Programming Project (Registration application)

You are tasked with building an application that handles club registration for school students. This application is used by school administrators whenever a student wants to register for some sessions offered by a club. There are several clubs that offer extracurricular activities. For example, Yoga club, Drama club, Music club, Sports club, Debate club, Dance club, and Outdoor club.

What do I need to do?

These tasks involve developing a registration system. You will also be required to enter values in this registration system. These values will be fictitious, and you need to make them up. Each task mirrors what you have done in class (I.e. the Supermarket example, for which we build the Cashier application), and you can refer to your class notes to help guide you to do the tasks in this programming project/assessment successfully.

The application is called "Rego" and is written in Python. The entry-point (main) source code is stored in a file named "rego.py". Rego runs in Terminal. It is launched by typing "python rego.py" in Terminal after navigating to the directory where the file rego.py is located (and after activating the suitable Python Conda environment).

Rego has six functionalities, which are described in the following sections. To score marks, you need to implement these functionalities. The maximum mark is 100.

The submission instruction is provided in the last section.

(Functionality 0) Menu display and selection (mark: 15/100)

Once launched, Rego displays a set of five activity menus, and asks for a menu selection from the user. If a valid menu is selected, Rego will execute the corresponding procedure. After this procedure finishes, Rego will display the five activity menus again. This cycle (I.e., menu selection, menu execution, then menu selection, and so on) is repeated until the user enters the number 0, by which Rego terminates.

Rego provides five menus for its user to select from. Each menu is represented by a positive integer, namely 1, 2, 3, 4, and 5. If a number outside the range of 0, 1, …, 5 is entered, then Rego will simply ignore it and display the menu message again, continuing the menu loop. Note that the five menus are explained in the next sections.

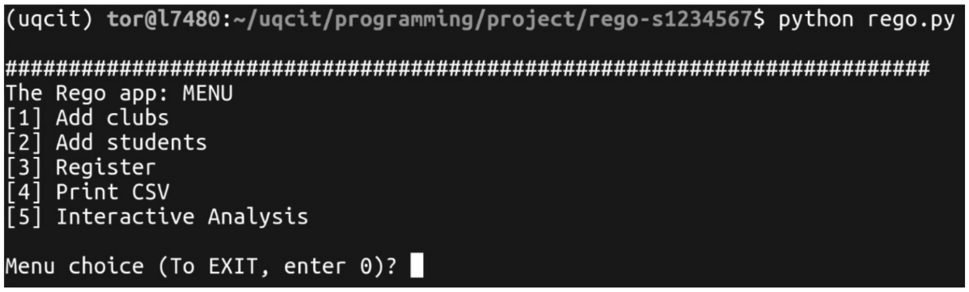
Write code to implement Functionality 0 in a file called "rego.py", which also serves as the entry-point for Rego (such that Rego can be launched by typing the command: "python rego.py").

Functionality 0: breakdowns

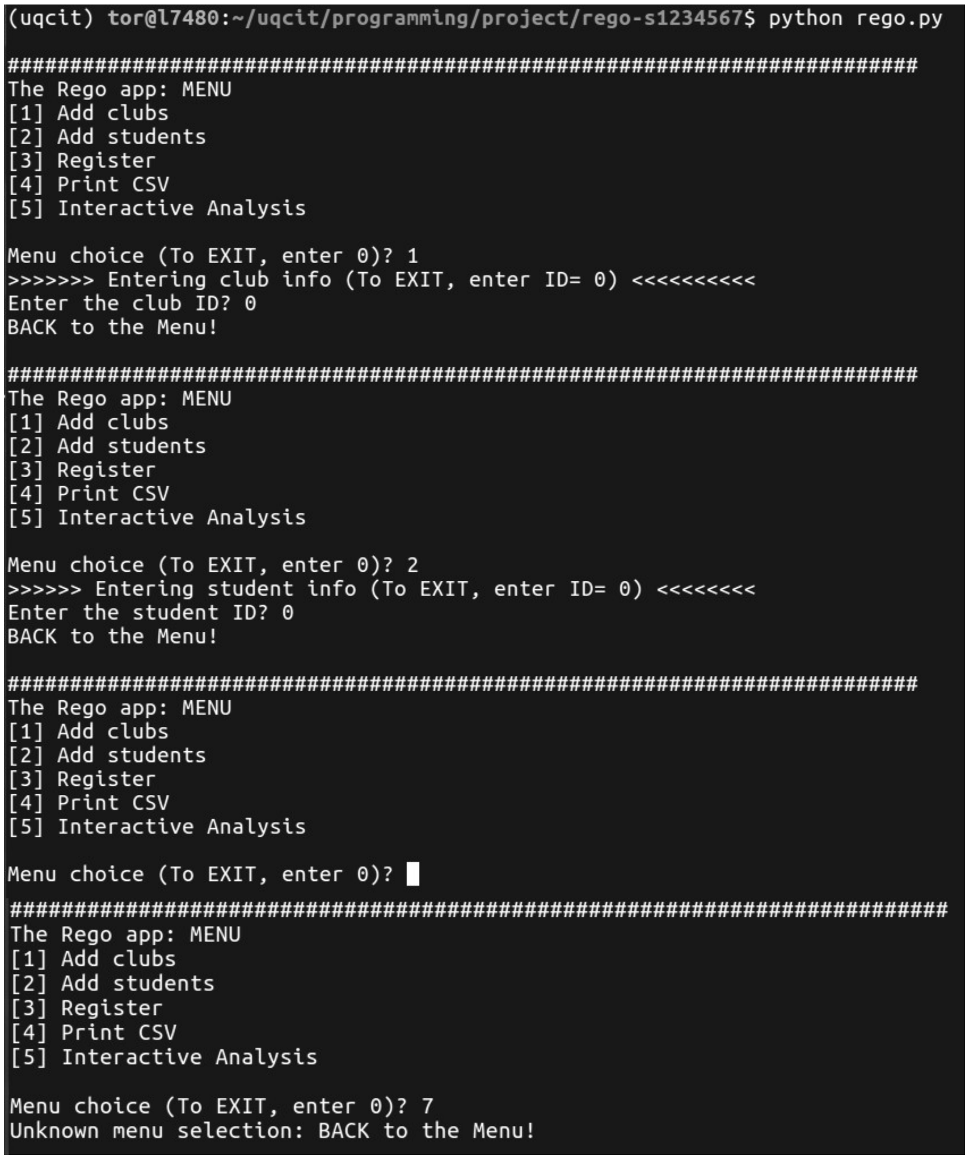
a) Menu message
b) Menu selection input
c) Menu selection/dispatching
d) Menu looping
e) Termination by the number 0
f) Unknown menu number handling

