Programming Fundamentals (COSC2531) Assignment 1

Assessment Type	**Individual assignment** (no group work). Submit online via Canvas/Assignments/Assignment 1. Marks are awarded per rubric (please see the rubric on Canvas). Clarifications/updates may be made via announcements. Questions can be raised via the lectorial, practical sessions or Canvas discussion forum. Note the Canvas discussion forum is preferable.
Due Date	**End of Week 6** (exact time is shown in Canvas/Assignments/Assignment 1) Deadline will not be advanced, but they may be extended. Please check Canvas/Assignments/Assignment 1 for the most up to date information regarding the assignment. As this is a major assignment, a university standard late penalty of 10% per each day (e.g., 2 marks/day) applies, unless special consideration has been granted.
Weighting	**20 marks out of 100**

1. Overview

The objective of this assignment is to develop your programming and problem-solving skills in a step-by-step manner. The different stages of this assignment are designed to gradually introduce different basic programming concepts.

You should develop this assignment in an iterative fashion (as opposed to completing it in one sitting). You can and should get started now (when this assignment specification is posted on Canvas) as there are concepts from previous lessons that you can employ to do this assignment. If there are questions, you can ask via the lectorial, practical sessions or the Canvas discussion forum (Canvas/Discussions/Discussion on Assignment 1). Note that the **Canvas discussion forum is preferable** as it allows other students to see your questions as well. Also, you should ask questions in a general manner, for example, you should replicate your problem in a different context in isolation before posting, and **you must not post your code on the Canvas discussion forum**.

2. Assessment Criteria

This assignment will determine your ability to:

i. Follow coding, convention and behavioural requirements provided in this document and in the course lessons;
ii. Independently solve a problem by using programming concepts taught over the first several weeks of the course;
iii. Write and debug Python code independently;

- iv. Document code;
- v. Provide references where due;
- vi. Meet deadlines;
- vii. Seek clarification from your "supervisor" (instructor) when needed via the Canvas discussion forums; and
- viii. Create a program by recalling concepts taught in class, understand, and apply concepts relevant to solution, analyse components of the problem, evaluate different approaches.

3. Learning Outcomes

This assignment is relevant to the following Learning Outcomes:

1. Analyse simple computing problems.
2. Devise suitable algorithmic solutions and code these algorithmic solutions in a computer programming language.
3. Develop maintainable and reusable solutions.

Specifically, upon the completion of this assignment, you will be able to:

- Demonstrate knowledge of basic concepts, syntax, and control structures in programming
- Devise solutions for simple computing problems under specific requirements
- Encode the devised solutions into computer programs and test the programs on a computer
- Demonstrate understanding of standard coding conventions and ethical considerations in programming

4. Assessment Details

Please ensure that you have read Sections 1-3 of this document before going further.

Problem Overview: In this assignment, you are developing a system for a book rental service. The receptionists are the ones that use this system to process and print out receipts of the customers' rentals. You are required to implement the program following the below requirements.

Requirements: Your code must meet the following **functionalities**, **code**, and **documentation** requirements. Your submission will be graded based on the **rubric** published on Canvas. Please ensure you read all the requirements and the rubric carefully before working on your assignment.

A - Functionalities Requirements:
There are 3 parts; please ensure you only attempt one part after completing the previous part.

-- PART 1 (6 marks) --

In this part, your program can perform some simple interactions with users (i.e., the receptionists):

1. Display a message asking the user to enter the customer's name. In this part, you can assume the customer's name to be entered only consists of alphabet characters.
2. Display a message asking the user to enter the name of the book the customer wants to rent. In this part, you can assume the book to be entered is always a valid book.
3. Display a message asking the number of days the customer wants to borrow. In this part, you can assume the number of days to be entered is always a positive integer, e.g., 1, 2, 3 …

