Group Assignment

CT010-3-1-FSD

Fundamentals of Software Development

UC1F1809 SE/IT/CS/CS(DA)/CE/IS/MMT/IT(IOT)/CGD

Hand Out Date: 08 October 2018

Hand In Date: 31 December 2018

Weight: 50%

Instructions To Candidates:

1. Submit your assignment at the administrative counter.
2. Students are advised to underpin their answers with the use of references (cited using the Harvard Name System of Referencing).
3. Late submission will be awarded zero (0) unless Extenuating Circumstances (EC) are upheld.
4. Cases of plagiarism will be penalized.
5. The assignment should be bound in an appropriate style (comb bound or stapled).
6. Where the assignment should be submitted in both hard copy and soft copy, the soft copy of the written assignment and source code (where appropriate) should be on a CD in an envelope / CD cover and attached to the hard copy.
7. You must obtain 50% overall to pass this module.

Version 1.2 UCTI AssCv 2007-10-23

Marking Grid

Group Component (50%)			
Fully	Partially	Not	Performance Criteria
			Pass (20 - 29 marks)
			Program execution
			Module integration
			Between 50% - 65% of the system's modules are coded
			Basic programming structures (control structure) implemented
			Design of the solution in pseudo-code and flowchart covers 50% - 60% of the basic requirements of the system
			Some errors / omissions in design - pseudo-code / flowchart
			Less than 50% of documentation complete
			Credit (30 - 39 marks)
			All pass criteria fully met
			Good system design & module integration
			Between 65% - 75% of the system's modules are coded
			Basic programming structures (control structures) implemented
			Intermediate programming structures (lists, functions and file I/O) implemented
			Design of the solution in pseudo-code and flowchart covers 65% - 75% of the basic requirements of the system
			Minor errors / omissions in design - pseudo-code / flowchart
			Between 60% - 80% of the documentation complete
			Distinction (40 - 50 marks)
			All credit criteria fully met
			Excellent system design & module integration
			More than 80% of the system's modules coded
			Intermediate programming structures (lists, functions and file I/O) implemented
			Detailed design of the solution in pseudo-code and flowchart in terms of style and unique logics
			Hardly any errors / omissions in design - pseudo-code / flowchart
			Above 80% of the documentation complete

Individual Component (50%)			
	Grace Ong TP053002	Lau Dian Heng TP053385	Toh Hon Jun TP053047
Grade/Marks			
Understanding of algorithm (10%)	Fully Partially Not	Fully Partially Not	Fully Partially Not
Contribution toward group work (10%)	Fully Partially Not	Fully Partially Not	Fully Partially Not
Understanding of programming structures (15%)	Fully Partially Not	Fully Partially Not	Fully Partially Not
Presentation and QnA (15%)	Fully Partially Not	Fully Partially Not	Fully Partially Not

Additional Comments

FSD Group Assignment

```
    dMMMMMP dMMMMMP dMMMMb  dMMMMb  dMP dMP
   dMP      dMP      dMP.dMP dMP.dMP dMP.dMP
  dMMMP    dMMMP    dMMMMK" dMMMMK"  VMMMMP
 dMP      dMP      dMP"AMF dMP"AMF dA .dMP
dMP      dMMMMMP dMP dMP dMP dMP  VMMMP"

 dMMMMMMP dMP .aMMMb  dMP dMP dMMMMMP dMMMMMMP dMP dMMMMb  .aMMMMP
   dMP   amr dMP"VMP dMP.dM" dMP         dMP   amr dMP dMP dMP"
  dMP    dMP dMP     dMMMK"  dMMMP       dMP   dMP dMP dMP dMP MMP"
 dMP    dMP dMP.aMP dMP"AM" dMP         dMP   dMP dMP dMP dMP.dMP
dMP    dMP  VMMMP" dMP dMP dMMMMMP     dMP   dMP dMP dMP  VMMMP"

   .dMMMb  dMP dMP .dMMMb dMMMMMMP dMMMMMP dMMMMMMMMb
  dMP" VP dMP.dMP dMP" VP   dMP    dMP      dMP"dMP"dMP
  VMMMb   VMMMMP  VMMMb    dMP    dMMMP    dMP dMP dMP
dP .dMP dA .dMP dP .dMP   dMP    dMP      dMP dMP dMP
VMMMP"  VMMMP"  VMMMP"   dMP    dMMMMMP dMP dMP dMP
```

Members

1. Grace Ong - TP053002
2. Lau Dian Heng - TP053385
3. Toh Hon Jun - TP053047

Table of Contents

Workload Matrix

Student	Workload (%)	Signature
Grace Ong TP053002	33	
Lau Dian Heng TP053385	33	
Toh Hon Jun TP053047	33	

Basic Requirements

Build a ferry ticketing system for a ferry company.

