FIT9136 Algorithm and programming foundation in Python

Assignment 2

Lecturer in Charge: Shirin Ghaffarian Maghool

April 2022

Table of Contents

1. Key Information

Purpose	This assessment is related to the following learning objectives (LO): ● **LO2:** Restructure a computational program into manageable units of modules and classes using the object-oriented methodology ● **LO3:** Demonstrate Input/Output strategies in a Python application and apply appropriate testing and exception handling techniques
Your task	It is an Individual assignment where you will write a Python code for simple emulation of data processing. Your application should be able to ask the user to input a username, password to login into your system. The system should be able to identify the role of this user and allow corresponding operations.
Value	**30**% of your total marks for the unit
Due Date	**[Friday 6th May 2022, week 9] 4:30 pm**
Submission	● Via Moodle Assignment Submission. ● Turnitin will be used for similarity checking of all submissions.
Assessment Criteria	See Moodle Assessment page
Late Penalties	● 10% deduction of the available mark per calendar day or part thereof for up to one week ● Submissions more than 7 calendar days after the due date will receive a mark of zero (0) and no assessment feedback will be provided.
Support Resources	See Moodle Assessment page
Feedback	Feedback will be provided on student work via: general cohort performance specific student feedback ten working days post submission

1.1. Do and Do NOT

Do	Do NOT
● Maintain academic integrity[1] ● Get support early from this unit and other services in the university ● Apply for special consideration for extensions if needed (optional)[2] ● Attend an interview (No attendance = 0 for Assignment 2)	● Leave your assignment in draft mode ● Submit late (10% daily penalty applies)[3] ● Submission is not accepted after 7 days of the due date, unless you have special consideration.

1.2. Documentation

Commenting your code is essential as part of the assessment criteria (refer to Marking Rubrics).

You should also include comments at the beginning of your program file, which specify your name, your Student ID, the start date and the last modified date of the program, as well as with a high-level description of the program. In-line comments within the program are also part of the required documentation.

1.3. Submission

Your have to submit your assignment via the assignment submission link (i.e., "Assignment 1 Submission") on the Moodle site by the deadline specified in Section 1, i.e. **6th May 2022 (Friday) by 4:30 pm**:

- There will be NO hard copy submission required for this assignment.

- You are required to submit your assignment as a .zip file named with your Student ID. For example, if your Student ID is 12345678, you would submit a zipped file named 12345678.zip

- The submission files are to be submitted and must exist in your FITGitLab server repo (i.e, Assessments/Assignment02), which shows your development history of **at least three** pushes.

---

[1] https://www.monash.edu/rlo/research-writing-assignments/referencing-and-academic-integrity/academic-integrity

[2] https://www.monash.edu/exams/changes/special-consideration (**All the Special Consideration should be applied no later than two University working days after the due date of the affected assessment**).

[3] eg: original mark was 70/100, submitting 2 days late results in 50/100 (10 marks off). This includes weekends

- Do not include any unnecessary file in this folder
- Note that marks will be deducted if this requirement is not strictly complied with.
- No submission is accepted via email.
- Please note we **cannot mark any work on the FITGitLab Server**. You need to ensure that you submit via Moodle where you need to accept the student declaration. It is your responsibility to **ENSURE** that the submitted files are the correct files. We strongly recommend after uploading a submission, and prior to actually submitting in Moodle, that you download the submission and double-check its contents. Marks will be deducted for any of the requirements that are not complied with.

1.4. Deliverables

Your submission should contain the following documents:

- `Course.py, Review.py, User.py, Admin.py, Instructor.py, Student.py and Main.py files in one zipped file.`

- **Marks will be deducted for any of these requirements that are not strictly complied with.**

2. Getting help

2.1. English language skills
if you don't feel confident with your English.

- Talk to English Connect: https://www.monash.edu/english-connect

2.2. Study skills
If you feel like you just don't have enough time to do everything you need to, maybe you just need a new approach

- Talk to a learning skills advisor:
  https://www.monash.edu/library/skills/contacts

2.3. Things are tough right now
Everyone needs to talk to someone at some point in their life, no judgement here.

- Talk to a counsellor:
  https://www.monash.edu/health/counselling/appointments
  (friendly, approachable, confidential, free)

2.4. Things in the unit don't make sense
Even if you're not quite sure what to ask about, if you're not sure you won't be alone, it's always better to ask.