4. Calculate the total rental price for the customer. The total rental price is equal to *the original total rental price – the discount*. The *original total rental price* of a book is equal to the book's rental price per day multiplying with the number of borrowing days. Note that there are two levels of rental price for each book (the 2nd rental price is always less than the 1st rental price). If the number of borrowing days is less than or equal 10 days, then the 1st rental price is used, otherwise, the 2nd rental price is used. For example, if the book is *Harry Potter 1*, the two rental prices of this book are *0.5$* and *0.4$*, respectively, then if the number of borrowing days is *5*, then the original total rental price is *2.5$*. For the *discount*, see No. 5 below.
5. A member will have a discount of 10% over the original total cost. For example, if the original total cost is *2.5$*, then the discount will be *0.25$*. A customer can register to be a member **after** their first rental, i.e., after the first rental, the program will ask if the customer would like to become a member. It's free to become a member. Note the discount is not applicable for the first rental (before becoming a member).
6. All the rental information will be displayed as a formatted message to the user as follows. Note that the rental price per day, all the costs, and the discount are all displayed with two digits after the decimal point.

```
---------------------------------------------------------------------------------
Receipt for <customer_name>
---------------------------------------------------------------------------------
Books rented:
   - <book_name> for <number_days> days (<price_per_day> AUD/day)
---------------------------------------------------------------------------------
Original cost:      <original_cost> (AUD)
Discount:           <discount> (AUD)
Total cost:         <total_cost> (AUD)
```

7. In the program, you should have some lists (or dictionaries or other data types) to store the names of all customers, members, the book categories, the books, the rental prices per day of the books. You can assume the customer names, the book names, and the book categories are all unique and case sensitive.
8. When a new customer finishes a rental, your program will automatically add the customer's name to the customer list. If the customer agrees to become a member, your program will add the customer to the member list.
9. Your program needs to be initialized with the following existing customers: *Emily* and *James*, with Emily being a member and James not being a member.
10. Your program will be initialized with the following book categories, books, and rental prices per day. All books belonging to the same book category have the same rental prices per day.

Book category	**Books**	**Rental prices per day**
Fantasy	Harry Potter 1, The Hobbit	0.5 / 0.4
Crime	Gone Girl, Sherlock Holmes 1	0.5 / 0.4
Classics	Pride and Prejudice	0.3 / 0.25
Modern Classics	To Kill a Mockingbird	0.4 / 0.3
History	The Diary of a Young Girl	0.4 / 0.3
Philosophy	The Republic	0.3 / 0.25
Science	A Brief History of Time	0.5 / 0.4

Textbooks	Introduction to the Theory of Computation	0.75 / 0.6
Art	The Story of Art	0.5 / 0.4
Other	Thinking Fast and Slow, Atomic Habits	0.5 / 0.4

11. Note: in the requirements No. 7, we use the term 'list' when describing the customer list, the book category list, the book list, etc, but you can use other data types to store this information such as dictionaries and other data types. Make sure you think and analyse the requirements in detail so that you can choose the most appropriate/suitable data types.

---------- PART 2 (6 marks, please do not attempt this part before completing PART 1) -----------

In this part, your program can: (a) perform some additional requirements compared to PART 1, and (b) be operated using a **menu**.

First, compared to the requirements in PART 1, now your program will be able to handle invalid inputs from users:

a. Display an error message if the customer's name entered by the user contains non-alphabet characters. When this error occurs, the user will be given another chance, until a valid name (names contain only alphabet characters) is entered.
b. Display an error message if the book entered by the user is not a valid book. When this error occurs, the user will be given another chance, until a valid book is entered.
c. Display an error message if the number of borrowing days entered is 0, negative, or not an integer. When this error occurs, the user will be given another chance, until a valid quantity is entered.
d. Each book category belongs to 2 types: *Rental* and *Reference*. All books belonging to the Rental type can have unlimited number of borrowing days whilst all books belonging to the Reference type can only be borrowed up to 14 days. Currently the book categories *Textbooks* and *Other* belong to the Reference type whilst other book categories belong to the Rental type. During the rental, if a book belongs to the Reference type and if the number of borrowing days is more than 14 days, then the program will display a message saying this book can't be borrowed for more than 14 days and ask the user to enter again until a valid quantity is entered (note the valid quantity still needs to satisfy the requirement c in the above).

