

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

This assignment's objective is to develop your programming and problem-solving skills step-by-step. 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:

- Follow coding, convention and behavioural requirements provided in this document and in the course lessons;
- Independently solve a problem by using programming concepts taught over the first several weeks of the course;
- 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 you have read Sections 1-3 of this document before going further.

Problem Overview: In this assignment, you are tasked to develop a point of sales (POS) system for a serviced apartment company branded 'Pythonia' that has three buildings under its management. The company's staff, i.e. the booking managers, use this POS system to take booking requests, process them and print out receipts on behalf of guests who would request to stay in an apartment unit for a short duration between 1-7 nights.

To simplify the case and for background information about the serviced apartment company, these few assumptions are to be made:

- The company manages 3 buildings namely Swan, Goose, and Duck, that contain multiple apartment units.
- The booking manager enters information on behalf of guests.
- Every apartment unit is given a unique ID and may have 1-3 bedrooms, however unit capacity is a more relevant attribute to keep track of. Unit capacity here is indicated by number of beds – to know how many people can fit inside the apartment unit at most.
- In addition to the nightly stay, the guests through the booking manager can also order supplementary items which are products or services that are available exclusively to apartment guests for additional costs. The supplementary items are optional and can only be ordered during the apartment booking and cannot be ordered without any apartment booking.
- It is not required to keep track of inventory or check for availability, as such:

- The apartment unit listed are always available to book.
- Supplementary items are always available to book and have sufficient stocks.
- The scope and limitation of this program is that it can only book 1 apartment unit per booking for maximum 7 nights, however it can order as many supplementary items as required.

You are required to implement the program following 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 booking managers) to book an apartment:

1. Display a message asking the user to enter the guest's name. In this part, you can assume the guest's name to be entered only consists of alphabet characters. If there will be more than 1 guest, only the main guest's will be required to be entered.
2. Display a message asking for number of guests that will stay in the apartment in numerical format.
3. Display a message asking the user to enter the apartment ID, which consists of the letter U, the unit number and apartment building name spelled as one word without any space. For example, if the chosen apartment is Unit 12 at Swan Building, it should be abbreviated and entered as U12swan).
4. Display a message asking for check-in date in Australian date format, that is d/m/yyyy for example 8/8/2024 for 8 August 2024 and 25/12/2024 for 12 December 2024.
5. Display a message asking for check-out date in Australian date format, that is d/m/yyyy for example 8/8/2024 for 8 August 2024 and 25/12/2024 for 12 December 2024.
6. Display a message asking for the length of stay. In this part, you can assume that the length of stay is always a positive integer, e.g., 1, 2, 3, and so on.
7. Display a message asking for the booking date (i.e. the current date in which the booking is being made) in Australian date format, that is d/m/yyyy for example 8/8/2024 for 8 August 2024 and 25/12/2024 for 12 December 2024.
8. Calculate the total cost for the guest, based on the apartment rate multiplied by length of stay. If the rate to stay in per night is \$200 and the length of stay is 3, then the total cost should be \$600.
9. Calculate the reward points earned by the purchase. For each dollar, there will be 1 reward point. The reward points will be rounded to the closest integer. For example, if the booking cost is \$599.50, the corresponding reward point is 600. If the booking cost is \$350.4, the corresponding reward point is 350.
10. All the booking information will be displayed as a formatted message to the user as follows. Note that the rate and total cost are all displayed with two decimal points.

=====

Pythonia Serviced Apartments - Booking Receipt

```

=====
Guest Name:      <guest_name>
Number of guests: <number_of_guests>
Apartment name:  <apartment_name>
Apartment rate:   $<apartment_rate_per_night> (AUD)
Check-in date:    <checkin_date>
Check-out date:   <checkout_date>
Length of stay:   <length_of_stay> (nights)
Booking date:     <booking_date>
-----
Total cost:       $<total_cost> (AUD)
Earned rewards:   <reward_points> (points)

Thank you for your booking! We hope you will have an enjoyable stay.
=====
  
