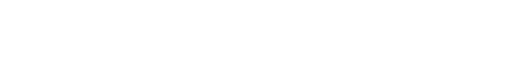
|  |  |
| --- | --- |
| **Programme** | HND in Computing |
| Student Name | Kalana Dissanayake |
| ANC ID | 1001262 |
| **Unit Number and Title** | Unit 19: Data Structures & Algorithms |
| Academic Year | 2023 – SU23 |
| Unit Tutor | Mahesh Wijenayake |
| **Assignment Title** | The Use and Application of Data Structures to an Organisation |
| Issue Date | 08-July-2023 |
| Submission Date | 23-Sep-2023 |
| Submitted on | 23-Sep-2023 |

**Unit 19: Data Structures & Algorithms**



Higher National Diploma in Computing

# Student Declaration

**Student declaration**

I certify that the assignment submission is entirely my own work and I fully understand the consequences of plagiarism. I understand that making a false declaration is a form of malpractice.

**Student signature**: Kalana **Date: 23/09/2023**

# Final Grade

|  |  |  |
| --- | --- | --- |
| **Grade:** | **Assessor Signature:** | **Date:** |
| **Assessor Feedback:** | |  |
| **Grade:** | **Assessor Signature:** | **Date:** |
| **Internal Verifier’s Comments:** | |  |

**Table of Content**

[01. Description of the Scenario and the Designed Algorithm 3](#_Toc146268681)

[1.1 Overview of the Scenario 3](#_Toc146268682)

[1.2 Designed Algorithm (Pseudo Code) 4](#_Toc146268683)

[02. Implementation of the Program 6](#_Toc146268684)

[03. Error Handling and the Test Results 10](#_Toc146268685)

[3.1 Error Handling 10](#_Toc146268686)

[3.2 Test Cases and Result 11](#_Toc146268687)

[4. How the ADT/algorithm implementation resolves the issue 16](#_Toc146268688)

[5. The degree of complexity of the methods used 17](#_Toc146268689)

[5.1 Algorithms 17](#_Toc146268690)

[5.2 Data Structures 18](#_Toc146268691)

**Table of Figures**

[Figure 1: Pseudo Code - Part 01 5](#_Toc146269002)

[Figure 2:Pseudo Code - Part 02 6](#_Toc146269003)

[Figure 3:Pseudo Code - Part 03 6](#_Toc146269004)

[Figure 4: Pseudo Code - Part 04 7](#_Toc146269005)

[Figure 5: Pseudo Code - Part 05 7](#_Toc146269006)

[Figure 6: Implementation of the Class Menu 8](#_Toc146269007)

[Figure 7: Implementation of the Classes Table and Program 9](#_Toc146269008)

[Figure 8: Implementation of the Place\_order Method 9](#_Toc146269009)

[Figure 9: Methods of Income, Table status, Popularity, Beverages 10](#_Toc146269010)

[Figure 10: Implementation of the Run Method 11](#_Toc146269011)

[Figure 11: Try-Except code block 12](#_Toc146269012)

[Figure 12: Output of Case 01 13](#_Toc146269013)

[Figure 13: Output of Case 03 14](#_Toc146269014)

[Figure 14: Output of Case 03 14](#_Toc146269015)

[Figure 15: Output of Case 04 15](#_Toc146269016)

[Figure 16: Output of Case 05 16](#_Toc146269017)

[Figure 17: Output of Case 06 16](#_Toc146269018)

[Figure 18: Output of Case 07 17](#_Toc146269019)

**Programming Report**

# 01. Description of the Scenario and the Designed Algorithm

## 1.1 Overview of the Scenario

Luccio Carlo's, a tiny, family-run Italian eatery, has acquired its very first Michelin star, a rare distinction. This notoriety has substantially improved the restaurant's image and customer demand, necessitating an upgrade from its outdated manual system for ordering to a computerized solution for managing many aspects of its operations. Luccio Carlo's, a relatively small business, ran all aspects of business using a straightforward paper-based method. Due to the dining establishment's recent success and popularity, there is now a demand for a more modern and efficient way to manage ordering, billing and submitting reports for each evening's service.