Second, your program will be operated using a **menu**. A menu-driven program is a computer program in which options are offered to the users via the menu. Your program will have the following options: rent a book, update information of a book category, update books of a book category, display existing customers, display existing book categories, and exit the program (please see Section 5 in this document regarding an example of how the menu program might look like). Below are the specifications of the options:

1. *Rent a book*: this option includes all the requirements from 1 to 11 in PART 1 and the requirements a) to d) in the first part of PART 2.
2. *Update information of a book category*: this option displays a message asking the user for the information of a book category (type, two rental prices per day) to be updated. This information will be separated by commas and must be entered with the following format: *book_category, type, price_1, price*_2. For example, the user can enter *Fantasy, Rental, 0.4, 0.3* to update the book category Fantasy. The values entered (the type, prices) will replace the existing values of the book category. You can assume users always enter the correct format of the book category,

type, the prices but note that there could be multiple white spaces before and after the commas. You can also assume the book category, type, prices to be entered are valid values, i.e. the book category is an existing category, type is either Rental or Reference, and prices should be valid positive floating-point numbers.

3. *Update books of a book category:* this option first displays a message asking the user if they want to add (a) or remove (r) books from a book category, then displays a message asking for the book category and the list of books to be added or removed. If the user enters *a*, then the books will be added to the category; if the user enters *r*, then the books will be removed from the category. The book category and the list of books to be added/removed will be separated by commas with the following format: *book_category, book_1, book_2, ...* You can assume the user always enters *a* or *r*. You can assume the list of books to be added contain only new books. You can assume the list of booked to be removed are existing books from the category. You can assume user always enter the correct format of book_category, book_1, book_2, … but note that there could be multiple white spaces before and after the commas.
4. *Display existing customers*: this option displays on screen all existing customers and their membership. The messages are flexible (your choice).
5. *Display existing book categories*: this option displays on screen all existing book categories with all their information: type, rental prices, list of books. The messages are flexible (your choice).
6. *Exit the program*: this option allows users to exit the program.

Note that in your program, when a task (option) is accomplished, the menu will appear again for the next task.

----------- PART 3 (5 marks, please do not attempt this part before completing PART 2) ----------

In this part, your menu program is equipped with some advanced features. Note, some features maybe very challenging.

1. In this part, in the *"Rent a book"* option, your program will allow customers to rent multiple books. That is, after a book and the corresponding number of borrowing days are entered into the system, the user will be asked if another book is rented. If the answer by the user is *y* (meaning yes), then the system will ask for the book name and the number of borrowing days. The process will be repeated until the user answers *n* (meaning no). In this option, invalid inputs (books and number of borrowing days) need to be handled. The answer by the user should only be *y* or *n* – if the user enters a value that is different to *y* or *n*, they will be given another chance until a valid answer is entered. The program should be able to handle all the invalid inputs as in PART 2. The formatted message for the receipt is as follows.

```
-------------------------------------------------------------------------------------
Receipt for <customer_name>
-------------------------------------------------------------------------------------
Books rented:
   - <book1_name> for <number_days> days (<price_per_day> AUD)
   - <book2_name> for <number_days> days (<price_per_day> AUD)
   - <book1_name> for <number_days> days (<price_per_day> AUD)
                        ..........
-------------------------------------------------------------------------------------
Original cost:        <original_cost> (AUD)
```

```
Discount:        <discount> (AUD)
Total cost:      <total_cost> (AUD)
```

2. In this part, the program should be able to handle the invalid inputs in the option *Update information of a book category*. In particular, if the user enters the invalid book category, type or prices, the program should ask the user to enter all the information again.
3. In this part, the program should also be able to handle invalid inputs in the option *Update books of a book category* (including the answer *a* or *r*, the book category, and the list of books). If the user enters invalid answers (not a or r) or book category, the program should ask the user to enter again. For the list of books to be updated, for the adding option, if the user enters books that are existing books, the program should display the message saying these books are existing books so no adding to the data collection. Note that new books are still added to the data collection as usual. For the removing option, if the user enters books that are not existing books, the program should display the message saying these books are not existing books so no removing from the data collection. Other existing books are still removed from the data collection as usual.
4. The menu now has an option *"Display the most valuable customer"* to display the customer with the maximum money (total cost) spent to date and the amount of money they have spent. If there are multiple customers with the same maximum money spent, you can display only one customer or all customers, it's your choice.
5. The menu now has an option *"Display a customer rental history"*. This option will display a message asking the user to enter the name of the customer, and the program will display all the previous rentals of that customer, including the information of the books and number of borrowing days, the original cost, the total cost, and the discount. Invalid customer names will be handled, and the user will be given another chance until a valid customer name is entered. For example, if a customer named Emily made 3 previous rentals, one rental with the books Harry Potter 1 (7 days) and Gone Girl (12 days), one rental with the book A Brief History of Time (20 days), and another rental with the books The Republic (30 days) and The Hobbit (5 days), then the program will display the formatted message as follows.

```
This is the rental history of Emily.
            Books & Borrowing days                       Original Cost   Discount   Final Cost
Rental 1    Harry Potter 1: 7 days, Gone Girl: 12 days        8.30          0.83        7.47
Rental 2    A Brief History of Time: 20 days                  8.00          0.80        7.20
Rental 3    The Republic: 30 days, The Hobbit: 5 days        10.00          1.00        9.00
```