The company has 8 ferries. Each ferry has an unique ID:

- 001
- 002
- 003
- 004
- 005
- 006
- 007
- 008

Each ferry has 50 seats. 10 seats for business class, 40 seats for economy class.

There are two routes:

- Penang to Langkawi
- Langkawi to Penang

Ferries depart at following 8 times:

- 10am
- 11am
- 12pm
- 1pm
- 2pm
- 3pm
- 4pm
- 5pm

This system can sell ferry ticket to customer.

This system can print ticket for customer.

This system can show seating arrangement of a ferry.

Assumptions

This section is about those requirements and info which are not given in the instruction but are needed to implement this system.

The Ferry Ticketing System is designed for the ferry ticket counter staffs. There are few features added to the system for the staffs to cope with any situation.

1. The staffs can select the seats either manually or automatically depending on the customers' demands.
2. The staffs can select seats in bundle depending on the seat availability.
3. The staffs can reprint the customers' tickets using the search engine provided.
4. The staffs view customers' purchase details in the seating arrangement section.
5. The staffs can quickly check the seat availability in the purchase ticket section.

There are only 8 ferries, each with unique ferry ID in the system. At 10am, 4 ferries will depart Penang to Langkawi and another 4 ferries will depart from Langkawi to Penang. These ferries will travel back and forth at every one-hour intervals from 10am to 5pm.

Times	**Ferry 001 to 004**	**Ferry 005 to 008**
10am	Penang to Langkawi	Langkawi to Penang
11am	Langkawi to Penang	Penang to Langkawi
12pm	Penang to Langkawi	Langkawi to Penang
1pm	Langkawi to Penang	Penang to Langkawi
2pm	Penang to Langkawi	Langkawi to Penang
3pm	Langkawi to Penang	Penang to Langkawi
4pm	Penang to Langkawi	Langkawi to Penang
5pm	Langkawi to Penang	Penang to Langkawi

Besides that, the system only allows the staffs to sell ferry tickets that have date within 7 days from today. The system will automatically record the purchase data of each tickets and store them in a file according to the date. For example, the data of the ferry tickets of 2018-12-31 will be stored in a file named `data-2018-12-31.json`.

System Features

This section is about features added to the system that are not specified in the instruction.

Customer can choose seats manually, or can let computer auto-assign seats. When customers select their seats manually, the seating chart will be shown and passengers are able to choose ferry and seats (able to purchase multiple tickets); if passengers select auto-assign, then customers need to choose number of tickets to buy. After selecting tickets, they need to enter the name of passenger for every ticket.

This system can sell tickets for one week time.

This system can view passenger info, given seat number or customer name. Bought ticket can be reprinted.

Data will be stored, that means data will be retained after system restart.

There is a limit of trial for invalid input. It will return to main menu if that limit is reached.

User is able to go back to previous step, and able to return to main menu, at any step.

List of steps from main menu step to current step will be shown at each page.

Data Design

This section is about the format of data. This is the format of `data` variable in the code, and the format of content saved in the file.

Each file stores data for a day. E.g. `data_2018-12-01.json`.

Each day has 8 times, with the following indexes:

- 0: 10am
- 1: 11am
- 2: 12pm
- 3: 1pm
- 4: 2pm
- 5: 3pm
- 6: 4pm
- 7: 5pm

Each time has 8 ferries, with the following indexes:

Index	**Ferry ID**		**Route**
	If time index is even	**If time index is odd**	
0	001	005	Penang to Langkawi
1	002	006	
2	003	007	
3	004	008	
4	005	001	Langkawi to Penang
5	006	002	
6	007	003	
7	008	004	

Each ferry has 50 seats, indexed from 0 to 49.

Each seat can either be null or object containing:

- Name (string)
- Purchased date and time (timestamp, integer)

Flow Design

This section is about flow of using the system. A simplified format of flowchart is used here, because the original flowchart format is too detailed and too long. The flowchart and pseudocode should be used to show an overall view of the system, or a rough idea about certain part of the system. There is no point for translating the entire python source code into flowchart or pseudocode. That level of verbosity will not help anyone to get a big picture of the system. And if anyone want to look into that level of details, source code will serve the purpose better.