```

11. In the program, you should have some lists (or dictionaries or other data types) to store the names of all guests, the accumulated reward points of the guests, the available apartments, the rate of each apartment. You can assume the guest names and the apartment names are all unique and case sensitive.
12. When a new guest finishes a booking, your program will automatically add the guest's name to the guest list and the earned reward points to the guest profile. When an existing guest finishes the booking, your program will add the earned reward points to the guest profile.
13. Your program needs to be initialized with the following existing guests: *Alyssa* and *Luigi*, with the reward points being 20 and 32, respectively. Your program will also be initialized with the following apartment IDs: *U12swan*, *U209duck*, *U49goose*, with the corresponding prices: 95.0, 106.7, 145.2.
14. Note: in the requirements No. 11, we use the term 'list' when describing the guest list, the item list, etc, but you can use other data types to store this information such as dictionaries and other data types.
15. In PART 1, your program can only take apartment booking and not supplementary items. Whilst user input validation is not required at this stage, make sure you think and analyse the requirements in detail so that you can choose the most appropriate/suitable data types.

----- **PART 2 (5 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 have the following features:

1. During the booking, a supplementary item such as an amenity or breakfast can be offered to apartment guests, however this requires the guest to firstly book to stay for at least 1 night.

Besides the apartment list, another list or dictionary should be initialized to include supplementary items with the following item IDs and prices. Description is for your note only and is not required to be included in the program.

Item ID	Price in AUD	Description
car_park	25 (per night)	Car park for 1 car.

breakfast	21 (per person)	Continental breakfast meal.
toothpaste	5 (per tube)	Toothpaste – generic brand.
extra_bed	50 (per item per night)	Removable extra bed that can fit up 2 people.

2. After the apartment unit is booked, prompt the user and ask if they want to order for a supplementary item such as “Do you want to order a supplementary item? (y/n)”, in which case the user could response accordingly with one character response. If they chose ‘y’ for yes, the program would prompt for additional fields to fill out:
 - Supplementary item ID.
 - Price.
 - Quantity.
3. After the above was prompted and entered, it will display the price and cost and prompt for confirmation. If they entered ‘y’, it will save the order, display the total cost so far, then prompt for another supplementary item such as “Do you want to order for another supplementary item? (y/n)”. If they entered ‘n’, it will not save the order, and display ‘Item cancelled’ and prompt to offer another supplementary item.
4. The program can take as many complimentary items as required and at the end of each item it prompting the user and ask if they want to order for another supplementary item until they chose ‘n’ for no. At the end of the booking, it will calculate the total cost, and display the receipt.
5. Handle invalid inputs from users:
 - a. Display an error message if the guest’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 apartment id entered by the user is not a valid apartment id – i.e. does not correspond to any of the ids in the apartment list. When this error occurs, the user will be given another chance, until a valid apartment id is entered.
 - c. Display an error message if the length of stay entered is 0, negative, or not an integer or greater than 7. When this error occurs, the user will be given another chance, until a valid length of stay is entered.
 - d. Display an error message if the supplementary item id entered by the user is not a valid item id – i.e. does not correspond to any of the ids in the supplementary item list. When this error occurs, the user will be given another chance, until a valid item id is entered.
 - e. Display an error message if the quantity 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.
 - f. Display an error message if the answer by the user is not y or n when prompting for additional booking or asking for confirmation. When this error occurs, the user will be given another chance, until a valid answer (i.e., y, n) is entered.
6. The following is the required format for Booking Receipt. If there is no supplementary item in the order, then the whole section about it will not be displayed.

```

=====
Pythonia Serviced Apartments - Booking Receipt
=====
Guest Name:      <guest_name>
Number of guests: <number_of_guests>
  
```

Apartment id: <apartment_name>
 Apartment rate: \$<apartment_rate_per_night> (AUD)
 Check-in date: <checkin_date>
 Check-out date: <checkout_date>
 Length of stay: <length_of_stay> (nights)
 Booking date: <booking_date>

Supplementary items

Item id: <item_id>
 Quantity: <item_quantity>
 Price: \$<item_price>
 Cost: \$<item_cost>

Item id: <item_id>
 Quantity: <item_quantity>
 Price: \$<item_price>
 Cost: \$<item_cost>

...