- Ask questions in Ed: https://edstem.org/au/courses/7429/discussion/
- Attend a consultation:
  https://lms.monash.edu/course/view.php?id=135703&section=21

2.5. I don't know what I need
Everyone at Monash University is here to help you. If things are tough now, they won't magically get better by themselves. Even if you don't exactly know, come and talk with us, and we'll figure it out. We can either help you ourselves or at least point you in the right direction.

3. Key tasks (100 marks)

This assignment is a simple emulation of data processing. Your application should be able to ask the user to input username, password to login into your system. The system should be able to identify the role of this user and allow corresponding operations.

This is an individual assignment and must be your own work.

The allowed libraries are **random, math, os and re**. You will not receive marks for the components if you use any other libraries apart from the mentioned library.

Please finish reading the description of all classes before starting the implementation.

3.1. User class

User class handles the fundamental methods of all users. The User class is the parent class of Admin, Instructor and Student classes.

3.1.1 constructor

A user must have id(int, default value -1), username(str, default value ""), password(str, default value "").

3.1.2 generate_unique_user_id()

This method checks the files user_admin.txt, user_instructor.txt and user_student.txt to generate an unique user id. The return result is a 10 digits integer.

3.1.3 encryption(input_password)

This method encrypts the input_password to a string that is difficult to read by humans. Reuse the encryption algorithm that was in A1 to encode the input password. The encrypted password will be returned and the type is string.

3.1.4 login()

Each user can call the login method to perform authentication. In this login() method, it is required to call the encryption() method defined before to encode the password. The encoded password can be used to compare with the password extracted from files. You are required to to check the user_admin.txt, user_instructor.txt and user_student.txt file according to the username and password and return a tuple which contains the login_result(bool type), login_user_role(str type, the values can only be "Admin", "Instructor", "Student"), login_user_info(any type e.g. list or tuple

or str, this value can be used to create different types of user object (Admin, Student or Instructor)).

3.1.5 extract_info()

This method prints out a message "You have no permission to extract information".

3.1.6 view_courses(args=[])

This method prints out a message "You have no permission to view courses".

3.1.7 view_users()

This method prints out a message "You have no permission to view users".

3.1.8 view_reviews(args=[])

This method prints out a message "You have no permission to view reviews".

3.1.9 remove_data()

This method prints out a message "You have no permission to remove data".

3.1.10 __str__()

This method returns a formatted user string: "user_id;;;username;;;password". All the attributes are concatenated using ";;;".

3.2. Admin class

Admin class inherits from the User class.

3.2.1 constructor

Admin only have attributes id(int, default value -1), username(str, default value "") and password(str, default value "") which can be inherited from the parent class.

3.2.2 register_admin()

This method checks the user_admin.txt file to find out whether the username already exists or not. If not, register this admin. If it exists, do nothing.

3.2.3 extract_course_info()

This method can get course information from the raw_data.txt. The extracted course info should be saved into file following the format below:

`"course_id;;;course_title;;;image_100x100;;;headline;;;num_of_subscribers;;;avg_rating;;;course_content_length"`.

For each line in the raw_data.txt file, you can copy and paste it to [Json Parser Online](#) to check the format. Each line contains more than one course. All the corresponding attributes can be found in the text. You can use the library **re** or str methods to extract the data. The course content length in the text is like "40.5 hours". Only 40.5 need to be retrieved. The course data will be saved into the course.txt file.

3.2.4 extract_review_info()

This method can get review information from the review_data folder. The extracted review info saving format is
`"review_id;;;review_content;;;review_rating;;;course_id"`
The course id can be obtained from each file's name in the review_data folder.