B - Code Requirements:

The program **must be entirely in one Python file named ProgFunA1_<Your Student ID>.py**. For example, if your student ID is s1234567, then the Python file must be named as ProgFunA1_s1234567.py. Other names will not be accepted.

Your code needs to be formatted consistently. You must not include any unused/irrelevant code (even inside the comments). What you submitted must be considered as the final product.

You should use appropriate data types and handle user inputs properly. You must not have any redundant parts in your code.

You must demonstrate your ability to program in Python by yourself, i.e., you should not attempt to use external special Python packages/libraries/classes that can do most of the coding for you. **The only Python library allowed in this assignment is the sys module.**

Note that in places where this specification may not tell you how exactly you should implement a certain feature, you need to use your judgment to choose and apply the most appropriate concepts from our course materials. You should follow answers given by your "client" (or "supervisor" or the teaching team) under Canvas/Discussions/Discussion on Assignment 1.

C - Documentation Requirements:

You are required to write comments (documentation) as a part of your code. Writing documentation is a good habit in professional programming. It is particularly useful if the documentation is next to the code segment that it refers to. Note that you don't need to write an essay, i.e., you should keep the documentation succinct.

Your comments (documentation) should be in the same Python file. Please DO NOT write a separate file for comments (documentation).

At the beginning of your Python file, your code must contain the following information:

1. **Your name and student ID.**
2. **The highest part you have attempted.**
3. **Any problems of your code and requirements that you have not met.** For example, scenarios that might cause the program to crash or behave abnormally, the requirements your program do not satisfy. Note, you do no need to handle errors that are not covered in the course.

Besides, the comments (documentation) in this assignment should serve the following purposes:

- Explain your code in a precise but succinct manner. It should include a brief analysis of your approaches instead of simply translating the Python code to English. For example, you can comment on why you introduce a particular function/method, why you choose to use a while loop instead of other loops, why you choose a particular data type to store the data information. These comments can be placed before the code blocks (e.g., functions/methods, loops, if) and important variable declarations that the comments refer to.
- Document some analysis/reflection as a part of your code. Here, you need to write some paragraphs (could be placed at the end or at the beginning of your code) to explain in detail your design process, e.g., how you came up with the design of the program, how you started writing the code after the design process, the challenges you met during the code development.
- Document the references, i.e., any sources of information (e.g., websites) you used other than the course contents directly under Canvas/Modules, **you must give acknowledgement of the sources, explaining in detail how you use the sources in this assignment**. More detailed information regarding the references can be found in Section 7.

D - Rubric:

Overall:

Part	**Points**
Part 1	6
Part 2	6
Part 3	5
Others (code quality, modularity, comments/reflection)	3