Main

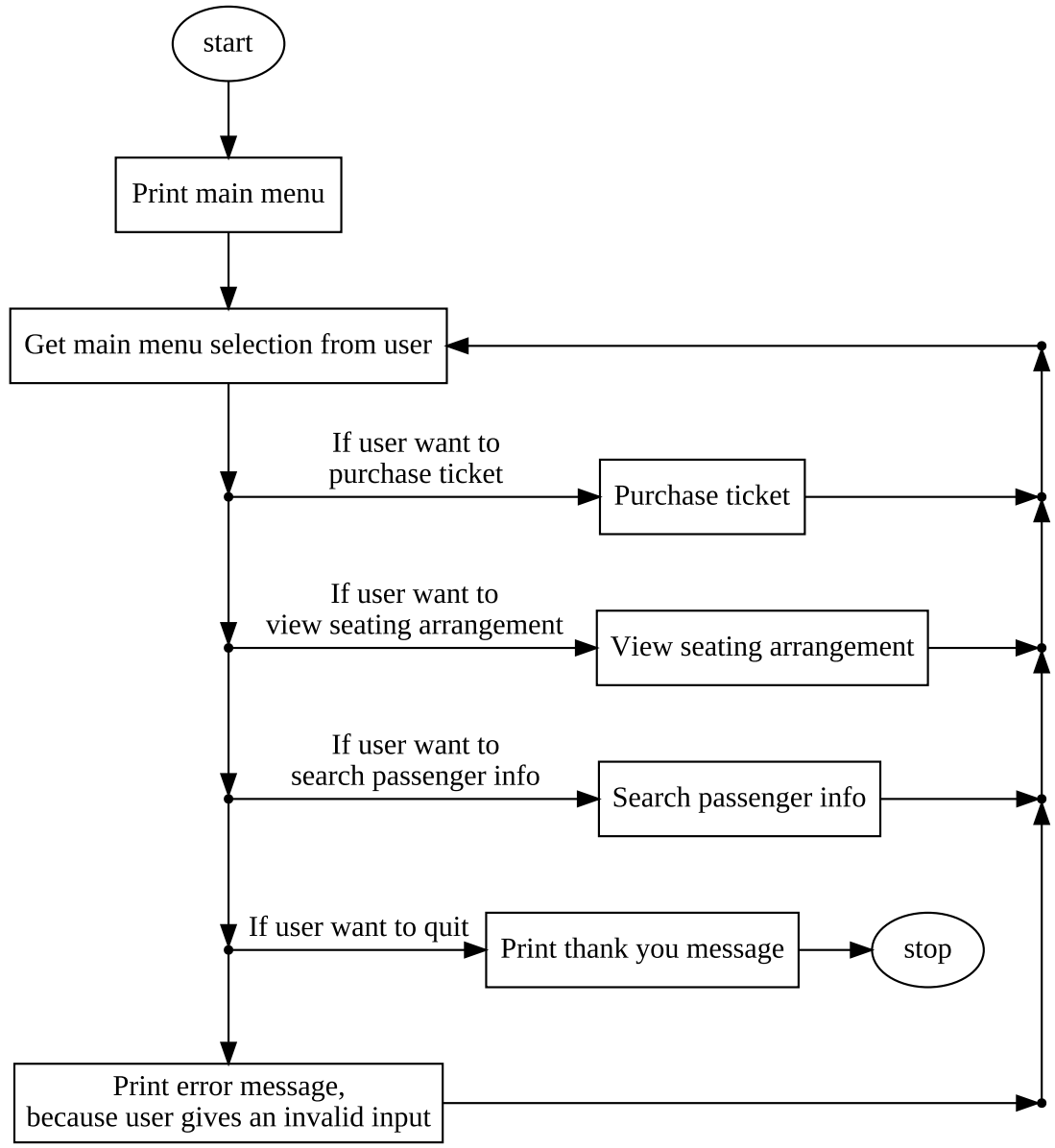


Figure 1: Main Flow

Purchase Ticket

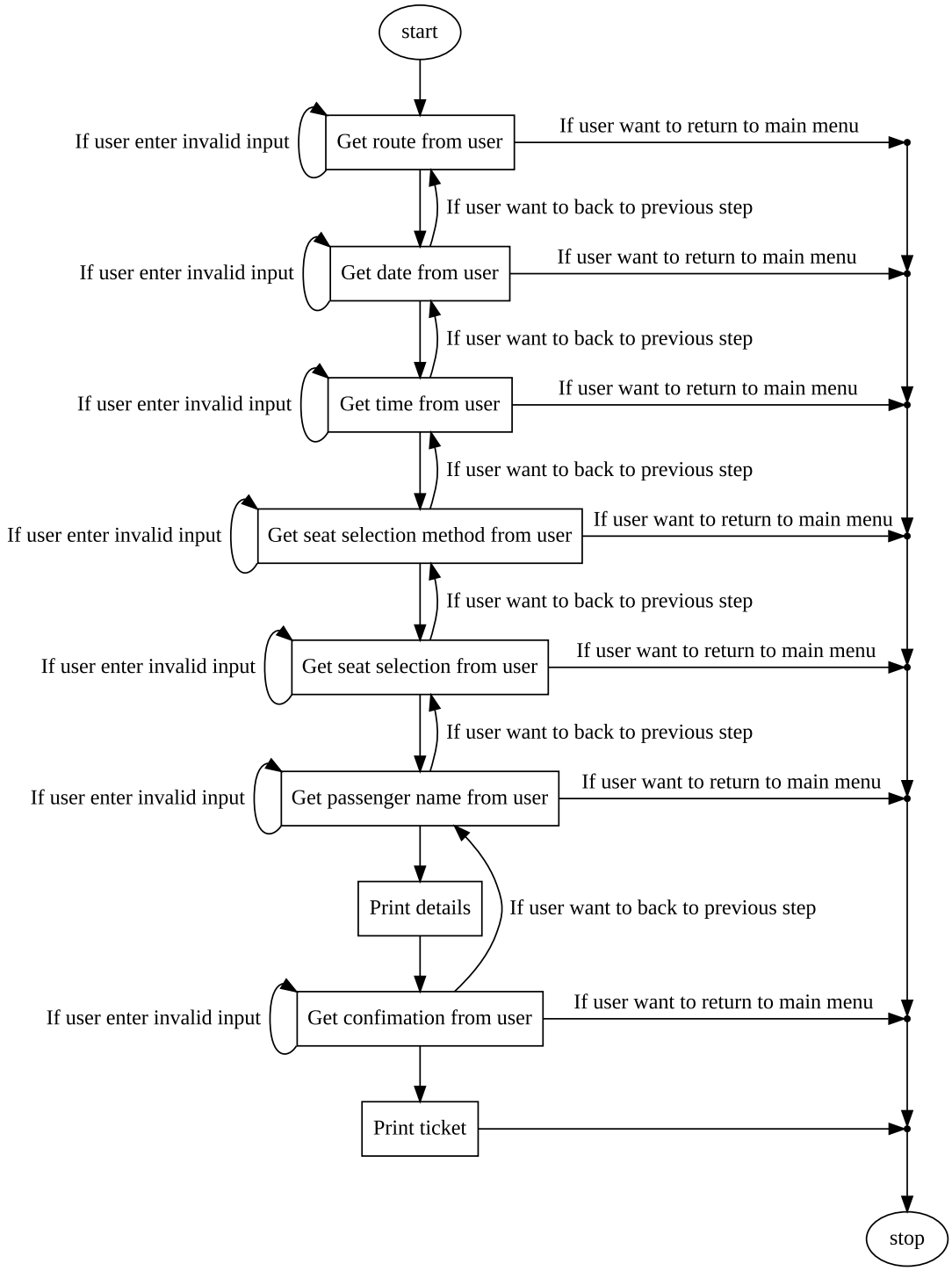


Figure 2: Purchase Ticket

View Seating Arrangement

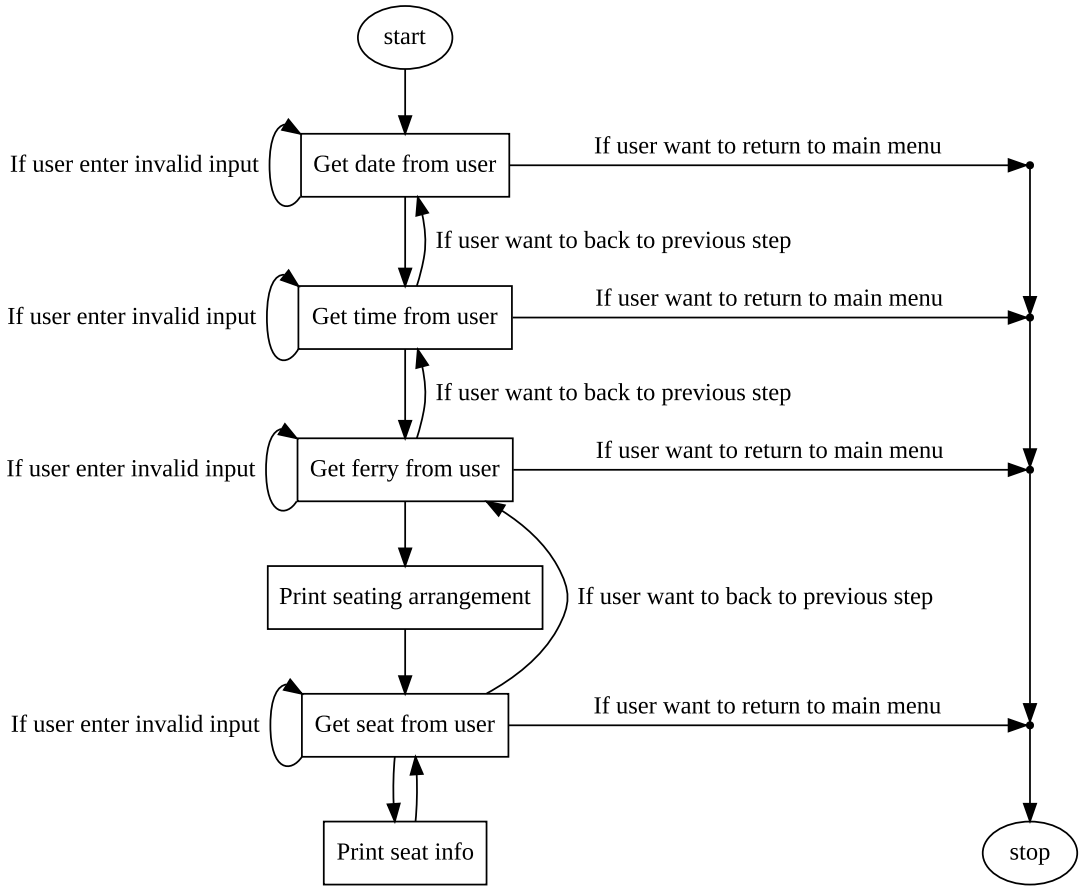


Figure 3: View Seating Arrangement

Search Passenger Info

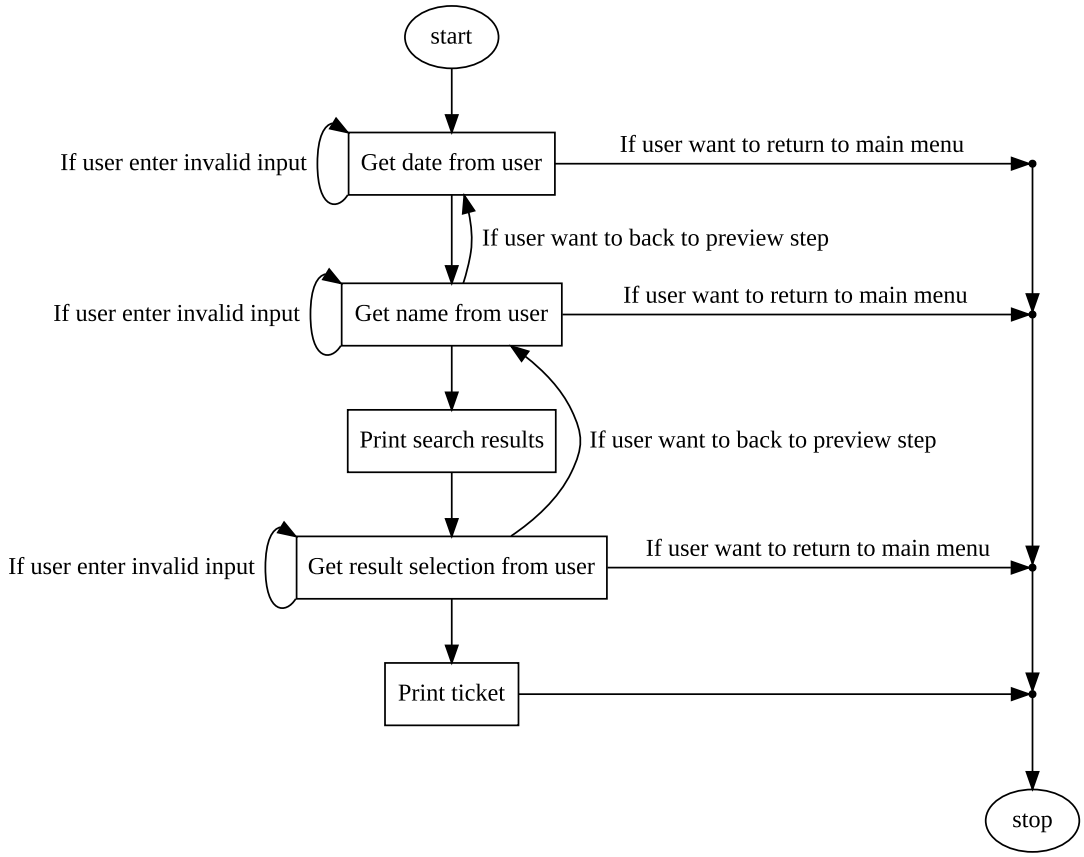


Figure 4: Search Passenger

Architectural Design

This section is about the system architecture.

Based on the flow design, it makes sense to write each step as a function. The question is, how to combine all these steps to form our system. Here is trivial architecture design:

```
PROGRAM
    print_main_menu()

    LOOP
        get_main_menu_selection_from_user()

        IF user_want_to_purchase_ticket THEN
            get_route_from_user()
            get_date_from_user()
            get_time_from_user()
            # ...
        ELSE IF user_want_to_view_seating_arrangement THEN
            # ...
        ELSE IF user_want_to_search_passenger_info THEN
            # ...
        ELSE IF user_want_to_quit THEN
            BREAK LOOP