* The restaurant has undergone a significant shift that has expanded complexity and business since receiving the Michelin Star. The manual system could no longer fulfill its criteria as a result.
* Luccio Carlo's has looked into a variety of choices with software companies, but the offered answers were deemed to be too pricey and complex for the restaurant's present stage of growth.
* Currently, the restaurant only has one waitstaff member who is in responsible of serving each table. Only one device is used for handling orders and client interactions.
* On the smartphone or tablet that the server utilizes, the client anticipates that the program will be compact and run in memory, ensuring that it won't consume an excessive amount of CPU power.

**Business Requirements:**

* Each evening, there are five set tables available.
* There are two meal times: early (6–8 pm) and late (8–10 pm).
* Management of table reservations to prevent duplicate bookings.
* Monitoring the availability and occupancy of tables.
* Flexible dining choices with up to eight people per table.
* A wide selection of appetizers, entrées, desserts, and drinks.
* Calculating and showing the price per diner and the total price per table.
* Managing payment information, which includes a 10% fee for credit card transactions.
* The choice to tip at the discretion of the diners.
* Diane, give processing orders involving lobster or steak special attention.
* End-of-evening reports on revenue, the table with the greatest expenditure, total tips, and the most popular menu items.

## 1.2 Designed Algorithm (Pseudo Code)

For the organization Luccio Carlo's, we created the algorithm below with the intention of building up software that replicates the functioning of a management system for restaurants. Additionally, we developed the following Pseudo code into a full program using the Python programming language.