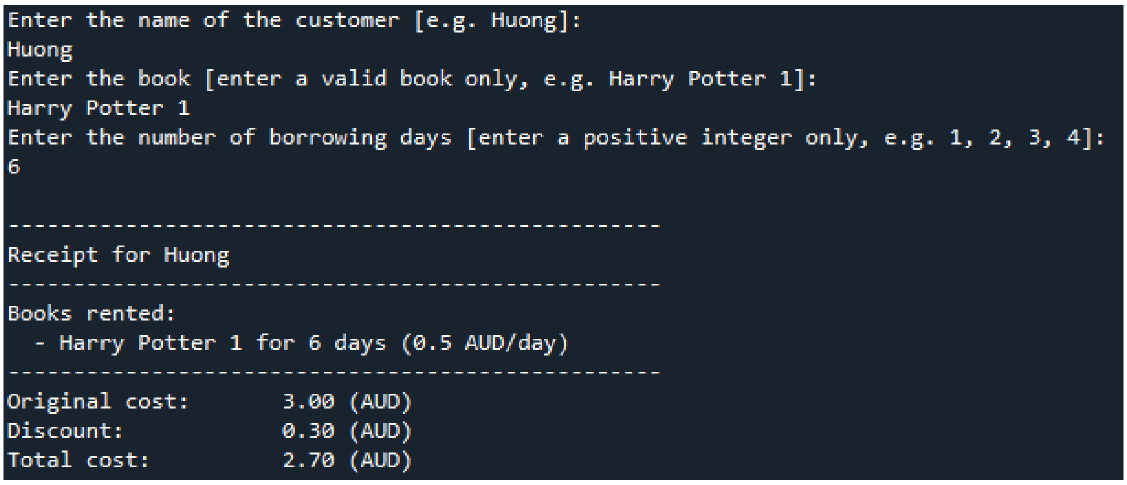
More details of the rubric of this assignment can be found on Canvas (here). Students are required to look at the rubric to understand how the assignment will be graded.

5. Example Program

We demonstrate a **sample program** that satisfies the requirements specified in Section 4. Note that this is just an example, so it is okay if your program looks slightly different, but you need to make sure that **your program satisfies the requirements listed in Section 4**.

5.1. PART 1

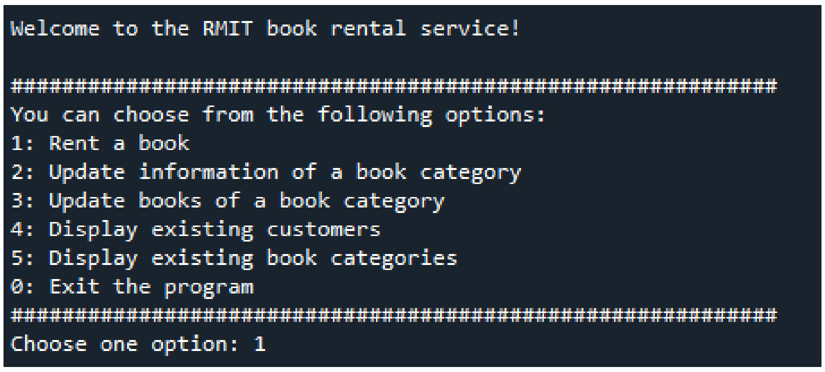
As an example, this is how the output screen of our sample program looks like for PART 1. In this example, the customer has the name *Huong* (who is also a member), rent the book Harry Potter 1 for 6 days. Note that in this test, we use the values *Huong, Harry Potter 1,* and *6* as examples only. You should test your program with different test cases, e.g., when a new customer (e.g., James) rents another book with another number of borrowing days, to make sure your program satisfies the requirements in Section 4.



```
Enter the name of the customer [e.g. Huong]:
Huong
Enter the book [enter a valid book only, e.g. Harry Potter 1]:
Harry Potter 1
Enter the number of borrowing days [enter a positive integer only, e.g. 1, 2, 3, 4]:
6

--------------------------------------------------
Receipt for Huong
--------------------------------------------------
Books rented:
  - Harry Potter 1 for 6 days (0.5 AUD/day)
--------------------------------------------------
Original cost:      3.00 (AUD)
Discount:           0.30 (AUD)
Total cost:         2.70 (AUD)
```