3.2.5 extract_students_info()

This method can get student information from the review_data folder. Each review string contains one user information. Assume each user only has one review. The attributes of each student are id, username, password, user_title, user_image_50x50, user_initials and review_id. If a student's id cannot be found in the string, you need to generate it by calling the generate_unique_user_id() method defined in the User class. The username should be generated by converting the user_title to lowercase and replacing all the whitespace to underscore. The password is generated by converting user_initials to lowercase and combining the user id. The password expression looks like "user_initials + user_id + user_initials". User_initials can be obtained from the review data. For example, user_id is 12345, user initials is "DE", the password is "de12345de". The student info will be written into file user_student.txt and the format example is:
`"id;;;username;;;password;;;user_title;;;user_image_50x50;;;user_initials;;;review_id"`.

3.2.6 extract_instructor_info()

This method extracts information from the raw_data.txt file in the course_data folder. Each course item contains several instructor information. The username is generated by converting the instructor_display_name to lowercase and replacing all the whitespace to underscore. The password uses the instructor id directly. The instructor info saving format example is:
`"id;;;username;;;password;;;display_name;;;job_title;;;image_100x100;;;course_id1-course_id2-course_id3-course_id4"`.

Since each instructor can teach more than one course, before saving the instructor information, you need to check whether the instructor already exists in the user_instructor.txt file or not. If it exists, the course id should be appended to the existing instructor string. For example, given an instructor string:

```
4466306;;;colt_steele;;;***4---***4---***6---***6---***3---***0---***6---;;;
Colt Steele;;;Developer and Bootcamp Instructor;;;https://img-c.udemycdn.com
/user/100x100/4466306_6fd8_3.jpg;;;625204
```

If a new course id 55555 is teached by this instructor, the string will be updated as below:

```
4466306;;;colt_steele;;;***4---***4---***6---***6---***3---***0---***6---;;;
Colt Steele;;;Developer and Bootcamp Instructor;;;https://img-c.udemycdn.com
/user/100x100/4466306_6fd8_3.jpg;;;625204--55555
```

The format of instructor data will be explained in the Instructor class section.

3.2.7 extract_info()

This method calls all the extract info methods defined before, including extract_course_info(), extract_instructor_info(), extract_students_info() and extract_review_info().

3.2.7 remove_data()

This method can delete all the data in the course.txt, review.txt, user_student.txt and user_instructor.txt files.

3.2.8 view_courses(args=[])

This method will call the methods implemented in Course class to perform various view course methods. The variable "args" can be an empty list or must contain two elements. The first element is the command(can only be "TITLE_KEYWORD/ID/INSTRUCTOR_ID") and the second element is the value(could be title keyword, course id or instructor id). For example, args=["TITLE_KEYWORD", "web"]. If the "args" is an empty list, printing out the overview of courses (using the course_overview method implemented in Course class). Add validations to provide proper output message if user input incorrect command or values.

3.2.9 view_users()

This method prints out the total number of admin, instructor and students separately with proper description messages.

3.2.10 view_reviews(args=[])

This method will call the methods implemented in Review class to perform various view review methods. The args list can be empty or must contain two elements. The first element is the command(can only be "ID/KEYWORD/COURSE_ID") and the second element is the value(could be review id, review keyword or course id). If the args is an empty list, print out the overview of reviews(using the review_overview method implemented in Review class); Add validations to provide proper output message if user input incorrect command or values.

3.2.11 __str__()

Since the Admin class does not have any instance variables different from the User class, you can use the parent class's __str__ method directly.

3.3. Instructor class

The Instructor class inherits from the User class.

3.3.1 constructor

The attributes of instructor are id(int, default value -1), username(str, default value ""), password(str, default value ""), display_name(str, default value ""), job_title(str, default value ""), image_100x100(str, default value ""), course_id_list(list, default value []).

3.3.2 view_courses(args=[])

Print out all the courses taught by this instructor. This can be reached by calling the method implemented in the Course class. If the number of courses is greater than 10, only print 10 courses. The args positional argument is not used in this method.

3.3.3 view_reviews(args=[])

Print out all the reviews belong to the courses this instructor teaches. This can be achieved by calling the method implemented in the Course class. If the number of reviews is greater than 10, only print 10 reviews. Also print out the total number of all reviews. The args positional argument is not used in this method.