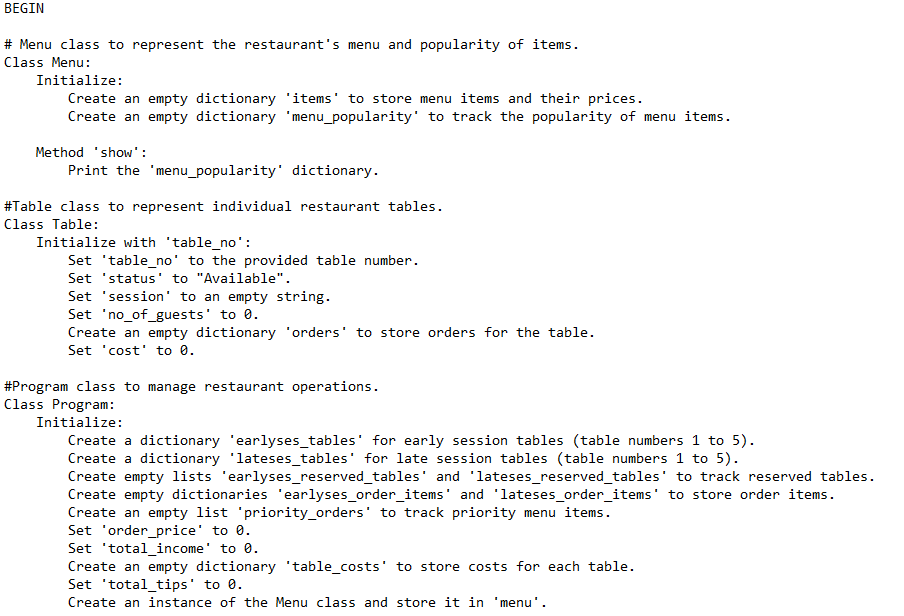


Figure : Pseudo Code - Part 01

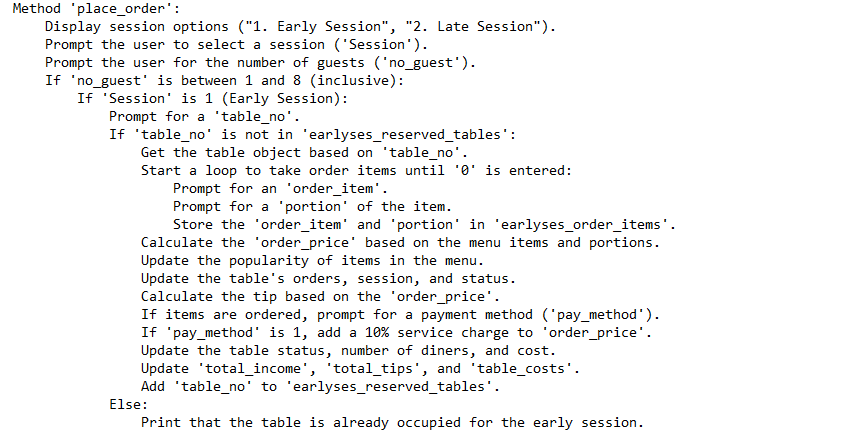


Figure :Pseudo Code - Part 02

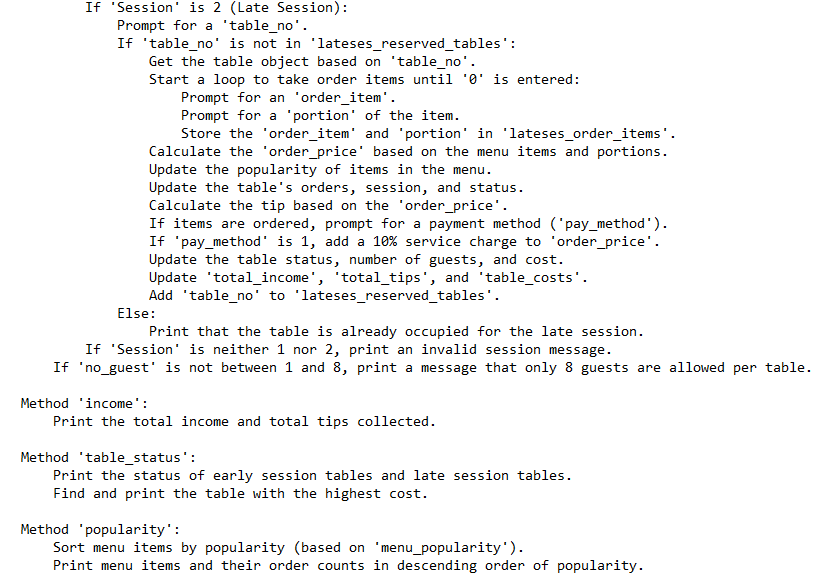


Figure 3:Pseudo Code - Part 03

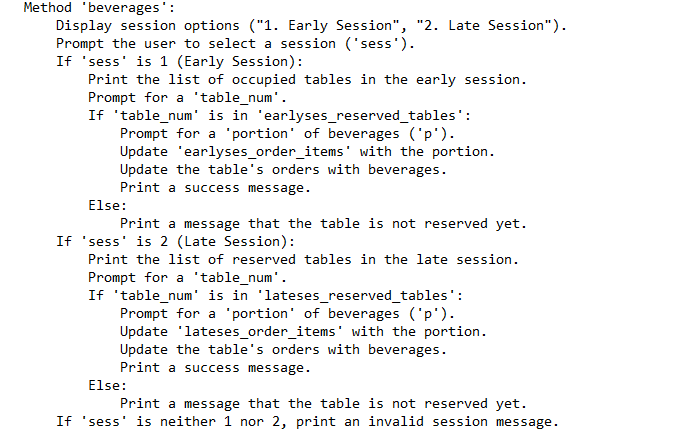


Figure 4: Pseudo Code - Part 04

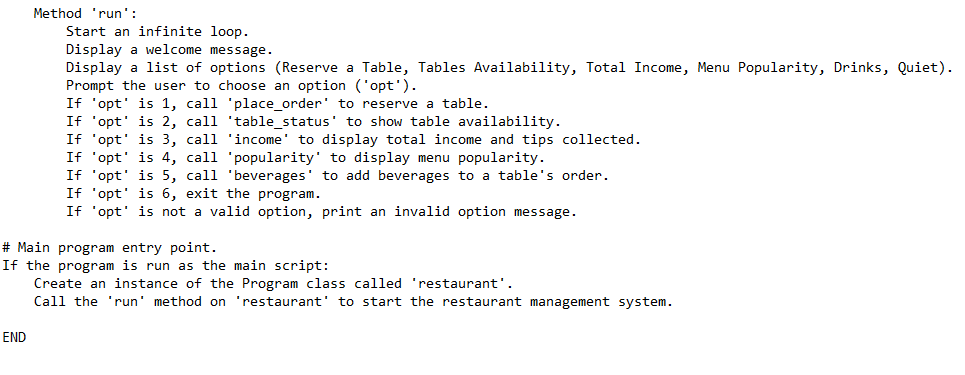
****

Figure 5: Pseudo Code - Part 05

# 02. Implementation of the Program

**Class Menu** - The Menu class specifies the menu elements and their costs for the restaurant. to start a dictionary called self. items, the menu items are utilized as keys, and the prices that go with them as values.

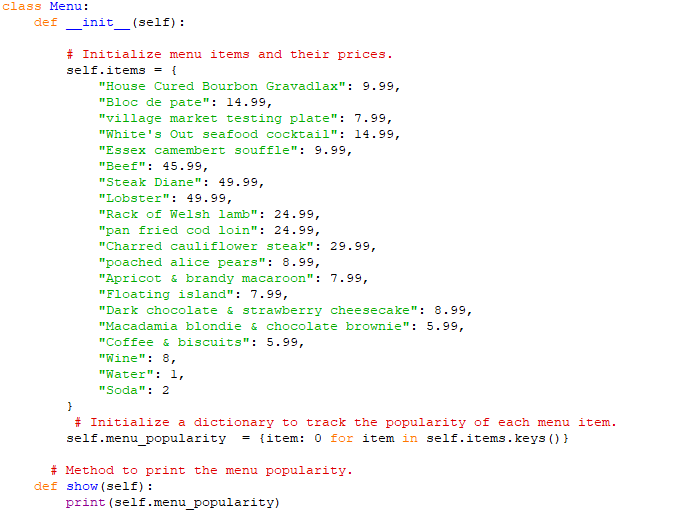
****

Figure 6: Implementation of the Class Menu

**Class Table** – Table class shows the tables in the restaurant.