Functionality 0: breakdowns

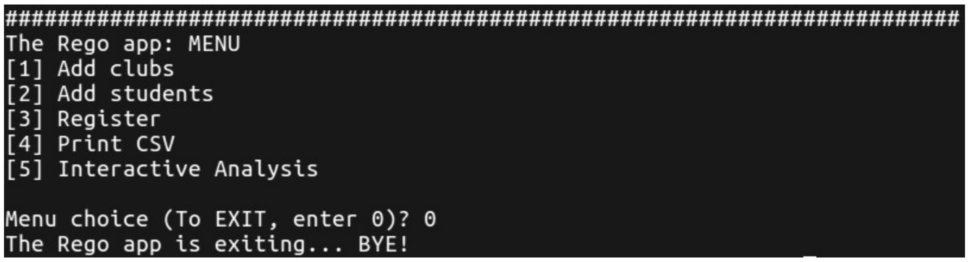
Functionality 0: Sample outputs



```
(uqcit) tor@l7480:~/uqcit/programming/project/rego-s1234567$ python rego.py

##########################################################################
The Rego app: MENU
[1] Add clubs
[2] Add students
[3] Register
[4] Print CSV
[5] Interactive Analysis

Menu choice (To EXIT, enter 0)? 
```



```
(uqcit) tor@l7480:~/uqcit/programming/project/rego-s1234567$ python rego.py

##########################################################################
The Rego app: MENU
[1] Add clubs
[2] Add students
[3] Register
[4] Print CSV
[5] Interactive Analysis

Menu choice (To EXIT, enter 0)? 1
>>>>>>> Entering club info (To EXIT, enter ID= 0) <<<<<<<<<<
Enter the club ID? 0
BACK to the Menu!

##########################################################################
The Rego app: MENU
[1] Add clubs
[2] Add students
[3] Register
[4] Print CSV
[5] Interactive Analysis

Menu choice (To EXIT, enter 0)? 2
>>>>>> Entering student info (To EXIT, enter ID= 0) <<<<<<<<
Enter the student ID? 0
BACK to the Menu!

##########################################################################
The Rego app: MENU
[1] Add clubs
[2] Add students
[3] Register
[4] Print CSV
[5] Interactive Analysis

Menu choice (To EXIT, enter 0)? 
##########################################################################
The Rego app: MENU
[1] Add clubs
[2] Add students
[3] Register
[4] Print CSV
[5] Interactive Analysis

Menu choice (To EXIT, enter 0)? 7
Unknown menu selection: BACK to the Menu!
```



```
##########################################################################
The Rego app: MENU
[1] Add clubs
[2] Add students
[3] Register
[4] Print CSV
[5] Interactive Analysis

Menu choice (To EXIT, enter 0)? 0
The Rego app is exiting... BYE!
```

(Functionality 1) Menu number 1: Add clubs (marks: 20/100)

Once selected (by entering number 1), this selection asks for four types of information, namely

a. ID: This is a club identification (ID), represented as an integer.
b. Name: This is a club name, represented as a string.
c. Quantity: This refers to the number of sessions available in a club. One session can only be taken by one student. This is represented as an integer.
d. Price: This refers to the price of each session (i.e., the unit price). This is represented as a float. Assume there is a unit price per session for casual memberships.

After the above four kinds of information are entered, this club information is added to the existing club record, which is stored in a YAML format called “clubs.yaml”. The updated club record is finally stored again in the same file “clubs.yaml”; overwriting its previous content. Note that when Rego is launched for the first time, the file “clubs.yaml” does not exist. The file “clubs.yaml” will be created after the first club addition.

This menu handles multiple club record additions. That is, by repeatedly asking the set of information, one after another in a loop. To terminate this input loop, the number 0 should be entered as ID.

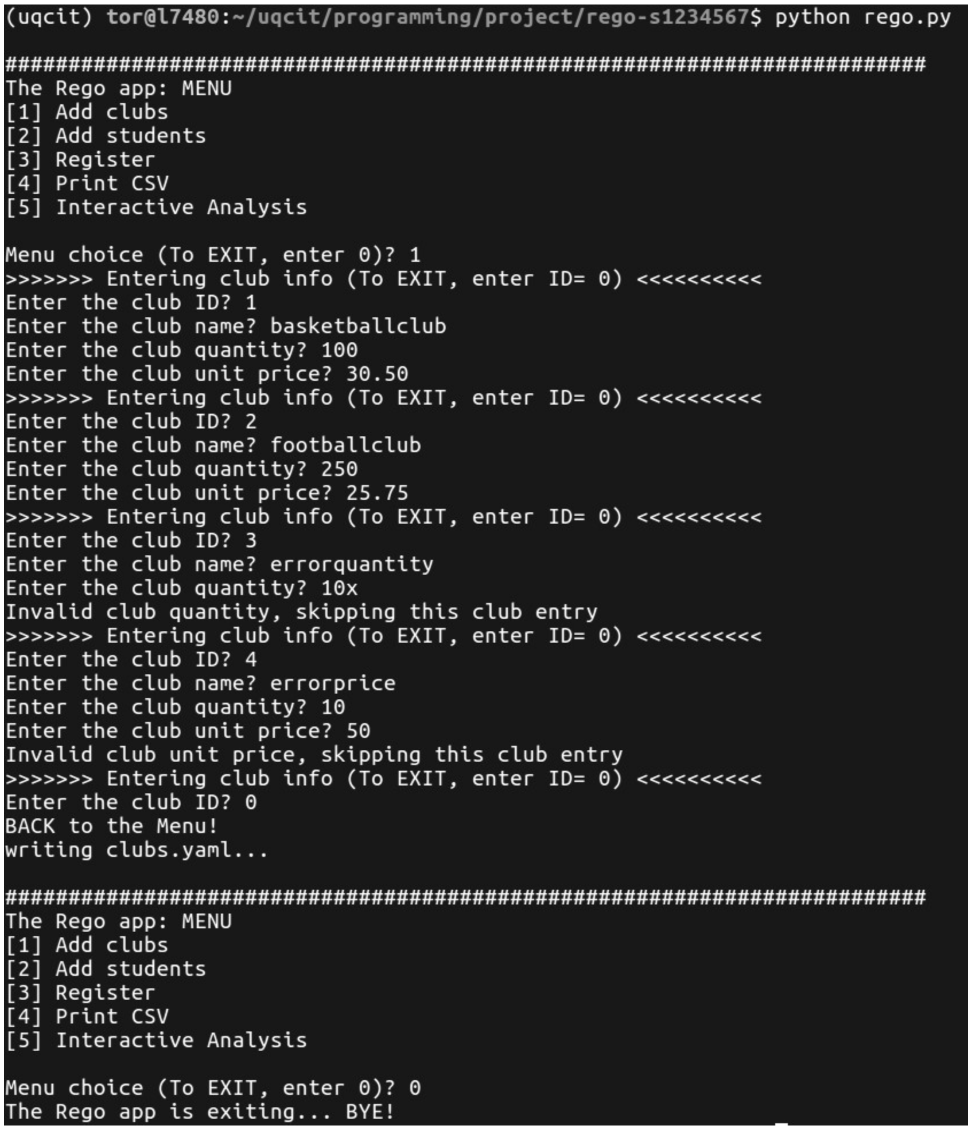
Write code to implement Functionality 1 in a file called “club.py”.

Functionality 1: breakdowns

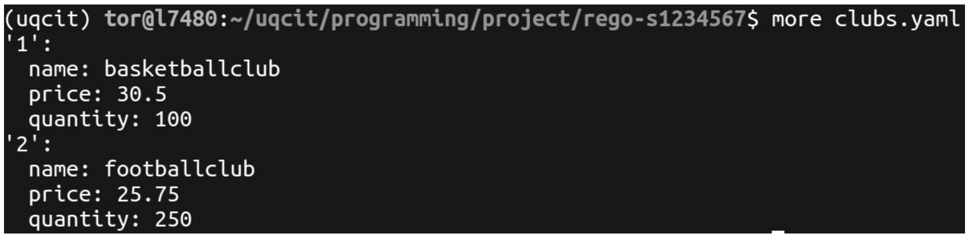
a) Handling of creating the (currently non-existent) file “clubs.yaml” in the first launch.
b) Handling multiple club additions through the club input loop.
c) Club information inputs. This includes handling erroneous non-number inputs for Quantity and Price, for example: if Quantity is entered as “10x”, which cannot be converted to a decimal number. In such a non-number input, the current entry is ignored (and discarded), and the club input loop continues.
d) Updating the club record (from the file “clubs.yaml”) and saving the updated record to the same file “clubs.yaml”. The club record is represented as a dictionary, where the keys are club IDs and the values are dictionaries containing club information (other than ID), for example {“name”: “BasketballClubXYZ”, “quantity”: 100, “price”: 30.00}.