Sub-total: \$<supplementary_items_sub_total>

Total cost: \$<total_cost> (AUD)
 Earned rewards: <reward_points> (points)

Thank you for your booking! We hope you will have an enjoyable stay.

=====

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: make a booking, add/update information of a product, display existing guests, display existing products, 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. *Make a booking*: this option includes all the requirements from 1 to 10 in PART 1 and the entire first part of PART 2.
2. *Add/update information of an apartment unit*: this option displays a message asking the user for the information of an apartment unit (id, rate) to be added or updated. Besides the two attributes, the apartment unit now has an additional attribute called capacity (number of beds) as an integer. The id, rate, and capacity will be separated by white spaces and must be entered with the following format: *apartment_id rate capacity*. An apartment unit id has this special format, wherein it should always contain the leading capital U, followed by the unit number, and then the building name, for example *U12swan* refers to Unit 12 of Swan Building. This format must be validated. If it is incorrect, it will not save the change and go back to the main menu.

3. *Add/update information of a supplementary item*: this option displays a message asking the user for the information of a supplementary item (id, price) to be added or updated. The id and price will be separated by white spaces and must be entered with the following format: *item_id price*. For example, the user can enter *toothpaste 5.2* to add/update the product *toothpaste* with the price 5.2. If the product is an existing product, the newly entered price will replace the existing price. If the product is new, then it, its price will be added to the data collection of the program. You can assume users always enter the correct formats of the product and price, but note they can enter multiple white spaces. In this part, you can assume the *price* entered is always valid and is a positive number; you can also assume the user will only enter one product. If it is incorrect, it will not save the change and go back to the main menu.
4. *Display existing guests*: this option displays on screen all existing guests and their accumulated reward points. The messages and display format are flexible (your choice).
5. *Display existing apartment units*: this option displays on screen all existing apartments with their nightly rates. The messages and display format are flexible (your choice).
6. *Display existing supplementary items*: this option displays on screen all existing products with their prices. The messages and display format are flexible (your choice).
7. *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 (6 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 “*Make a booking*” option, apartment unit price (rate per night) is automatically displayed after the user entered the apartment ID, by referring to the initialized apartment list.
2. In the “*Make a booking*” option, during ordering of a supplementary item, its price is automatically displayed after the item ID is entered, by referring to the initialized supplementary item list.
3. In this part, in the “*Make a booking*” option, your program will validate the number of guests against the capacity (number of beds). If the number of guests exceeds the apartment unit capacity, then it will display a warning “please consider ordering an extra bed”. Each extra bed which is one of the default supplementary items would allow for two extra people to be accommodated. At the maximum two extra beds can be ordered to temporarily increase the apartment capacity by 4. If the number of guests still exceeds the capacity even after extra bed is ordered, display a message saying that booking cannot proceed and exit to main menu. If quantity entered is more than 2, display an error message accordingly. Have the option to confirm or cancel the order just like for any other supplementary item.
4. Furthermore, in this part, in the option “*Make a booking*”, your program will check the current reward points of a guest before finishing a booking, if they are larger than 100, then the program can deduct some amount of money (corresponding to the reward points) from the total cost and update the remaining reward points of the guests. Specifically, every 100 reward points can be converted to \$10 and be deducted from the order. Note the earned reward points from the order are still based on the total cost before the deduction. For example, if the total cost of the order is \$45, the guest currently has 120 reward points (before the booking), then the total cost will be \$35 (as the guest can convert 100 reward points to \$10), the earned reward points of this

purchase are still 45, and the guest will now have $20 + 45 = 65$ reward points after the purchase. The guest can choose to or not to spend (use) their reward points against the total order cost.

