
```

NUMPY:

```
gpas = np.round(gpas,2)

gpas = gpas.reshape(-1,1)
gpas_header = np.array([['GPA']])  # Create a header for the GPA column
gpas_with_header = np.vstack([gpas_header, gpas])  # Stack header with GPA value

#print(gpas_with_header)

Add GPA column to the data (ignoring first row header of data)
data = np.hstack((data, gpas_with_header))
```

```
def calculateAverage():
    global data
    index = marksAddded()
    if index != -1:
        try:
            # Convert the column to integers (excluding the header row)
            marks = data[1:, index].astype(float)  # Convert strings to floats
            average = np.mean(marks)  # Calculate the average
            print(f"AVERAGE MARKS FOR {data[0][index]}: {average}")
            #print(data)
        except ValueError:
            print("Error: Non-numeric values found in the column.")
```

```
  total_weighted_scores = np.sum(weighted_scores, axis=1)
  total_weighted_scores = total_weighted_scores.reshape(-1,1)
```

```
if roll_no !=-1:
    data = np.delete(data,roll_no,axis=0)
    print(f"DATA FOR {index} deleted.")
```

FUNCTION:

```
def studentDetails():
    global data
    index = getRollNumber()
    if index == -1:
        print("INVALID ROLL NUMBER.")
        studentDetails()
    else:
        headers = [str(x) for x in data[0][1:]]
        #print(headers)
        print(tabulate([data[index]], headers=headers, tablefmt='grid'))
        # print("name",data[index][1])
```

Imports and Libraries

These libraries are imported to handle:

- **NumPy** (numpy): Efficient manipulation of arrays and numerical calculations.
- **CSV** (csv): Reading and writing CSV files.
- **Tabulate** (tabulate): Formatting data in a tabular format for clear console output.

Class Definitions

Subject Class

The Subject class represents a subject, storing its name, abbreviation, teacher, and credit hours. Instances of this class are created for each subject, with relevant details stored in Subject_list.

File Handling and Data Loading

open_csvfile() Function

- This function opens and reads data from a CSV file, converting it to a NumPy array for easy data manipulation.
- The data array is then globally accessible in the program.

data_to_CSV() Function

- **Purpose**: Saves the data array back to the CSV file after any updates.
- **Usage**: Called whenever data is modified to ensure that changes persist.

Student and Subject Searching

getRollNumber()

- Purpose: Finds the index of a student by roll number.
- Returns the index if found, else returns -1 for invalid entries.

getSubject()

- **Purpose**: Identifies a subject either by its full name or abbreviation.
- Returns the Subject object if matched, else -1.

marksAddded()

- **Purpose:** Checks if marks for a subject are added and returns its column index.

Displaying Data

studentDetails()

- **Purpose**: Displays details of a student in a formatted table.

- **Operation**: Retrieves data for a given roll number and displays it.

```
ENTER: 1
Enter Roll no:10
INVALID ROLL NUMBER.
Enter Roll no:104
+-----+------------+----------------------+------+------+-------+------+---------+-------+
|     | Name       | Email Address        |   LA |   DB |   PAI |   CN |   Stats |   GPA |
+=====+============+======================+======+======+=======+======+=========+=======+
| 104 | Lisa White | lisawhite@example.com |   55 |   45 |    49 |   56 |      86 |  2.29 |
+-----+------------+----------------------+------+------+-------+------+---------+-------+
```

Data Modifying Functions

updateMarks()

- **Purpose:** Updates a student's marks for a given subject.
- **Operation:** Finds the column for the subject and modifies the student's marks, then saves the updated data.

```
ENTER: 2
Enter Roll no:10
INVALID ROLL NUMBER.
Enter Roll no:102
Enter subject: pf
Invalid subject name.
Enter subject: LA
Enter updated marks: 97
data saved
```

Csv file before updating marks:

A	B	C	D	E	F	G	H	I
Roll No	Name	Email Addr	LA	DB	PAI	CN	Stats	GPA
102	Jane Smith	janesmith(	90	78	50	86	67	3
103	Robert Bro	robertbrow	56	67	56	98	98	3.04
104	Lisa White	lisawhite@	55	45	49	56	86	2.29
105	Michael Jo	michaeljol	47	34	30	76	79	2.14
106	Emily Davi	emilydavis	89	67	98	87	78	3.32

Csv file after updating marks:

Marks for Jane Smith has been updated from 90 to 97 for lA.

Roll No	Name	Email Addr	LA	DB	PAI	CN	Stats
102	Jane Smith	janesmith(	97	78	50	86	67
103	Robert Bro	robertbrow	56	67	56	98	98
104	Lisa White	lisawhite@	55	45	49	56	86
105	Michael Jo	michaeljol	47	34	30	76	79
106	Emily Davi	emilydavis	89	67	98	87	78

Calculations

calculateAverage()

- **Purpose:** Calculates and prints the average marks for a given subject.
- **Operation:** Converts column values to floats, calculates the mean, and displays it.