        ELSE
            # User give an invalid input
            print_error_message()
        ENDIF
    ENDLOOP

    print_thank_you_message()
ENDPROGRAM
```

Let us focus on the first 3 steps purchasing ticket:

```
get_route_from_user()
get_date_from_user()
get_time_from_user()
# ...
```

This method is simple, but it does not support going back to previous step. To support it, loops need to be added:

```
LOOP
    get_route_from_user()
    IF user_want_to_back_to_previous_step THEN
        BREAK LOOP
    ENDIF
```

```
    LOOP
        get_date_from_user()
        IF user_want_to_back_to_previous_step THEN
            BREAK LOOP
        ENDIF

        LOOP
            get_time_from_user()
            IF user_want_to_back_to_previous_step THEN
                BREAK LOOP
            ENDIF

            # ...
        ENDLOOP
    ENDLOOP
ENDLOOP
```

And to support option to return to main menu on each step, multiple breaks are needed:

```
break_until_the_end = FALSE

LOOP
    get_route_from_user()
    IF user_want_to_back_to_previous_step THEN
        BREAK LOOP
    ENDIF
    IF user_want_to_return_to_main_menu THEN
        break_until_the_end = TRUE
        BREAK LOOP
    ENDIF

    LOOP
        get_date_from_user()
        IF user_want_to_back_to_previous_step THEN
            BREAK LOOP
        ENDIF
        IF user_want_to_return_to_main_menu THEN
            break_until_the_end = TRUE
            BREAK LOOP
        ENDIF

        LOOP
            get_time_from_user()
            IF user_want_to_back_to_previous_step THEN
                BREAK LOOP
            ENDIF
            IF user_want_to_return_to_main_menu THEN
                break_until_the_end = TRUE
                BREAK LOOP
            ENDIF
```

```
            # ...

            IF break_until_the_end THEN
                BREAK LOOP
            ENDIF
        ENDLOOP

        IF break_until_the_end THEN
            BREAK LOOP
        ENDIF
    ENDLOOP

    IF break_until_the_end THEN
        BREAK LOOP
    ENDIF
ENDLOOP
```

This is cumbersome. First, there are too many levels of nesting, which is hard to read. Second, the logic for checking intention of user is repeating, which violate the DRY (Don't Repeat Yourself) principle. Is there a better way? Is there a way to factor out those repeating lines?

Yes, there is.

A pattern can be observed from the code above:

```
do_a_step()
IF user_want_to_back_to_previous_step THEN
    go_to_previous_step()
ELSE IF user_want_to_return_to_main_menu THEN
    go_to_first_step() # The main menu step
ELSE
    go_to_next_step()
ENDIF
```

To implement that pattern, a list of steps can be created, and an index can be used to keep track of current step. After current step is done, index can be modified based on result of current step, and the corresponding step will be executed:

```
steps = list(
    main_menu,
    get_route_from_user,
    get_date_from_user,
    get_time_from_user,
    ...
)

i = 0
LOOP WHILE i >= 0
    current_step = steps[i]
```

```
    current_step()

    IF user_want_to_back_to_previous_step THEN
        i = i - 1
    ELSE IF user_want_to_return_to_main_menu THEN
        i = 0
    ELSE
        i = i + 1
    ENDIF
ENDLOOP
```