5. In this part, the option "Add/update information of a supplementary item" will become "Add/update information of supplementary items". This means, this option now can take a list of multiple items and prices separating by white spaces and commas. The format of the list is as following: *item_1 price_1, item_2 price_2, ...* For example, the user can enter *toothpaste 5.2, shampoo 8.2* to add/update the products *toothpaste* and *shampoo*. You can assume the users always enter the list in the right format, but there could be multiple spaces around the colon and commas. In this part, your program will check the prices entered in this option, if one of them is not a valid number, or negative number, or 0, the program will ask the user to enter the whole list again, until a valid list (contain all valid prices) is entered. This extra feature is not applicable for apartment unit.
6. The menu now has an option "Display a guest booking and order history". This option will display a message asking the user to enter the name of the guest, and the program will display all the previous orders of that guest, including the information of the products (apartment and supplementary) and quantities of their orders, the total cost, and the earned rewards. Invalid guest names will be handled, and the user will be given another chance until a valid guest name is entered. For example, if a guest named Alyssa made 3 previous purchases, one order with 1 *U12swan*, one order with 1 *U209duck* and 2 *breakfast*, and one order with 2 *U49goose* and 4 *breakfast*, and 2 *carpark* then the program will display the formatted message as follows.

This is the booking and order history for Alyssa.

	<i>List</i>	<i>Total Cost</i>	<i>Earned Rewards</i>
<i>Order 1</i>	<i>1 x U12swan</i>	<i>95.0</i>	<i>95</i>
<i>Order 2</i>	<i>1 x U209duck, 2 x breakfast</i>	<i>148.7</i>	<i>149</i>
<i>Order 3</i>	<i>2 x U49goose, 4 x breakfast, 1 carpark</i>	<i>424.4</i>	<i>424</i>

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. Dates should be simply be stored in string variables in Assignment 1. 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 not 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, tools) 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	5
Part 3	6
Others (code quality, modularity, comments/reflection)	3

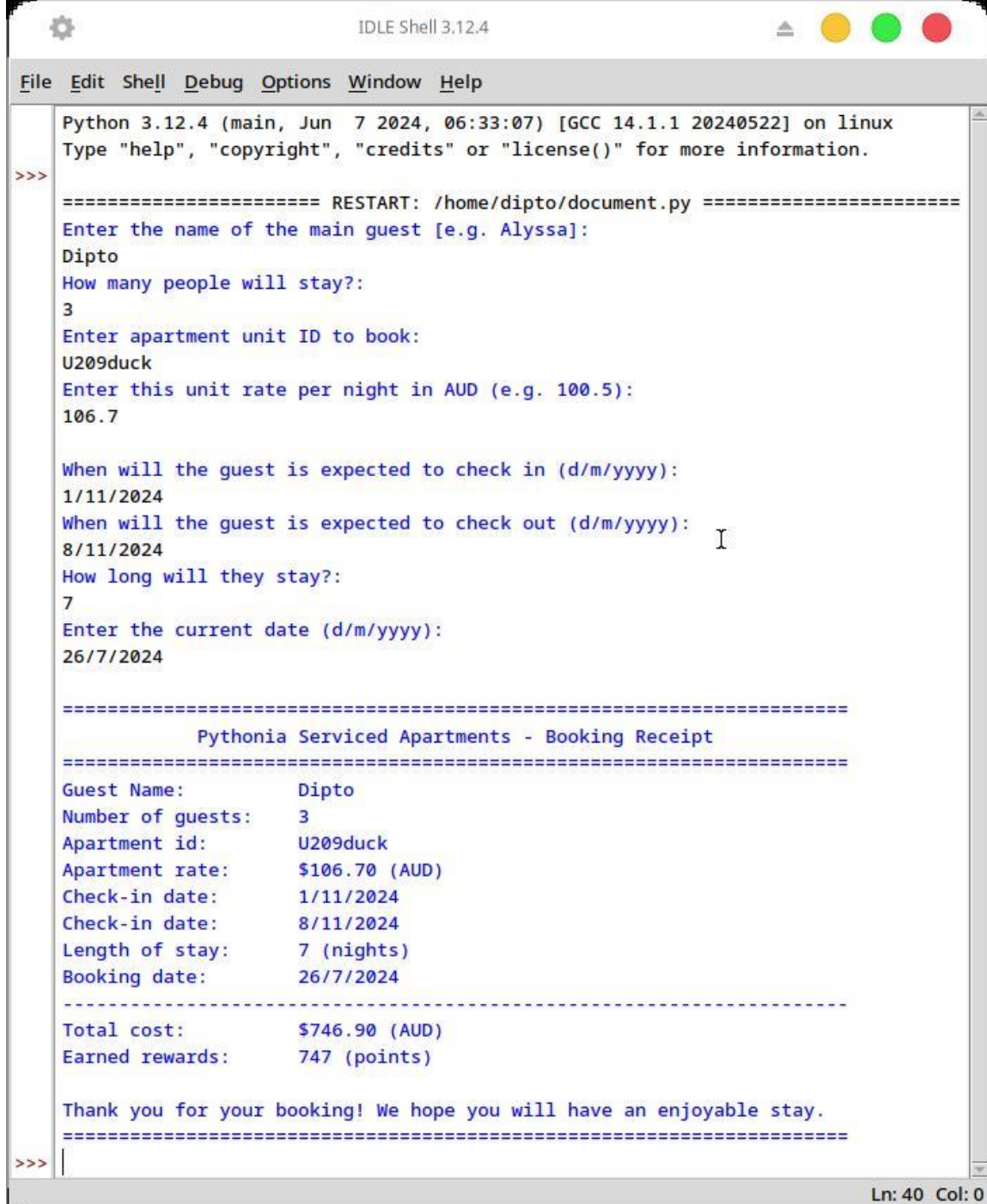
More details of the rubric of this assignment can be found on Canvas Assignment 1 page. Students are required to look at the rubric to understand how the assignment will be graded.