Functionality 1: sample outputs

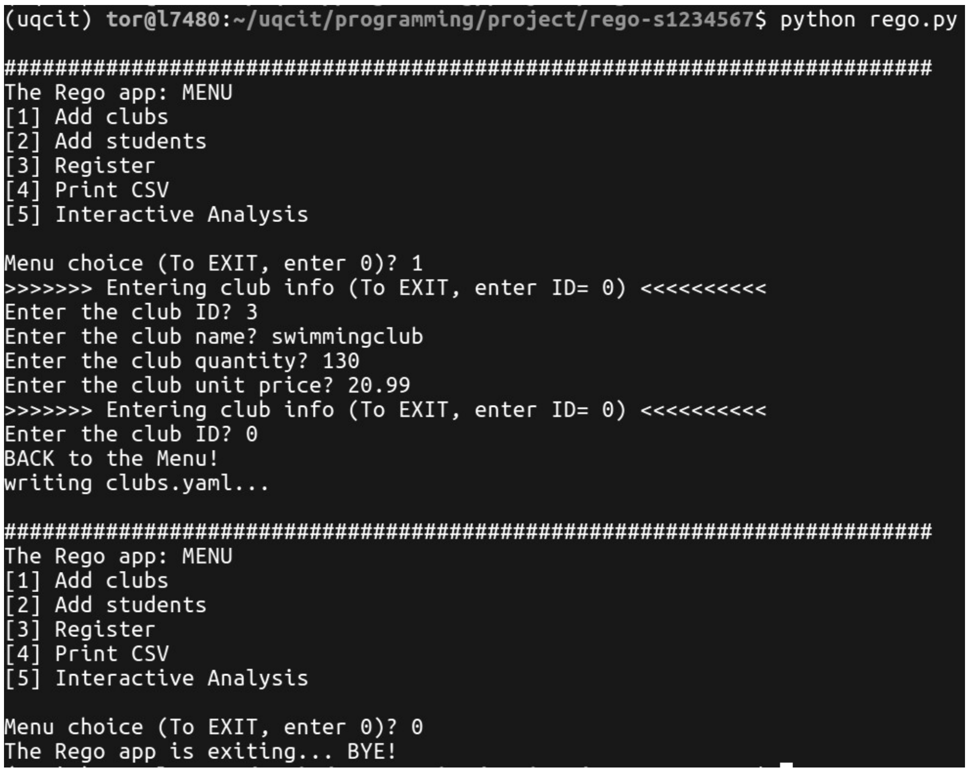
Note that the command “more somefile.txt” in Terminal is used to display the content of “somefile.txt” file in Terminal.



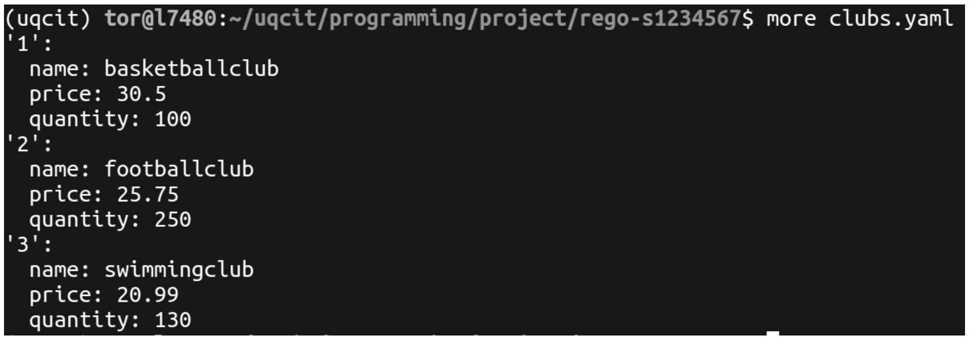
```
(uqcit) tor@l7480:~/uqcit/programming/project/rego-s1234567$ python rego.py

#########################################################################
The Rego app: MENU
[1] Add clubs
[2] Add students
[3] Register
[4] Print CSV
[5] Interactive Analysis

Menu choice (To EXIT, enter 0)? 1
>>>>>>> Entering club info (To EXIT, enter ID= 0) <<<<<<<<<<
Enter the club ID? 1
Enter the club name? basketballclub
Enter the club quantity? 100
Enter the club unit price? 30.50
>>>>>>> Entering club info (To EXIT, enter ID= 0) <<<<<<<<<<
Enter the club ID? 2
Enter the club name? footballclub
Enter the club quantity? 250
Enter the club unit price? 25.75
>>>>>>> Entering club info (To EXIT, enter ID= 0) <<<<<<<<<<
Enter the club ID? 3
Enter the club name? errorquantity
Enter the club quantity? 10x
Invalid club quantity, skipping this club entry
>>>>>>> Entering club info (To EXIT, enter ID= 0) <<<<<<<<<<
Enter the club ID? 4
Enter the club name? errorprice
Enter the club quantity? 10
Enter the club unit price? 50
Invalid club unit price, skipping this club entry
>>>>>>> Entering club info (To EXIT, enter ID= 0) <<<<<<<<<<
Enter the club ID? 0
BACK to the Menu!
writing clubs.yaml...

#########################################################################
The Rego app: MENU
[1] Add clubs
[2] Add students
[3] Register
[4] Print CSV
[5] Interactive Analysis

Menu choice (To EXIT, enter 0)? 0
The Rego app is exiting... BYE!
```



```
(uqcit) tor@l7480:~/uqcit/programming/project/rego-s1234567$ more clubs.yaml
'1':
  name: basketballclub
  price: 30.5
  quantity: 100
'2':
  name: footballclub
  price: 25.75
  quantity: 250
```



```
(uqcit) tor@l7480:~/uqcit/programming/project/rego-s1234567$ python rego.py

#######################################################################
The Rego app: MENU
[1] Add clubs
[2] Add students
[3] Register
[4] Print CSV
[5] Interactive Analysis

Menu choice (To EXIT, enter 0)? 1
>>>>>>> Entering club info (To EXIT, enter ID= 0) <<<<<<<<<<
Enter the club ID? 3
Enter the club name? swimmingclub
Enter the club quantity? 130
Enter the club unit price? 20.99
>>>>>>> Entering club info (To EXIT, enter ID= 0) <<<<<<<<<<
Enter the club ID? 0
BACK to the Menu!
writing clubs.yaml...

#######################################################################
The Rego app: MENU
[1] Add clubs
[2] Add students
[3] Register
[4] Print CSV
[5] Interactive Analysis

Menu choice (To EXIT, enter 0)? 0
The Rego app is exiting... BYE!
```



```
(uqcit) tor@l7480:~/uqcit/programming/project/rego-s1234567$ more clubs.yaml
'1':
  name: basketballclub
  price: 30.5
  quantity: 100
'2':
  name: footballclub
  price: 25.75
  quantity: 250
'3':
  name: swimmingclub
  price: 20.99
  quantity: 130
```

(Functionality 2) Menu number 2: Add students (marks: 15/100)

This menu is similar to Menu number 1 (Functionality 1). Once selected (by entering number 2), this selection asks for four types of information, namely

a. ID: This is a student identification (ID), represented as an integer.
b. Name: This is a student name, represented as a string.
c. DOB: This refers to the date of birth (DOB) of a student. This is represented as a string in the following format: dd/mm/yyyy, for example, "31/12/2021"
d. Suburb: This refers to the suburb where the student lives. This is represented as a string.

After the above four kinds of information are entered, this student information is added to the existing student record, which is stored in a YAML format called "students.yaml". The updated student record is finally stored again in the same file "students.yaml". Note that when Rego is launched for the first time, the file "students.yaml" does not exist. The file "students.yaml" will be created after the first customer addition.