**Class Program** - The Program class stands in for the core components of the restaurant management system. It initializes several data structures for handling tables, orders, revenue, and other data. Each table contains characteristics such as status, session, visitors, orders, and price. As dictionaries, the tables are shown. Reserved tables are kept in two lists, earlyses\_reserved\_tables and lateses\_reserved tables. The priority\_orders list keeps track of prioritized orders, such Lobster or Steak Diane. Additionally monitored are total revenue, gratuity, and table expenses. A self.menu instance of the Menu class is constructed to manage the restaurant's menu.

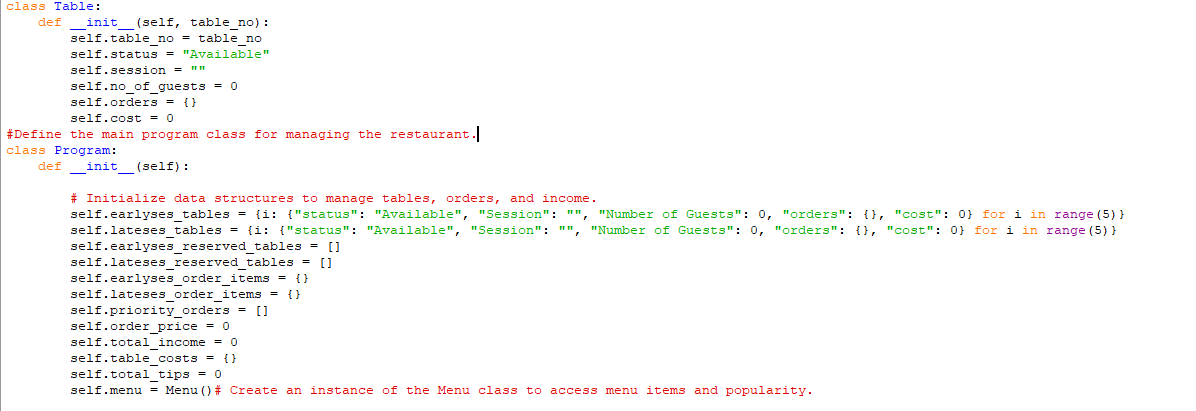


Figure 7: Implementation of the Classes Table and Program

**Place order method** - Customers can use this method to order meals. Inputs include the time period, client name, number of guests, table number, food choices, and payment type. In addition to updating different data structures and gathering advice, the order cost is determined.