3.3.4 __str__()

Use the Parent class __str__ method. The return result format is: "`user_id;;;username;;;password;;;display_name;;;job_title;;;image_100x100;;;123123-323-32-3123-3123`".

3.4. Student class

Student class inherits from the User class.

3.4.1 constructor

The attributes of student are id(int, default value -1), username(str, default value ""), password(str, default value ""), user_title(str, default value ""), user_image_50x50(str, default value ""), user_initials(str, default value ""), review_id(int, default value -1).

3.4.2 view_courses(args=[])

This method prints out the course this student registered. If a student reviews a course, it is assumed that the student is registered in that course. The args positional argument is not used in this method. Just have it when doing method declaration.

3.4.3 view_reviews(args=[])

This method prints out the review this student wrote. The args positional argument is not used in this method. Just have it when doing method declaration.

3.4.4 __str__()

Use the parent class __str__ method. The return result format is:"`user_id;;;username;;;password;;;title;;;image_50x50;;;initials;;;review_id`".

3.5. Course class

3.5.1 constructor

The attributes of course are course_id(int, default value -1), course_title(str, default value ""), course_image_100x100(str, default value ""), course_headline(str, default

value ""), course_num_subscribers(int, default value -1), course_avg_rating(float, default value -1.0), course_content_length(float, default value -1.).

3.5.2 find_course_by_title_keyword(keyword)

This method has a positional argument keyword(str). Based on the given keyword, it searches the course title of all courses in the course.txt file to find the result. All the result courses will be created as a course object and added into a result list. Finally, return the result list. The result list looks like [Course(), Course(), Course()....]. If not found, return an empty list.

3.5.3 find_course_by_id(course_id)

This method has a positional argument course id(int or str). You are required to search for the course according to the course id. A course object will be returned. If not found, return None.

3.5.4 find_course_by_instructor_id(instructor_id)

This method has a positional argument instructor id(int or str). Based on the instructor id, a list of course objects will be generated and returned. The result list looks like [Course(), Course(), Course()....]. If not found, return an empty list.

3.5.5 courses_overview()

This method returns a string that shows the total number of courses.

3.5.6 __str__()

Return a string containing all course information.

3.6. Review class

3.6.1 constructor

The attributes of review are id(int, default value -1), content(str, default value ""), rating(float, default value -1.0), course_id(int, default value -1).

3.6.2 find_review_by_id(review_id)

This method has a positional argument review id(int or srt). It finds the review information based on the given id from the review.txt file. A review object will be returned. If not found, return None.

3.6.3 find_review_by_keywords(keyword)

This method has a positional argument keyword(str). It finds the reviews whose content contains the given keyword from the review.txt file. A list of review objects will be generated and returned. The result list looks like [Review(), Review(), Review()….]. If not found, return an empty list. The keyword cannot be empty.

3.6.4 find_review_by_course_id(course_id)

This method has a positional argument course id(int or str). It finds the reviews that belong to the given course id. A list of review objects will be generated and returned. The result list looks like [Review(), Review(), Review()….]. If not found, return an empty list.

3.6.5 reviews_overview()

This method returns a string that shows the total number of reviews.

3.6.6 __str__()

Return a string containing all review attributes.

3.7. Main test file

3.7.1 show_menu()

This method prints out the available options that the user can choose. You can add positional arguments if you need. Fig1 shows an example of the menu output.

```
Welcome to our system
Please input username and password to login:(format username password)
admin admin
Admin login successfully
Welcome admin. Your role is Admin.
Please enter Admin command for further service:
1. EXTRACT_DATA
2. VIEW_COURSES
3. VIEW_USERS
4. VIEW_REVIEWS
5. REMOVE_DATA
```

Fig1 show menu example

3.7.2 process_operations(user_object)

This method has one positional argument user_object.
Admin can take commands "1", "2", "3", "4", "5". For command "2" and "4", the end user needs to enter "TITLE_KEYWORD/ID/INSTRUCTOR_ID" and "ID/KEYWORD/COURSE_ID" as a second command, followed by a value. For example, if the login user is admin, he can input "2 TITLE_KEYWORD web"(the value after the second command can only be one word). The system will call corresponding view course methods and print out all the courses whose title contains "web" (case insensitive). If the user does not enter any other arguments for command "2" and "4", a default overview output will be displayed.