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.



```

Python 3.12.4 (main, Jun  7 2024, 06:33:07) [GCC 14.1.1 20240522] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /home/dipto/document.py =====
Enter the name of the main guest [e.g. Alyssa]:
Dipto
How many people will stay?:
3
Enter apartment unit ID to book:
U209duck
Enter this unit rate per night in AUD (e.g. 100.5):
106.7

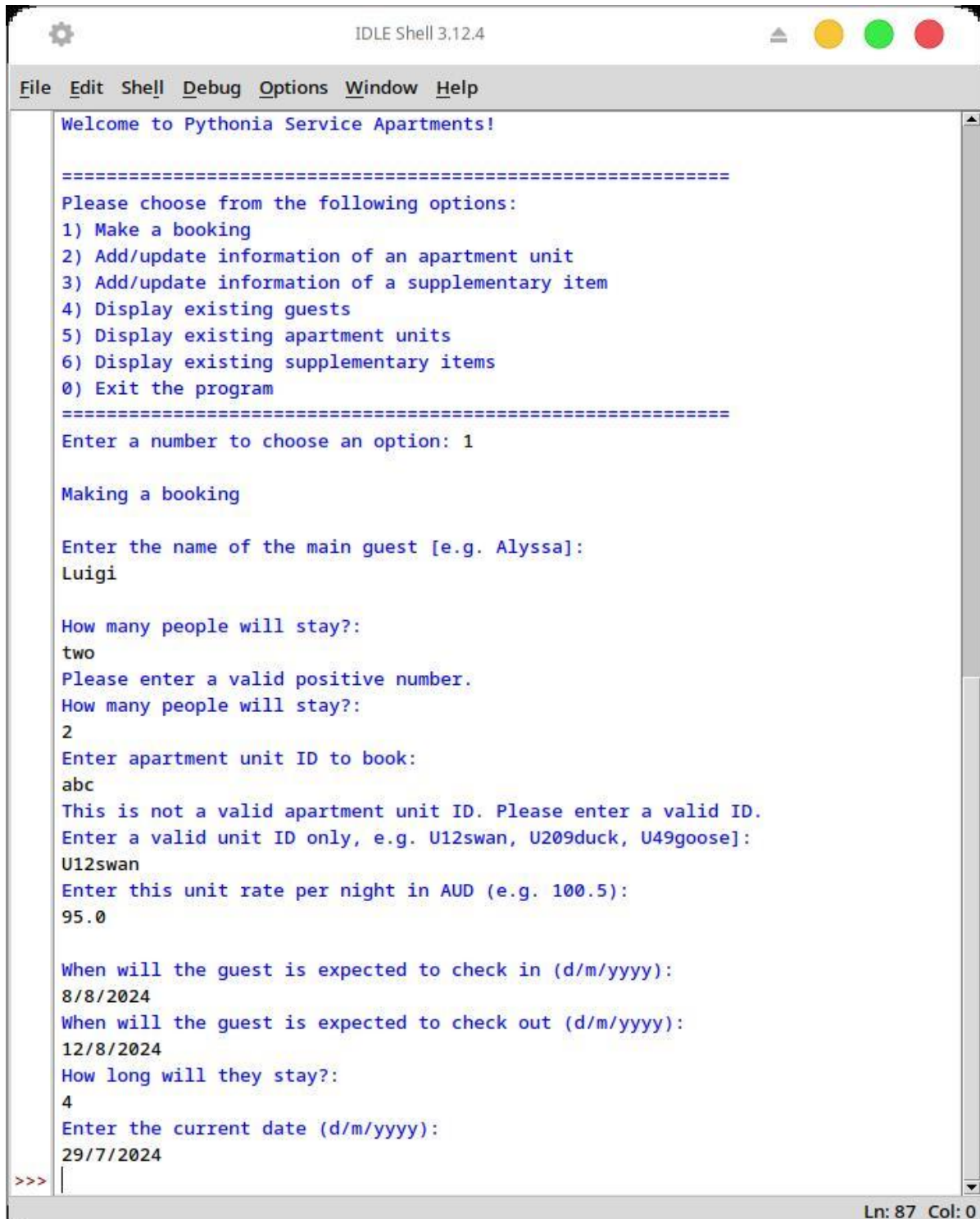
When will the guest is expected to check in (d/m/yyyy):
1/11/2024
When will the guest is expected to check out (d/m/yyyy):
8/11/2024
How long will they stay?:
7
Enter the current date (d/m/yyyy):
26/7/2024

=====
                Pythonia Serviced Apartments - Booking Receipt
=====
Guest Name:           Dipto
Number of guests:      3
Apartment id:          U209duck
Apartment rate:        $106.70 (AUD)
Check-in date:         1/11/2024
Check-in date:         8/11/2024
Length of stay:        7 (nights)
Booking date:          26/7/2024
-----
Total cost:            $746.90 (AUD)
Earned rewards:        747 (points)

Thank you for your booking! We hope you will have an enjoyable stay.
=====
>>>
  