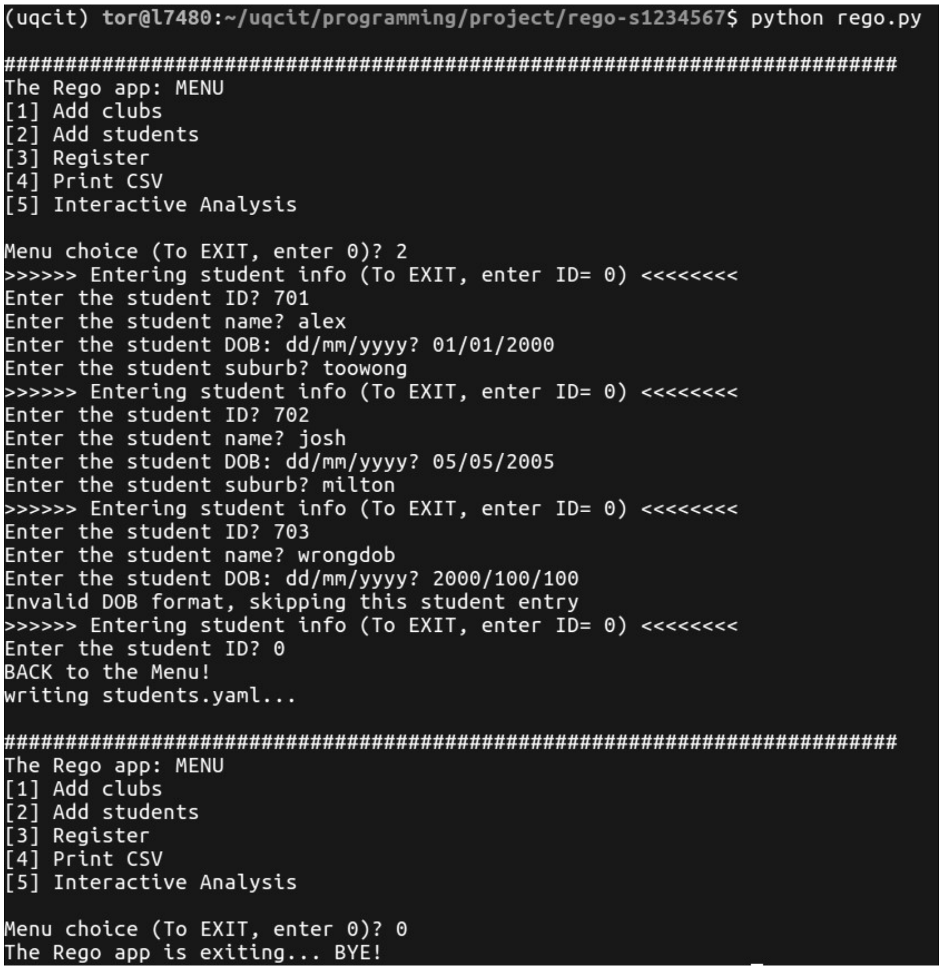
This menu handles multiple student record additions. That is, by repeatedly asking the set of information, one after another in a loop. To terminate this input loop, the number 0 should be entered as ID.

Write code to implement Functionality 2 in a file called "student.py".

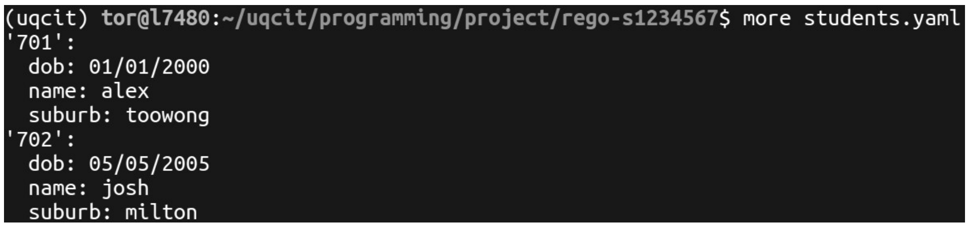
Functionality 2: breakdowns

a) Handling of creating the (currently non-existent) file “students.yaml” in the first launch.
b) Handling multiple student additions through the student input loop.
c) Student information inputs. This includes handling erroneous date format and values for DOB. For example: if DOB is entered as “2000/100/100”, which does not adhere to the required format. In such a wrong format, the current student input is ignored (and discarded), and the student input loop continues.
d) Updating the student record (from the file “students.yaml”) and saving the updated record to the same file “students.yaml”. The student record is represented as a dictionary, where the keys are student IDs and the values are dictionaries containing student information (other than ID), for example {“name”: “Joshua”, “dob”: 20/12/2003, “suburb”: “Milton”}.