With this architecture, operation common for all steps can be added easily, such as printing a step list from main menu to current step. To make it more flexible, the `steps` list can be passed into each step, so that the list can be modified inside a step. To enable communication between steps, a context variable can be passed into each step.

```
PROGRAM
    steps = list(main_menu)
    context = dictionary()

    i = 0
    LOOP WHILE i >= 0
        current_step = steps[i]
        current_step(steps, context)

        IF user_want_to_back_to_previous_step THEN
            i = i - 1
        ELSE IF user_want_to_return_to_main_menu THEN
            i = 0
        ELSE
            i = i + 1
        ENDIF
    ENDLOOP
ENDPROGRAM
```

Sample Output

Starting Page

	When the system starts, the starting page will show up.

Main Menu

Main Menu P: Purchase Ticket V: View Seating Arrangement S: Search Passenger Q: Quit System Select an option:	After the starting page shows up, the main menu page will be prompted. User is required to input the desired option.
Main Menu P: Purchase Ticket V: View Seating Arrangement S: Search Passenger Q: Quit System Select an option: j Invalid input Select an option: k Invalid input Select an option: p Invalid input Exceeded trial limit. Quit system.	**Error** If the user key in the wrong input, 3 trials are given to the user to key in correctly. If all 3 trials are exceeded, an error message is prompted, then the system will quit itself.

Purchase Ticket

Main Menu > Purchase Ticket > Route P: Penang to Langkawi L: Langkawi to Penang R: Return to Main Menu Select a route:	**Route Selection Page** After the user input P in the main menu, the route selection page will be prompted. User is required to choose which route to travel.
Main Menu > Purchase Ticket > Route > Date 0: 2018-12-30 1: 2018-12-31 2: 2019-01-01 3: 2019-01-02 4: 2019-01-03 5: 2019-01-04 6: 2019-01-05 7: 2019-01-06 B: Back R: Return to Main Menu Select a date:	**Date Selection Page** After the user choose the route, the date selection page will be prompted. User is required to choose which date to travel.
Main Menu > Purchase Ticket > Route > Date > Time 0: 10am (Seats left: 40 business, 158 economy) 1: 11am (Seats left: 40 business, 160 economy) 2: 12pm (Seats left: 35 business, 149 economy) 3: 1pm (Seats left: 40 business, 160 economy) 4: 2pm (Seats left: 40 business, 160 economy) 5: 3pm (Seats left: 40 business, 160 economy) 6: 4pm (Seats left: 40 business, 160 economy) 7: 5pm (Seats left: 40 business, 160 economy) B: Back R: Return to Main Menu Select a time:	**Time Selection Page** After the user choose the date, the time selection page will be prompted. User is required to choose which time to travel. In this page, user can quickly check the seats availability for each trip time.
Main Menu > Purchase Ticket > Route > Date > Time > Method Selection S: Select seats manually A: Auto-assign seats for me B: Back R: Return to Main Menu Select a method:	**Method Selection Page** After the user choose the time, the method selection page will be prompted. User is required to choose which method to use to select the seats.

<table>
<tr>
<td>

```
Main Menu > Purchase Ticket > Route > Date > Time > Method Selection > Seats

    Ferry 001          Ferry 002          Ferry 003          Ferry 004
+----------------+ +----------------+ +----------------+ +----------------+
| A1 A2 A3 __ __ | | A1 A2 A3 A4 A5 | | A1 A2 A3 A4 A5 | | A1 A2 A3 A4 A5 |
| B1 B2 B3 B4 B5 | | B1 B2 __ __ __ | | B1 B2 B3 B4 B5 | | B1 B2 B3 B4 B5 |
+----------------+ +----------------+ +----------------+ +----------------+
| __ __ C3 C4 C5 | | __ __ __ __ __ | | C1 C2 C3 C4 C5 | | C1 C2 C3 C4 C5 |
| D1 D2 __ __ __ | | D1 D2 D3 D4 D5 | | D1 D2 D3 D4 D5 | | D1 D2 D3 D4 D5 |
| E1 E2 E3 E4 E5 | | E1 E2 E3 E4 E5 | | E1 E2 E3 E4 E5 | | E1 E2 E3 E4 E5 |
| F1 F2 F3 F4 F5 | | F1 F2 F3 F4 F5 | | F1 F2 F3 F4 F5 | | F1 F2 F3 F4 F5 |
| G1 G2 G3 G4 G5 | | G1 G2 G3 G4 G5 | | G1 G2 G3 G4 G5 | | G1 G2 G3 G4 G5 |
| H1 H2 H3 H4 H5 | | H1 H2 H3 H4 H5 | | H1 H2 H3 H4 H5 | | H1 H2 H3 H4 H5 |
| I1 I2 I3 I4 __ | | I1 I2 I3 I4 I5 | | I1 I2 I3 I4 I5 | | I1 I2 I3 I4 I5 |
| J1 J2 J3 J4 J5 | | J1 J2 J3 J4 J5 | | J1 J2 J3 J4 J5 | | J1 J2 J3 J4 J5 |
+----------------+ +----------------+ +----------------+ +----------------+

Enter seats in the format: <ferry_id> <seat>... <ferry_id> <seat>...
For example: 001 A1 A2 G1 002 C1 D1 E1 F1 003 J5

To go back to previous step, enter "back".
To return to main menu, enter "return".

Enter seats:
```