```

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 Pythonia Service Apartments!

=====
Please choose from the following options:
1) Make a booking
2) Add/update information of an apartment unit
3) Add/update information of a supplementary item
4) Display existing guests
5) Display existing apartment units
6) Display existing supplementary items
0) Exit the program
=====
Enter a number to choose an option: 1

Making a booking

Enter the name of the main guest [e.g. Alyssa]:
Luigi

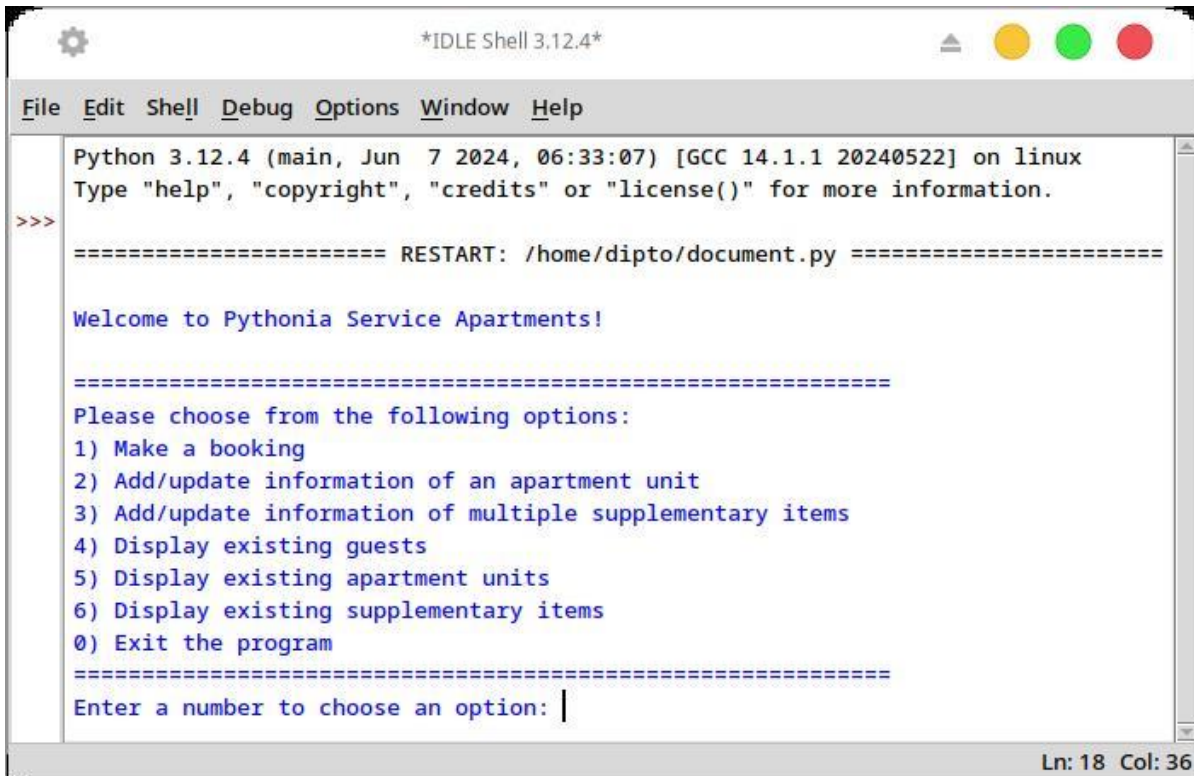
How many people will stay?:
two
Please enter a valid positive number.
How many people will stay?:
2
Enter apartment unit ID to book:
abc
This is not a valid apartment unit ID. Please enter a valid ID.
Enter a valid unit ID only, e.g. U12swan, U209duck, U49goose]:
U12swan
Enter this unit rate per night in AUD (e.g. 100.5):
95.0

When will the guest is expected to check in (d/m/yyyy):
8/8/2024
When will the guest is expected to check out (d/m/yyyy):
12/8/2024
How long will they stay?:
4
Enter the current date (d/m/yyyy):
29/7/2024
>>>
Ln: 87 Col: 0
```

Other requirements in PART 2 (add/update information of an apartment unit, display existing guests, display existing apartment units, etc) 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.