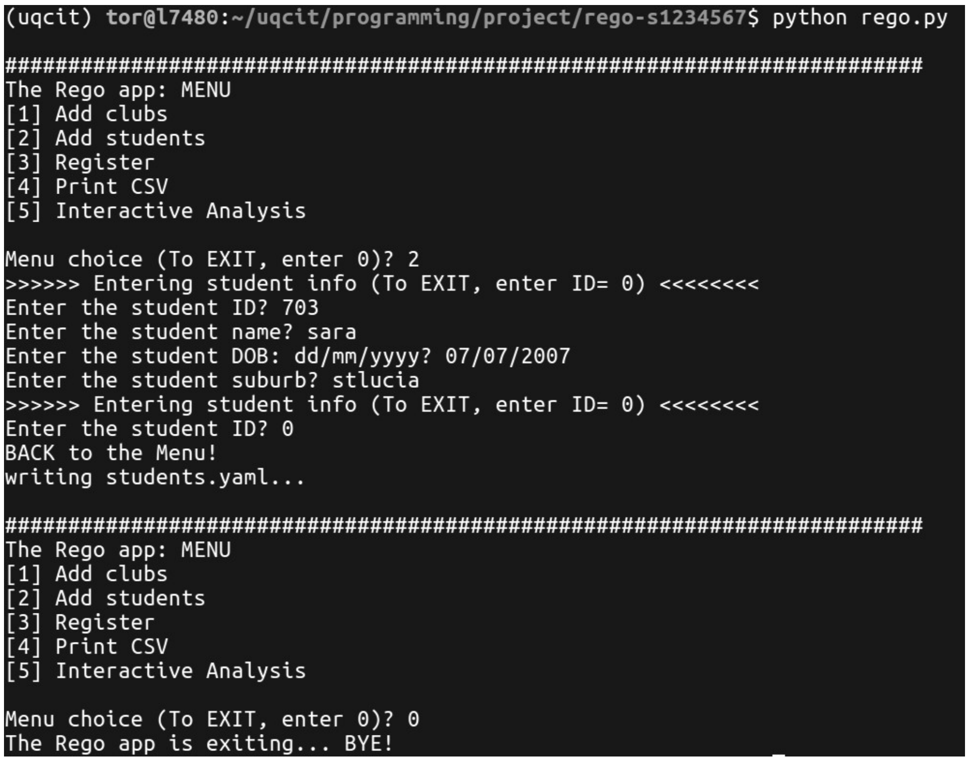
Functionality 2: sample outputs



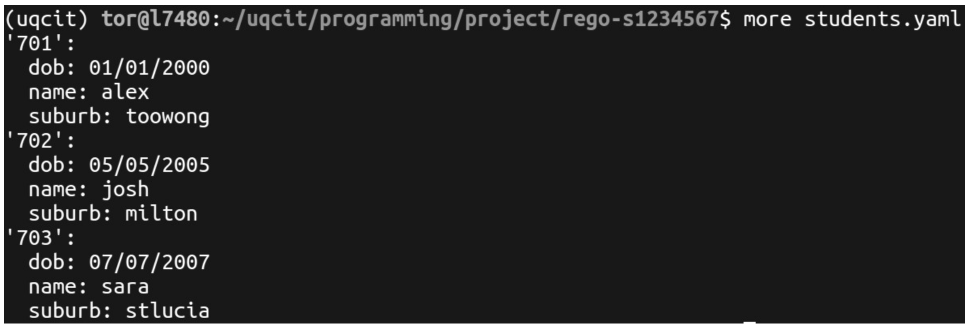
```
(uqcit) tor@l7480:~/uqcit/programming/project/rego-s1234567$ python rego.py

##########################################################################
The Rego app: MENU
[1] Add clubs
[2] Add students
[3] Register
[4] Print CSV
[5] Interactive Analysis

Menu choice (To EXIT, enter 0)? 2
>>>>>> Entering student info (To EXIT, enter ID= 0) <<<<<<<<
Enter the student ID? 701
Enter the student name? alex
Enter the student DOB: dd/mm/yyyy? 01/01/2000
Enter the student suburb? toowong
>>>>>> Entering student info (To EXIT, enter ID= 0) <<<<<<<<
Enter the student ID? 702
Enter the student name? josh
Enter the student DOB: dd/mm/yyyy? 05/05/2005
Enter the student suburb? milton
>>>>>> Entering student info (To EXIT, enter ID= 0) <<<<<<<<
Enter the student ID? 703
Enter the student name? wrongdob
Enter the student DOB: dd/mm/yyyy? 2000/100/100
Invalid DOB format, skipping this student entry
>>>>>> Entering student info (To EXIT, enter ID= 0) <<<<<<<<
Enter the student ID? 0
BACK to the Menu!
writing students.yaml...

##########################################################################
The Rego app: MENU
[1] Add clubs
[2] Add students
[3] Register
[4] Print CSV
[5] Interactive Analysis

Menu choice (To EXIT, enter 0)? 0
The Rego app is exiting... BYE!
```



```
(uqcit) tor@l7480:~/uqcit/programming/project/rego-s1234567$ more students.yaml
'701':
  dob: 01/01/2000
  name: alex
  suburb: toowong
'702':
  dob: 05/05/2005
  name: josh
  suburb: milton
```



```
(uqcit) tor@l7480:~/uqcit/programming/project/rego-s1234567$ python rego.py

#########################################################################
The Rego app: MENU
[1] Add clubs
[2] Add students
[3] Register
[4] Print CSV
[5] Interactive Analysis

Menu choice (To EXIT, enter 0)? 2
>>>>>> Entering student info (To EXIT, enter ID= 0) <<<<<<<<
Enter the student ID? 703
Enter the student name? sara
Enter the student DOB: dd/mm/yyyy? 07/07/2007
Enter the student suburb? stlucia
>>>>>> Entering student info (To EXIT, enter ID= 0) <<<<<<<<
Enter the student ID? 0
BACK to the Menu!
writing students.yaml...

#########################################################################
The Rego app: MENU
[1] Add clubs
[2] Add students
[3] Register
[4] Print CSV
[5] Interactive Analysis

Menu choice (To EXIT, enter 0)? 0
The Rego app is exiting... BYE!
```



```
(uqcit) tor@l7480:~/uqcit/programming/project/rego-s1234567$ more students.yaml
'701':
  dob: 01/01/2000
  name: alex
  suburb: toowong
'702':
  dob: 05/05/2005
  name: josh
  suburb: milton
'703':
  dob: 07/07/2007
  name: sara
  suburb: stlucia
```

(Functionality 3) Menu number 3: Register (marks: 30/100)

Once selected (by entering number 3), this selection asks for four types of information, namely

a. sID: This refers to the student ID, represented as an integer.
b. cID: This refers to the club ID, represented as an integer.
c. Quantity: This refers to the number of sessions of this club that the student registers for. This is represented as an integer.

This menu handles multiple registrations of one student. That is, by first asking the student ID once, then Rego repeatedly asking about the club ID (cID) along with the Quantity, one after another in a loop. To terminate this input loop, the number 0 should be entered as CID.
Once registration is finished, Rego outputs a file named "receipt.txt" that contains the registration receipt for the student with the specified SID.

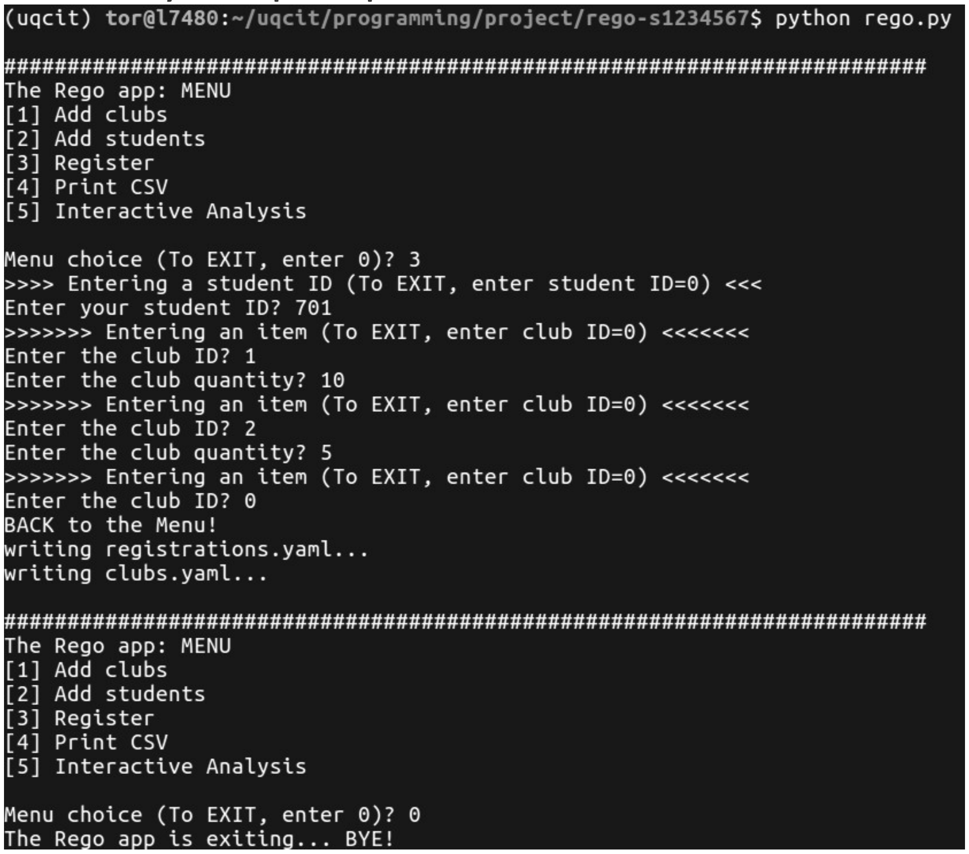
After the above three kinds of information are entered, this student information is added to the existing registration record, which is stored in a YAML format called "registrations.yaml". The updated registration record is finally stored again in the same file "registrations.yaml". Note that when Rego is launched for the first time, the file "registrations.yaml" does not exist. The file "registrations.yaml" will be created after the first registration.

Write code to implement Functionality 3 in a file called "registration.py".