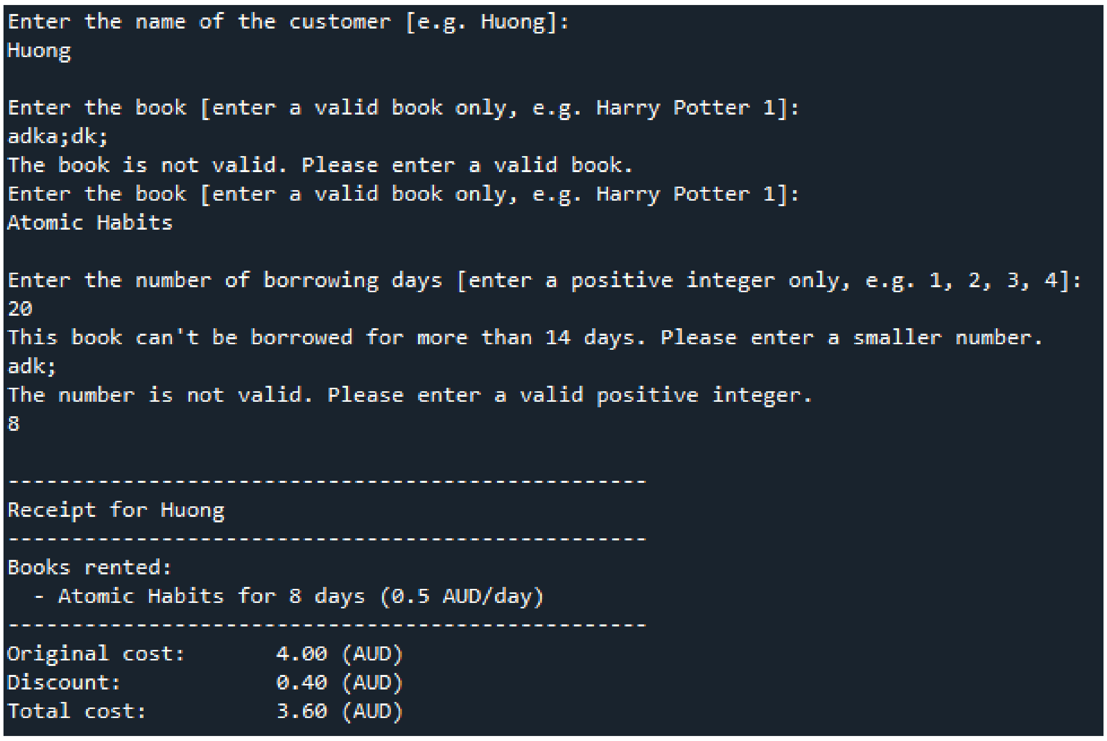
5.2. PART 2

As an example, this is how the output screen of our sample program looks like for a menu with all the options described in PART 2.



```
Welcome to the RMIT book rental service!

############################################################
You can choose from the following options:
1: Rent a book
2: Update information of a book category
3: Update books of a book category
4: Display existing customers
5: Display existing book categories
0: Exit the program
############################################################
Choose one option: 1
```

When the user (the receptionist) enters an option, the corresponding functionality will appear. For example, if the user chooses option 1, which is to rent a book, then the output screen of our sample program is as follows.

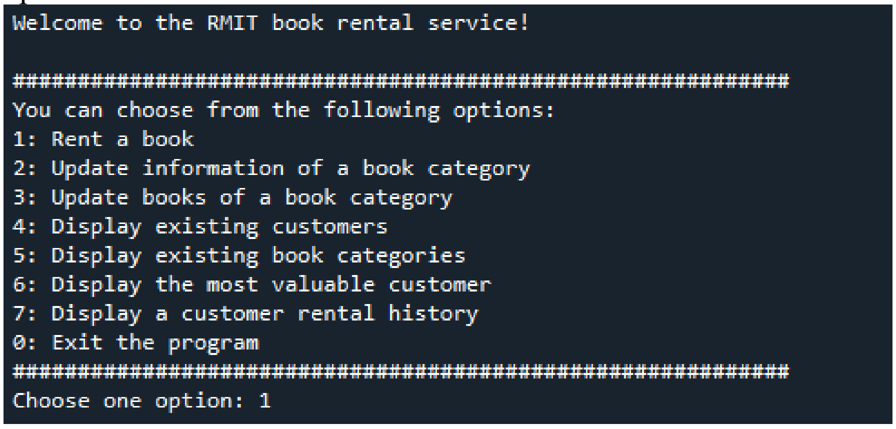


```
Enter the name of the customer [e.g. Huong]:
Huong

Enter the book [enter a valid book only, e.g. Harry Potter 1]:
adka;dk;
The book is not valid. Please enter a valid book.
Enter the book [enter a valid book only, e.g. Harry Potter 1]:
Atomic Habits

Enter the number of borrowing days [enter a positive integer only, e.g. 1, 2, 3, 4]:
20
This book can't be borrowed for more than 14 days. Please enter a smaller number.
adk;
The number is not valid. Please enter a valid positive integer.
8

--------------------------------------------------
Receipt for Huong
--------------------------------------------------
Books rented:
  - Atomic Habits for 8 days (0.5 AUD/day)
--------------------------------------------------
Original cost:       4.00 (AUD)
Discount:            0.40 (AUD)
Total cost:          3.60 (AUD)
```