```
ENTER: 4
Enter subject: pf
Invalid subject name.
Enter subject: dld
Invalid subject name.
Enter subject: LA
AVERAGE MARKS FOR LA: 67.4
```

calGPA()

- **Purpose**: Calculates and adds GPA for each student based on weighted subject marks.
- **Operation**: Converts scores to GPA scale, weights them by credit hours, calculates the final GPA, and appends it to data.

Csv file before calculating GPA:

Roll No	Name	Email Addr	LA	DB	PAI	CN	Stats
102	Jane Smith	janesmith(	97	78	50	86	67
103	Robert Bro	robertbrow	56	67	56	98	98
104	Lisa White	lisawhite@	55	45	49	56	86
105	Michael Jo	michaeljol	47	34	30	76	79
106	Emily Davi	emilydavis	89	67	98	87	78

Csv file after calculating GPA:

GPAS are calculated, added to NumPy array which then is saved into a csv file. Thus adding column of GPA in csv file.

A	B	C	D	E	F	G	H	I
Roll No	Name	Email Addr	LA	DB	PAI	CN	Stats	GPA
102	Jane Smith	janesmith(	90	78	50	86	67	3
103	Robert Bro	robertbrow	56	67	56	98	98	3.04
104	Lisa White	lisawhite@	55	45	49	56	86	2.29
105	Michael Jo	michaeljol	47	34	30	76	79	2.14
106	Emily Davi	emilydavis	89	67	98	87	78	3.32

Data Deletion Functions

deleteRow() and deleteColumn()

- **Purpose**: Deletes a row (student) or a column (subject).
- **Operation**: Locates and removes specified entries and updates the CSV file.

deleteColumn():

```
ENTER: 5
Do you want to delete:
1) ROW
2) COLUMN: 2
Enter column name: pf
pfdoes not exist.
Enter column name: GPA
GPA DELETED.
data saved
```

Csv file before deleting GPA column:

	A	B	C	D	E	F	G	H	I
	Roll No	Name	Email Addr	LA	DB	PAI	CN	Stats	GPA
	102	Jane Smith	janesmith(	90	78	50	86	67	3
	103	Robert Bro	robertbrov	56	67	56	98	98	3.04
	104	Lisa White	lisawhite@	55	45	49	56	86	2.29
	105	Michael Jo	michaeljol	47	34	30	76	79	2.14
	106	Emily Davi	emilydavis	89	67	98	87	78	3.32

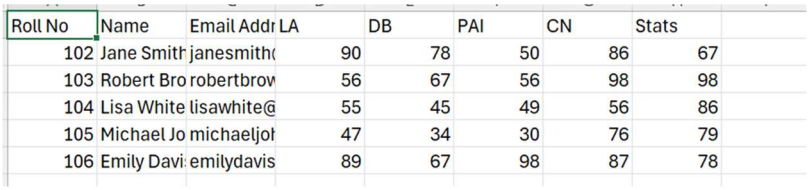
Csv file after deleting GPA column:

Roll No	Name	Email Addr	LA	DB	PAI	CN	Stats
102	Jane Smith	janesmith(	90	78	50	86	67
103	Robert Bro	robertbrov	56	67	56	98	98
104	Lisa White	lisawhite@	55	45	49	56	86
105	Michael Jo	michaeljol	47	34	30	76	79
106	Emily Davi	emilydavis	89	67	98	87	78

deleteRow():

```
1) ROW
2) COLUMN: 1
Enter Roll no:10
Invalid Roll number.
Enter Roll no:107
Invalid Roll number.
Enter Roll no:106
DATA FOR 106 deleted.
data saved
```

Csv file before deleting row:



Roll No	Name	Email Addr	LA	DB	PAI	CN	Stats
102	Jane Smith	janesmith(	90	78	50	86	67
103	Robert Bro	robertbrov	56	67	56	98	98
104	Lisa White	lisawhite@	55	45	49	56	86
105	Michael Jo	michaeljol	47	34	30	76	79
106	Emily Davi	emilydavis	89	67	98	87	78

Csv file after deleting row:

We can see that the row for roll number 106 has been deleted from the csv file.

Roll No	Name	Email Addr	LA	DB	PAI	CN	Stats
102	Jane Smith	janesmith(	97	78	50	86	67
103	Robert Bro	robertbrov	56	67	56	98	98
104	Lisa White	lisawhite@	55	45	49	56	86
105	Michael Jo	michaeljol	47	34	30	76	79

Additional Functionalities

ranking()

- **Purpose**: Displays the top three students based on GPA.

Output If GPAS Are Calculated:

```
ENTER: 6
Name:Emily Davis,GPA:3.32
Name:Robert Brown,GPA:3.06
Name:Jane Smith,GPA:3.0
```

Output If GPAS Are Not Calculated:

```
ENTER: 6
GPA NOT CALCULATED YET.
```

User Menu and Main Program Loop

menu()

- **Purpose**: Provides a user-friendly command menu to select various functionalities, loop back to the menu, or exit.

Summary

This code is a student management system that supports essential operations like:

- **Viewing and updating student information**.
- **Calculating GPAs and subject averages**.
- **Managing data by deleting rows and columns**.