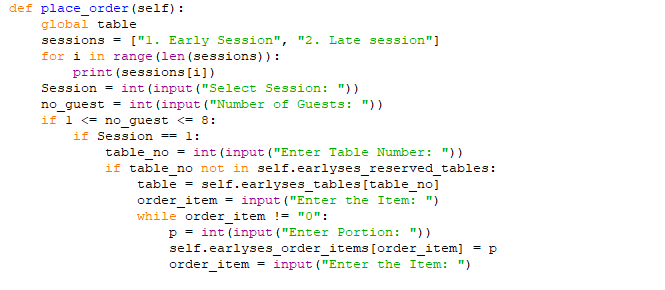


Figure 8: Implementation of the Place\_order Method

**Income Method** - You might view your daily revenue and tip totals by employing this technique.

**Table status Method** - The present condition of the tables during the early and late sessions, as well as their costs, are displayed for the table with the greatest cost.

**Popularity Method** - Based on the quantity of orders, this technique demonstrates how popular particular menu items are.

**Beverages Method** - This approach enables the addition of drinks to orders.



Figure 9: Methods of Income, Table status, Popularity, Beverages

**Run Method** - The main control loop of the software displays a list of options for the customer to choose from. It constantly requests input and carries out the selected action until the user decides to quit.

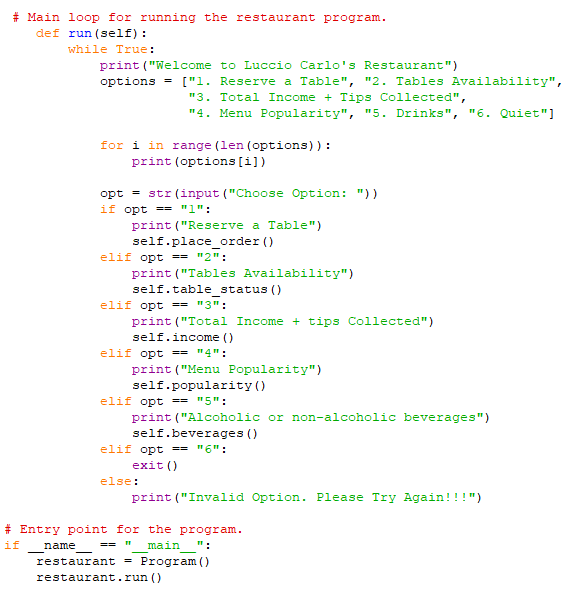


Figure 10: Implementation of the Run Method

# 03. Error Handling and the Test Results

## 3.1 Error Handling

Error management is a crucial component of the creation of software, and programmers should use effective error handling strategies to make sure their programs are dependable and resilient.

Here is how mistakes are handled by Luccie Carlo's restaurant management system. Try-except blocks were used throughout programming to account for probable problems.

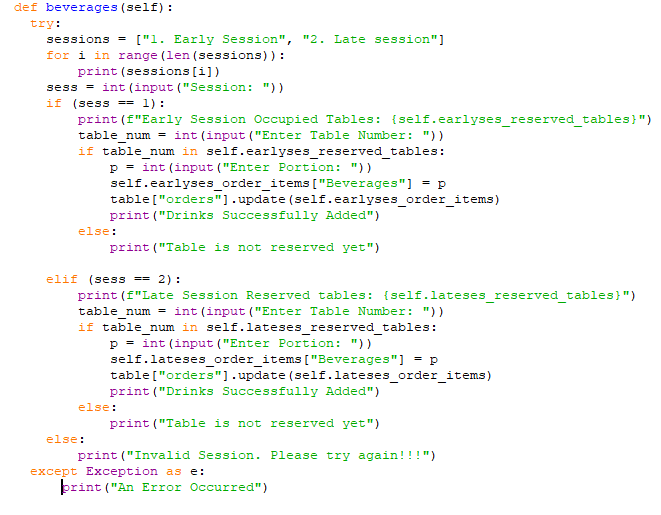


Figure 11: Try-Except code block

## 3.2 Test Cases and Result

**1. Case 01: Reserving a Table**

Table 1: Case 01 Input and Output

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Option** | **Session** | **No. of Guests** | **Table number** | **Item** | **Portion** | **Payment Method** |
| **Input** | **1** | **1** | **5** | **4** | **Beef** | **4** | **1** |
| **Output** | **Table Reserved Successfully** | | | | | | |