Other requirements in PART 2 (update information of a book category, update books of a book category, display existing customers, display existing book categories) can also be displayed in a similar manner.

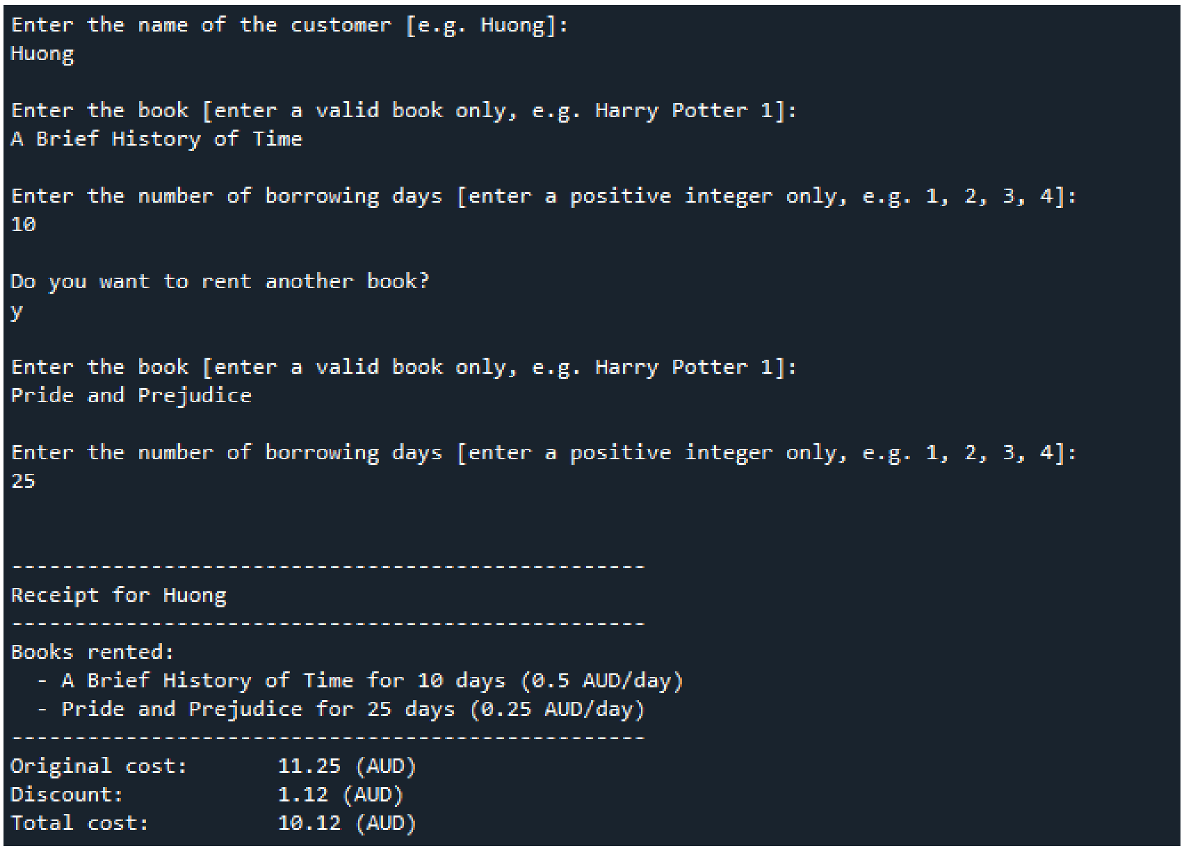
5.3. PART 3

As an example, this is how the output screen of our sample program looks like for a menu with all the options described in PART 3.



```
Welcome to the RMIT book rental service!

#############################################################
You can choose from the following options:
1: Rent a book
2: Update information of a book category
3: Update books of a book category
4: Display existing customers
5: Display existing book categories
6: Display the most valuable customer
7: Display a customer rental history
0: Exit the program
#############################################################
Choose one option: 1
```