Instructor and Student can take command "2", "4" and no other arguments are allowed. If the logged user is instructor or student and the command is "1", "3", "5", the inherited methods in User class will be executed.

When a user input "logout", logout the user but the system keeps running.

In this method, it is **not allowed** to create an object of Admin/Instructor/Student class. It is assumed that a User class object user_object is passed into this method and you are not sure about the exact type.

3.7.2 main()

This method handles the main logic of your system. No positional arguments here. It should keep running until the user input "exit". At the beginning, this method asks the user to enter username and password in format "username password". Next,
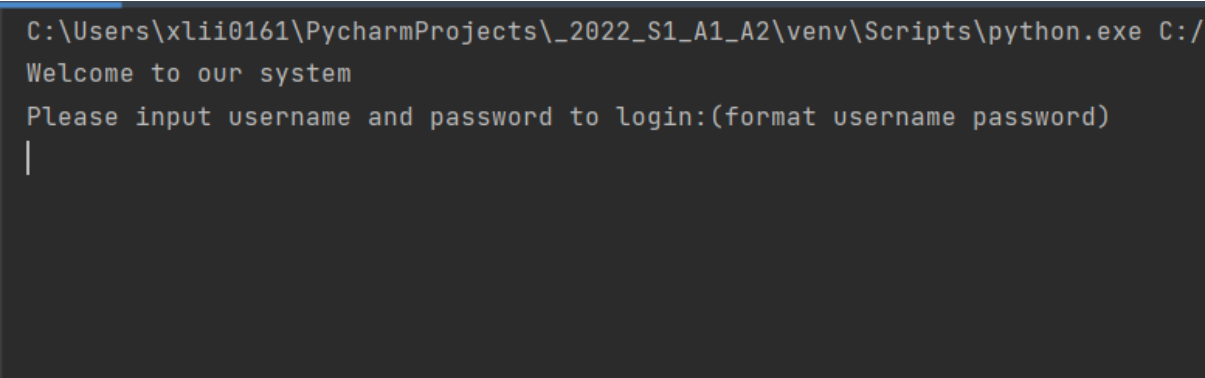
create a temp_user object and only assign the username and password to this object. Then you can use temp_user to call the login() method. Based on the login result, a corresponding user object can be created (Admin(), Instructor() or Student()). For different user objects, call the same process_operations(user_object) method to process all the logics. If the logged user is "Admin", 1) create an Admin object, 2) call show_menu() method, 3) call process_operations() method. If the login fails, print out an error message. If the username password format is incorrect, print out a different error message.

3.7.3 if __name__ == "__main__":

This part is the entry point of the whole program. In code part, 1. output a message to indicate the program start. 2. Manually register an admin account(call register_admin() method). 3. Call the main() method. When marking your assignment, we will run this file only.

3.8. Demo of program

1. Start the program, manually register an admin account in your test code.


```
C:\Users\xlii0161\PycharmProjects\_2022_S1_A1_A2\venv\Scripts\python.exe C:/
Welcome to our system
Please input username and password to login:(format username password)
|
```

Fig2 input username and password example.

2. After enter the admin username and password

17

Fig3 login success example

Fig4 login failure example

3. Input logout

Fig5 logout example

4. Login again and enter command.

Fig6 enter command example

Fig7 search with valid arguments example

```
Admin login successfully
Welcome admin. Your role is Admin.
Please enter Admin command for further service:
1. EXTRACT_DATA
2. VIEW_COURSES
3. VIEW_USERS
4. VIEW_REVIEWS
5. REMOVE_DATA
2 ejajfd;l qewr
The total number of course is 2942
```

Fig9 search with invalid arguments example

Important Notes:

- If any exception/errors happens when running your program, you will lose 50% of allocated method/class marks. For example, if the login functionality returns any error, then the maximum mark you can get is 5% instead of 10% in the login.

- Add proper validations and output messages to make your code more user friendly to end users.

- Run the test method at the end of your file to show that your program works fine.