Figure 12: Output of Case 01

Result: The application bookings the table charges a 10% service charge when using a debit or credit card, and then displays a confirmation message. Nothing goes wrong.

**2. Case 02: Checking the tables Availability**

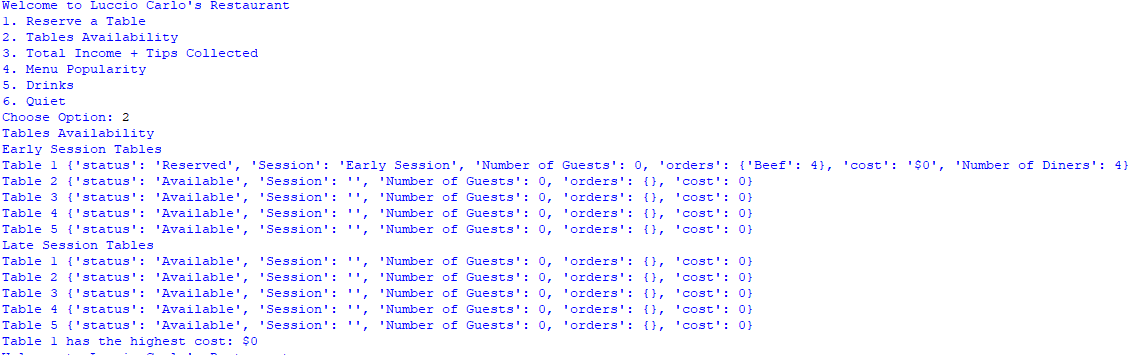


Figure 13: Output of Case 03

Result: For all early and late sessions, the application displays the most expensive table as well as each table's condition. Nothing goes wrong.

**3. Case 03: Checking the total income and tips collected**

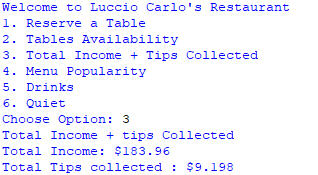
****

Figure 14: Output of Case 03

Result: By selecting the option as 3 It displays the total income and the amount of tips collected. So, it implements without making an error.

**4. Case 04: Checking Menu Popularity**

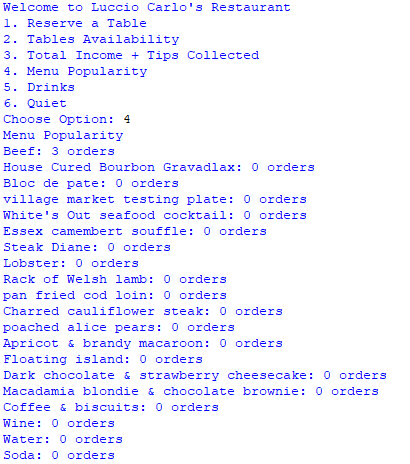
****

Figure 15: Output of Case 04

Result: According to the number of orders, the menu items are listed in descending sequence by the software. No errors generated.

**5. Case 05: Adding Beverages**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Option** | **Session** | **Table number** | **Portion** |
| **Input** | **5** | **1** | **4** | **5** |
| **Output** | **Drinks Successfully Added** | | | |

Table 2: Case 05 Input and Output

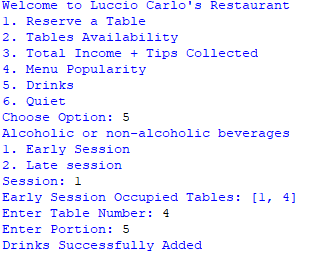


Figure 16: Output of Case 05

Result: By adding options as above It will add drinks to a selected table successfully.