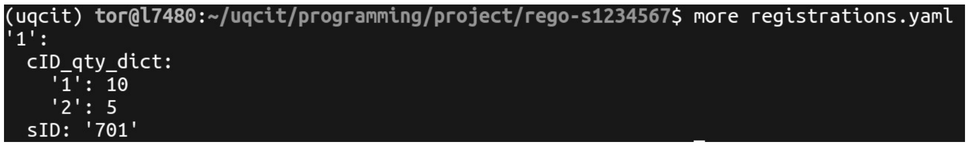
Functionality 3: breakdowns

a) Handling of creating the (currently non-existent) file “registrations.yaml” in the first launch.
b) Handling multiple club registrations for one student through the registration input loop.
c) Registration information inputs. This includes handling non-existent student and club IDs, and erroneous non-number quantity. For example: if Quantity is entered as “5x”, which cannot be converted to an integer. For such error cases, the current input is ignored (and discarded) then the registration loop continues.
d) Updating the registration record (from the file “registrations.yaml”) and saving the updated record to the same file “registrations.yaml”. The registration record is represented as a dictionary, where the keys are registration IDs, and the values are dictionaries containing the student ID and the club-quantity mapping. Here, the registration ID is automatically created starting from an ID number 1. The club-quantity mapping is a dictionary whose keys are club IDs, and values are the quantities of the sessions that the student registers for.
e) Printing the registration receipt containing the following information:
    a. School name
    b. Student name
    c. Timestamp: date and time
    d. A list of registered sessions along with their clubs and unit prices
    e. The total price of all registered sessions

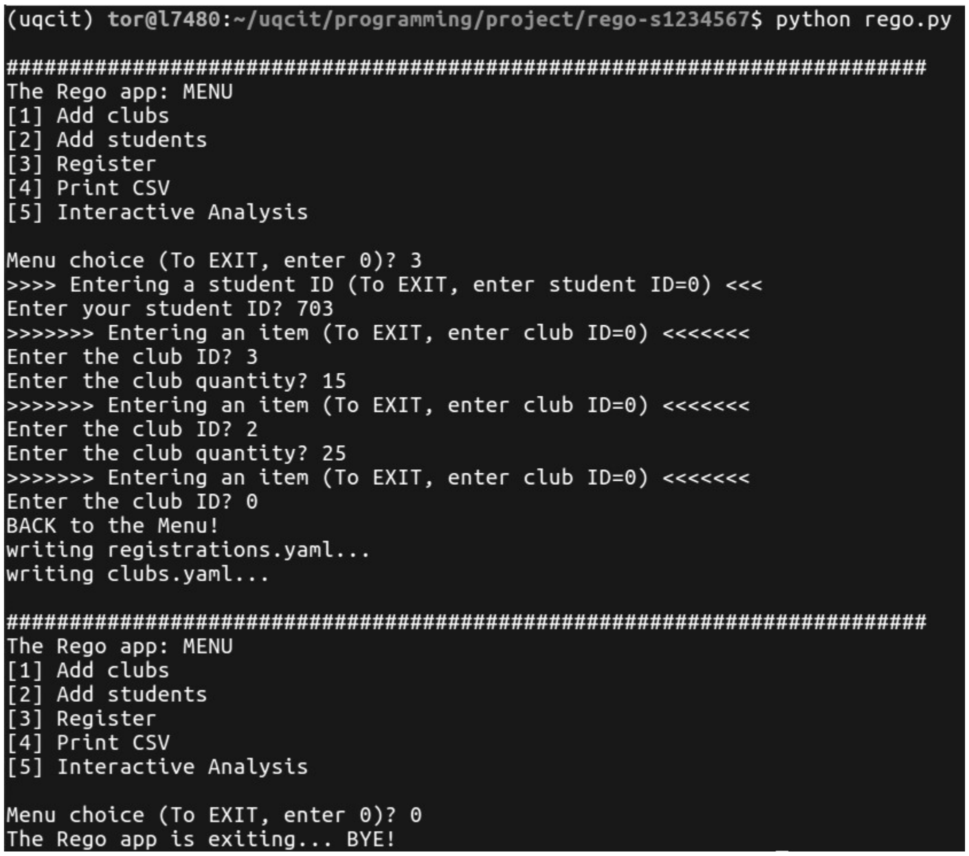
Functionality 3: sample outputs



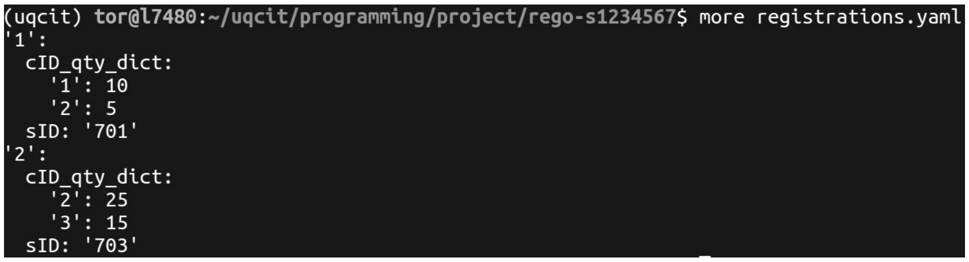
```
(uqcit) tor@l7480:~/uqcit/programming/project/rego-s1234567$ python rego.py

##########################################################################
The Rego app: MENU
[1] Add clubs
[2] Add students
[3] Register
[4] Print CSV
[5] Interactive Analysis

Menu choice (To EXIT, enter 0)? 3
>>>> Entering a student ID (To EXIT, enter student ID=0) <<<
Enter your student ID? 701
>>>>>>> Entering an item (To EXIT, enter club ID=0) <<<<<<<
Enter the club ID? 1
Enter the club quantity? 10
>>>>>>> Entering an item (To EXIT, enter club ID=0) <<<<<<<
Enter the club ID? 2
Enter the club quantity? 5
>>>>>>> Entering an item (To EXIT, enter club ID=0) <<<<<<<
Enter the club ID? 0
BACK to the Menu!
writing registrations.yaml...
writing clubs.yaml...

##########################################################################
The Rego app: MENU
[1] Add clubs
[2] Add students
[3] Register
[4] Print CSV
[5] Interactive Analysis

Menu choice (To EXIT, enter 0)? 0
The Rego app is exiting... BYE!
```



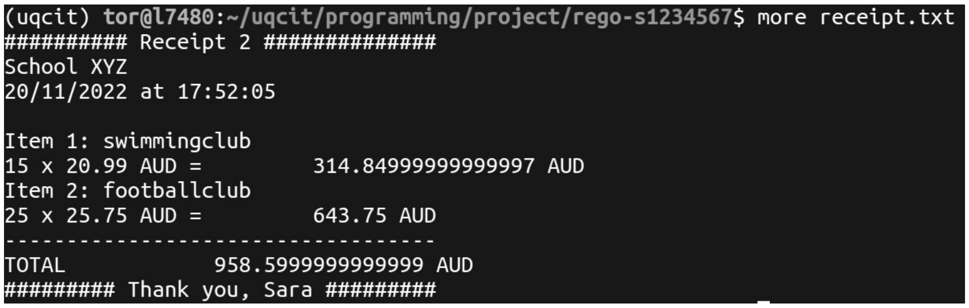
```
(uqcit) tor@l7480:~/uqcit/programming/project/rego-s1234567$ more registrations.yaml
'1':
  cID_qty_dict:
    '1': 10
    '2': 5
  sID: '701'
```