```
*IDLE Shell 3.12.4*

File Edit Shell Debug Options Window Help

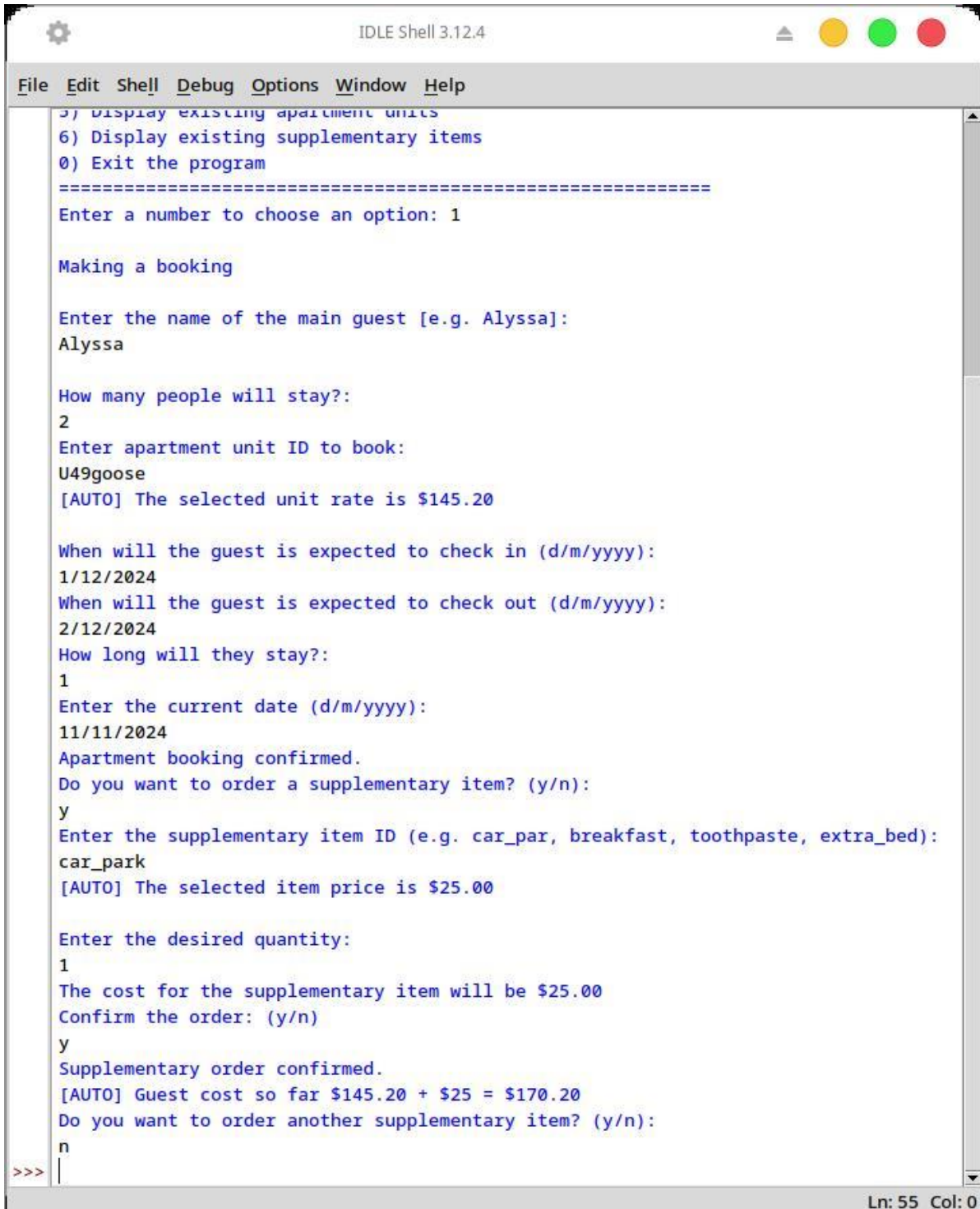
Python 3.12.4 (main, Jun 7 2024, 06:33:07) [GCC 14.1.1 20240522] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /home/dipto/document.py =====

Welcome to Pythonia Service Apartments!

=====
Please choose from the following options:
1) Make a booking
2) Add/update information of an apartment unit
3) Add/update information of multiple supplementary items
4) Display existing guests
5) Display existing apartment units
6) Display existing supplementary items
0) Exit the program
=====
Enter a number to choose an option: |

Ln: 18 Col: 36
```


This is an example showing the output screen of our sample program when we select option 1 and make a booking for apartment and then order for a supplementary item.



```
IDLE Shell 3.12.4
File Edit Shell Debug Options Window Help
5) Display existing apartment units
6) Display existing supplementary items
0) Exit the program
=====
Enter a number to choose an option: 1

Making a booking

Enter the name of the main guest [e.g. Alyssa]:
Alyssa

How many people will stay?:
2
Enter apartment unit ID to book:
U49goose
[AUTO] The selected unit rate is $145.20

When will the guest is expected to check in (d/m/yyyy):
1/12/2024
When will the guest is expected to check out (d/m/yyyy):
2/12/2024
How long will they stay?:
1
Enter the current date (d/m/yyyy):
11/11/2024
Apartment booking confirmed.
Do you want to order a supplementary item? (y/n):
y
Enter the supplementary item ID (e.g. car_par, breakfast, toothpaste, extra_bed):
car_park
[AUTO] The selected item price is $25.00

Enter the desired quantity:
1
The cost for the supplementary item will be $25.00
Confirm the order: (y/n)
y
Supplementary order confirmed.
[AUTO] Guest cost so far $145.20 + $25 = $170.20
Do you want to order another supplementary item? (y/n):
n
>>> |
```

Ln: 55 Col: 0

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 or 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>

Last update: 29-07-2024