**6. Case 06: Quit the Application**

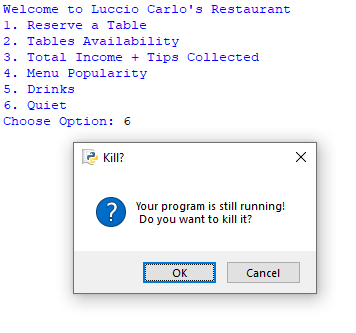
****

Figure 17: Output of Case 06

Result: The application ends successfully. No errors generated.

**7. Case 07: Checking an Invalid Input**

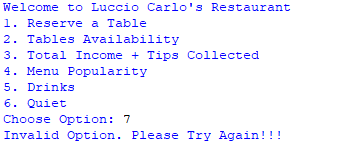
****

Figure 18: Output of Case 07

Result: By entering an invalid input to the program. It displays an error message as “Invalid Option. Please try Again”. So, the program is handling the invalid inputs without errors.

# 4. How the ADT/algorithm implementation resolves the issue

Python programming is used to set up an ordering and reservation process for Luccio Carlo's restaurant. The requirements of the scenario are accomplished by using an Object-Oriented technique.

* To manage tables, orders, money, tips, and menu items, the software begins by initializing a number of data structures and variables. Additionally, it generates a new instance of the Menu class, which has data about the menu items and their popularity in the restaurant.
* The customer is able to reserve a table, take orders, and set the pricing using the Placing order method.
* The restaurant's overall revenue and tips are shown using the income method.
* The state of every table is shown using the table\_status method for both the early and the late sessions. Additionally, it determines the most expensive table.
* Based on the number of orders made for each item, the popularity method shows how popular each menu item is.
* Customers can add drinks to their orders using this approach. The session and table numbers are requested, and a reservation status check is performed. If so, the consumer may include beverage products in their purchase.
* The user is given a menu of alternatives as the application loops continuously. Users have the option to make a reservation, verify the availability of a table, examine earnings and tips, look up menu popularity, add beverages, or quit the application.

This application efficiently handles the management of restaurant operations by monitoring table bookings, orders, revenue, and the popularity of menu items. Customers and restaurant workers may engage with the system through a user-friendly interface and make choices based on the information at hand.

# 5. The degree of complexity of the methods used

## 5.1 Algorithms

Placing order method:

* User input is required for the session, table number, menu item selection, and payment method in this manner. Since user input mostly determines the operation's difficulty, it is challenging to evaluate the operation's time complexity. The majority of individual actions, however, have much complexity, including updating dictionaries and figuring expenses.

Income Method:

* Simple mathematical calculations and dictionary searches are required to compute and display total income and tips.

Table status Method:

* This technique outputs the table data after iterating over the dictionaries that contain it.

Popularity Method

* Comparing each menu item's popularity score is necessary for arranging the menu popularity dictionary. The amount of menu options determines the complexity.

Beverage Method:

* User input is required for the session, table number, and portions in this technique. User input determines how sophisticated it is.

## 5.2 Data Structures

Menu Class:

* The menu items including their cost are kept in a dictionary by the Menu class. To keep tabs on the popularity of every menu item, it also employs another dictionary. Both of these data structures are effective because they support constant-time lookup, inserting, and deletion.

Table Class:

* Restaurant tables are represented using the Table class. Table-specific data, including table number, status, session, number of guests, orders, and cost, is stored using instance variables. Because there are no complicated data structures in this class, operations on the attributes are often simple and effective.

Program Class:

* The restaurant's management employs a number of dictionaries to keep data on tables, orders, revenue, tips, and the most popular menu items. Dictionary entries for ordered objects, session-specific tables, and table status are all included in these data structures. Specific menu items are prioritized using the priority\_orders list. These data structures are adequate for the application's requirements in general, although the complexity will vary depending on the precise actions carried out.

The ADT (Abstract Data Type) and methods developed in this application are typically simple and effective for running a small restaurant. However, some actions, such as sorting menu popularity, could cause a performance bottleneck as the program's scope and data volume expand. In such circumstances, scale and optimization considerations would be required.