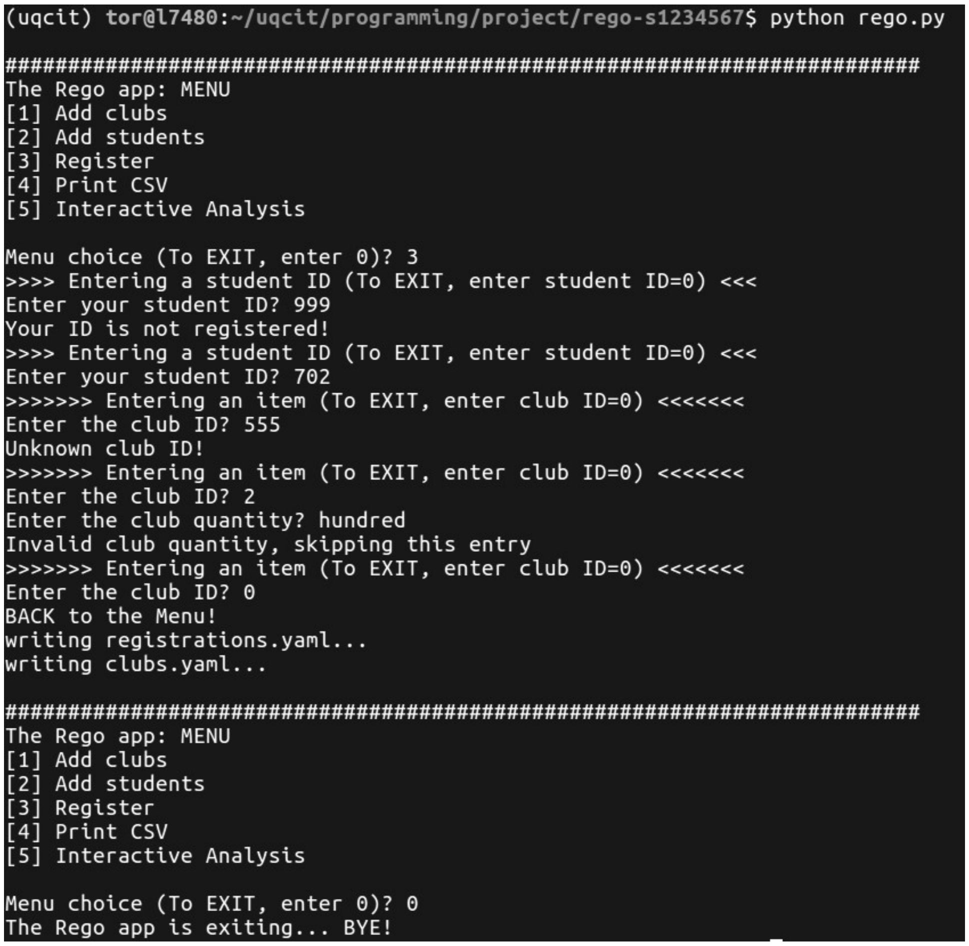
```
(uqcit) tor@l7480:~/uqcit/programming/project/rego-s1234567$ python rego.py

##########################################################################
The Rego app: MENU
[1] Add clubs
[2] Add students
[3] Register
[4] Print CSV
[5] Interactive Analysis

Menu choice (To EXIT, enter 0)? 3
>>>> Entering a student ID (To EXIT, enter student ID=0) <<<
Enter your student ID? 703
>>>>>>> Entering an item (To EXIT, enter club ID=0) <<<<<<<
Enter the club ID? 3
Enter the club quantity? 15
>>>>>>> Entering an item (To EXIT, enter club ID=0) <<<<<<<
Enter the club ID? 2
Enter the club quantity? 25
>>>>>>> Entering an item (To EXIT, enter club ID=0) <<<<<<<
Enter the club ID? 0
BACK to the Menu!
writing registrations.yaml...
writing clubs.yaml...

##########################################################################
The Rego app: MENU
[1] Add clubs
[2] Add students
[3] Register
[4] Print CSV
[5] Interactive Analysis

Menu choice (To EXIT, enter 0)? 0
The Rego app is exiting... BYE!
```



```
(uqcit) tor@l7480:~/uqcit/programming/project/rego-s1234567$ more registrations.yaml
'1':
  cID_qty_dict:
    '1': 10
    '2': 5
  sID: '701'
'2':
  cID_qty_dict:
    '2': 25
    '3': 15
  sID: '703'
```



```
(uqcit) tor@l7480:~/uqcit/programming/project/rego-s1234567$ more receipt.txt
########## Receipt 2 ##############
School XYZ
20/11/2022 at 17:52:05

Item 1: swimmingclub
15 x 20.99 AUD =        314.84999999999997 AUD
Item 2: footballclub
25 x 25.75 AUD =        643.75 AUD
-----------------------------------
TOTAL            958.5999999999999 AUD
######### Thank you, Sara #########
```



```
(uqcit) tor@l7480:~/uqcit/programming/project/rego-s1234567$ python rego.py

#########################################################################
The Rego app: MENU
[1] Add clubs
[2] Add students
[3] Register
[4] Print CSV
[5] Interactive Analysis

Menu choice (To EXIT, enter 0)? 3
>>>> Entering a student ID (To EXIT, enter student ID=0) <<<
Enter your student ID? 999
Your ID is not registered!
>>>> Entering a student ID (To EXIT, enter student ID=0) <<<
Enter your student ID? 702
>>>>>>> Entering an item (To EXIT, enter club ID=0) <<<<<<<
Enter the club ID? 555
Unknown club ID!
>>>>>>> Entering an item (To EXIT, enter club ID=0) <<<<<<<
Enter the club ID? 2
Enter the club quantity? hundred
Invalid club quantity, skipping this entry
>>>>>>> Entering an item (To EXIT, enter club ID=0) <<<<<<<
Enter the club ID? 0
BACK to the Menu!
writing registrations.yaml...
writing clubs.yaml...

#########################################################################
The Rego app: MENU
[1] Add clubs
[2] Add students
[3] Register
[4] Print CSV
[5] Interactive Analysis

Menu choice (To EXIT, enter 0)? 0
The Rego app is exiting... BYE!
```

(Functionality 4) Menu number 4: Print CSV (score: 10/100)

Once selected (by entering number 4), this selection outputs three CSV files, namely

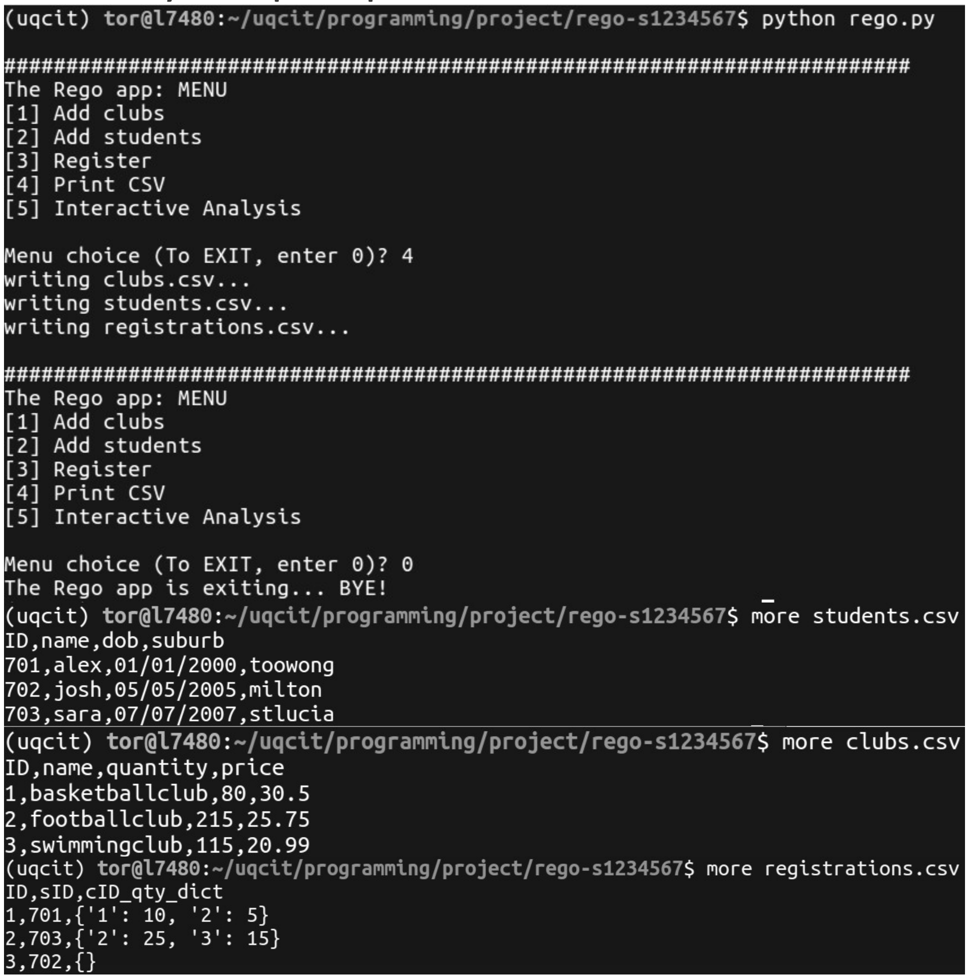
a. "clubs.csv": This contains the current information about the clubs, namely ID, name, quantity, price, which form the CSV column names (written in the first line). Note that the quantity column contains the current number of available sessions.
b. "students.csv": This contains the information about the students, namely ID, name, DOB, and suburb, which form the CSV column names (written in the first line).
c. "registrations.csv": This contains the current information about registrations, namely registration ID, sID, and cID-to-quantity (which is the mapping from cID to session quantities). These three entities form the CSV column names (written in the first line).