This is an example showing the output screen of our sample program when we select option 1 and rent multiple books.



```
Enter the name of the customer [e.g. Huong]:
Huong

Enter the book [enter a valid book only, e.g. Harry Potter 1]:
A Brief History of Time

Enter the number of borrowing days [enter a positive integer only, e.g. 1, 2, 3, 4]:
10

Do you want to rent another book?
y

Enter the book [enter a valid book only, e.g. Harry Potter 1]:
Pride and Prejudice

Enter the number of borrowing days [enter a positive integer only, e.g. 1, 2, 3, 4]:
25


--------------------------------------------------
Receipt for Huong
--------------------------------------------------
Books rented:
  - A Brief History of Time for 10 days (0.5 AUD/day)
  - Pride and Prejudice for 25 days (0.25 AUD/day)
--------------------------------------------------
Original cost:       11.25 (AUD)
Discount:            1.12 (AUD)
Total cost:          10.12 (AUD)
```

6. Submission

As mentioned in the Code Requirements, **you must submit only one file named ProgFunA1_<Your Student ID>.py** via Canvas/Assignments/Assignment 1. It is your responsibility to correctly submit your file. Please verify that your submission is correctly submitted by downloading what you have submitted to see if the file includes the correct contents. The final .py file submitted is the one that will be marked.

Late Submission

All assignments will be marked as if submitted on time. Late submissions of assignments without special consideration or extension will be automatically penalised at a rate of 10% of the total marks available per day (or part of a day) late. For example, if an assignment is worth 20 marks and it is submitted 1 day late, a penalty of 10% or 2 marks will apply. This will be deducted from the assessed mark.

Special Consideration

If you are applying for extensions for your assessment within five working days after the original assessment date or due date has passed, or if you are seeking extension for more than seven days, you will have to apply for Special Consideration, unless there are special instructions on your Equitable Learning Plan.

In most cases you can apply for special consideration online here. For more information on special consideration, visit the university website on special consideration here.

7. Referencing Guidelines

What: This is an individual assignment, and all submitted contents must be your own. If you have used any sources of information (e.g., websites, tools) other than the course contents directly under Canvas/Modules, you must give acknowledgement of the sources, explaining in detail how you use the sources in this assignment, and give references using the IEEE referencing format.

Where: You can add a code comment near the work (e.g., code block) to be referenced and include the detailed reference in the IEEE style.

How: To generate a valid IEEE style reference, please use the citethisforme tool if you're unfamiliar with this style.

8. Academic Integrity and Plagiarism (Standard Warning)

Academic integrity is about honest presentation of your academic work. It means acknowledging the work of others whilst developing your own insights, knowledge, and ideas. You should take extreme care that you have:

- Acknowledged words, data, diagrams, models, frameworks and/or ideas of others you have quoted (i.e., directly copied), summarized, paraphrased, discussed, or mentioned in your assessment through the appropriate referencing methods.
- Provided a reference list of the publication details so your readers can locate the source if necessary. This includes material taken from the internet sites.

If you do not acknowledge the sources of your material, you may be accused of plagiarism because you have passed off the work and ideas of another person without appropriate referencing, as if they were your own.

RMIT University treats plagiarism as a very serious offence constituting misconduct. Plagiarism covers a variety of inappropriate behaviors, including:

- Failure to properly document a source
- Copyright material from the internet of databases
- Collusion between students

For further information on our policies and procedures, please refer to the University website (link).

9. Assessment Declaration:

When you submit work electronically, you agree to the assessment declaration: https://www.rmit.edu.au/students/student-essentials/assessment-and-results/how-to-submit-your-assessments