Write code to implement Functionality 4 in a file called "recorder.py".

Functionality 4: breakdowns

a) Printing/writing a CSV file for the club record.
b) Printing/writing a CSV file for the student record
c) Printing/writing a CSV file for the registration record

Functionality 4: sample outputs


```
(uqcit) tor@l7480:~/uqcit/programming/project/rego-s1234567$ python rego.py

##########################################################################
The Rego app: MENU
[1] Add clubs
[2] Add students
[3] Register
[4] Print CSV
[5] Interactive Analysis

Menu choice (To EXIT, enter 0)? 4
writing clubs.csv...
writing students.csv...
writing registrations.csv...

##########################################################################
The Rego app: MENU
[1] Add clubs
[2] Add students
[3] Register
[4] Print CSV
[5] Interactive Analysis

Menu choice (To EXIT, enter 0)? 0
The Rego app is exiting... BYE!
(uqcit) tor@l7480:~/uqcit/programming/project/rego-s1234567$ more students.csv
ID,name,dob,suburb
701,alex,01/01/2000,toowong
702,josh,05/05/2005,milton
703,sara,07/07/2007,stlucia
(uqcit) tor@l7480:~/uqcit/programming/project/rego-s1234567$ more clubs.csv
ID,name,quantity,price
1,basketballclub,80,30.5
2,footballclub,215,25.75
3,swimmingclub,115,20.99
(uqcit) tor@l7480:~/uqcit/programming/project/rego-s1234567$ more registrations.csv
ID,sID,cID_qty_dict
1,701,{'1': 10, '2': 5}
2,703,{'2': 25, '3': 15}
3,702,{}
```

(Functionality 5) Menu number 5: Interactive Analysis (score: 10/100)

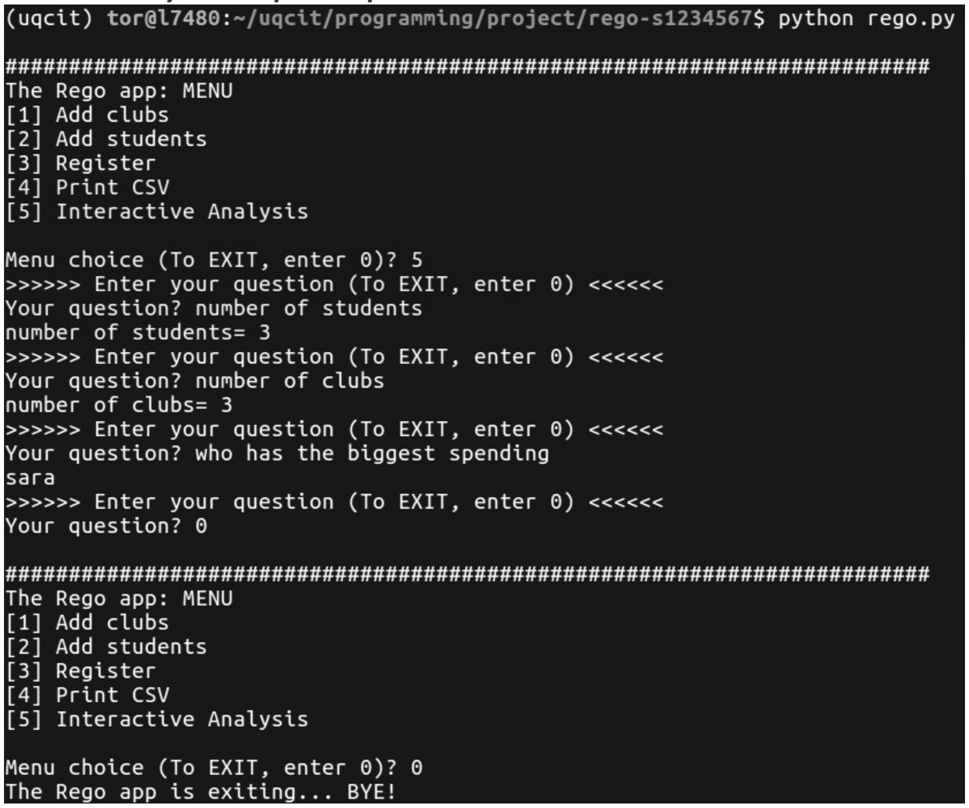
Once selected (by entering number 5), this selection provides an interactive analysis. It is interactive in that the user can type a limited number of questions in the prompt. Rego will then display the corresponding answers. The analysis is about students, clubs and registrations.

Write code to implement Functionality 5 in a file called "analysis.py".

Functionality 5: breakdowns

a) Handling the question of "number of clubs" (case-insensitive) by displaying the correct number of clubs in the club record.
b) Handling the question of "number of students" (case-insensitive) by displaying the correct number of students in the student record.
c) Handling the question of "who has the biggest spending" (case-insensitive) by displaying the name of student(s) who has the biggest spending for the club registration
d) Handling the input number 0 for terminating this menu selection
e) Handling multiple questions through a question loop
f) Handling unknown questions by continuing the question loop

Functionality 5: sample outputs



```
(uqcit) tor@l7480:~/uqcit/programming/project/rego-s1234567$ python rego.py

##########################################################################
The Rego app: MENU
[1] Add clubs
[2] Add students
[3] Register
[4] Print CSV
[5] Interactive Analysis

Menu choice (To EXIT, enter 0)? 5
>>>>>> Enter your question (To EXIT, enter 0) <<<<<<
Your question? number of students
number of students= 3
>>>>>> Enter your question (To EXIT, enter 0) <<<<<<
Your question? number of clubs
number of clubs= 3
>>>>>> Enter your question (To EXIT, enter 0) <<<<<<
Your question? who has the biggest spending
sara
>>>>>> Enter your question (To EXIT, enter 0) <<<<<<
Your question? 0

##########################################################################
The Rego app: MENU
[1] Add clubs
[2] Add students
[3] Register
[4] Print CSV
[5] Interactive Analysis

Menu choice (To EXIT, enter 0)? 0
The Rego app is exiting... BYE!
```

Submission instructions

1. Create a directory (folder) called “rego-s1234567”, where s1234567 is your student number.
2. Put all Python source-code files (see below) in the directory “rego-s1234567” (created in Step 1). There is no limitation on the number of Python source-code files that can be submitted, but at least there are six files as follows
   a. rego.py
   b. club.py
   c. student.py
   d. registration.py
   e. recorder.py
   f. analysis.py
3. Zip the directory “rego-s1234567” into a zipped file called “rego-s1234567.zip”.
4. Submit “rego-s1234567.zip” (created in Step 3) into the dedicated portal in Blackboard.