</td>
<td>

Manual Seat Selection Page

If the user chooses to select seats manually, this manual seat selection page will be prompted. User is required to key in desired seats that are available.

- A1 to B5 is business class.
- C1 to J5 is economy class.

If that particular seat is not available, seat number will be replaced with ___.

</td>
</tr>
<tr>
<td>

```
Enter seats: 001
Please choose at least one seat
Enter seats: A1
Please specify ferry ID
Enter seats: 001 C1
Ferry 001 seat C1 is occupied.
Enter seats: 006
Please choose at least one seat
Enter seats: k
Invalid seat number "k"
Enter seats: 001 G1 G2 G3 003 002 A4 A5 003 C1 C2
```

</td>
<td>

A proper error message will be shown according to 5 scenarios:

1. If the user only key in ferry ID
2. If the user only key in seat number
3. If the user key in the occupied seat number
4. If the user key in ferry ID that is not available
5. If the user key in invalid input

The system will keep asking user to key in seat number until the inputs are correct.

</td>
</tr>
<tr>
<td>

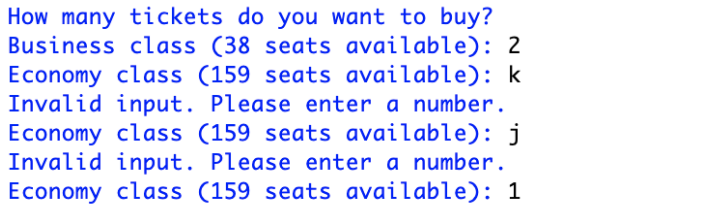
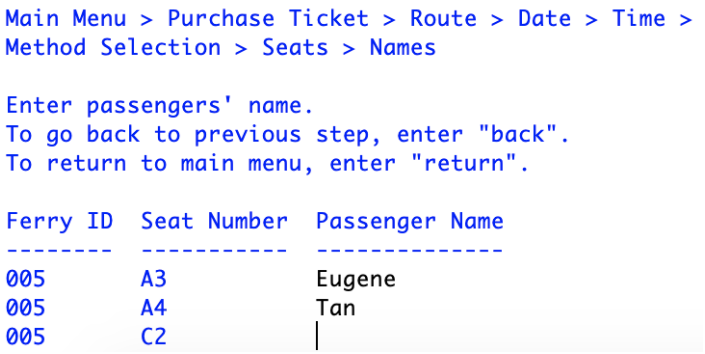
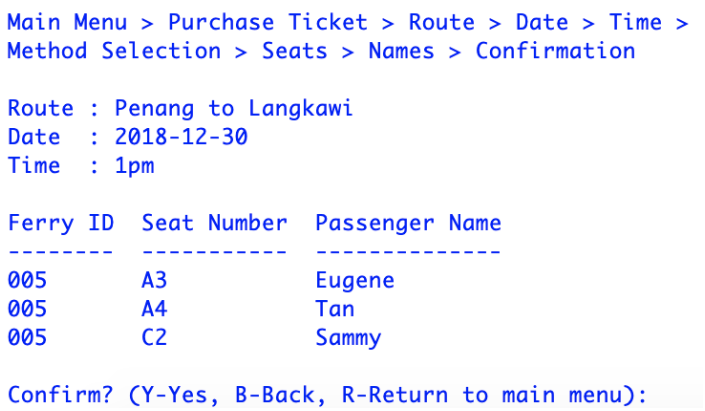
```
Main Menu > Purchase Ticket > Route > Date > Time >
Method Selection > Seats

To go back to previous step, enter "back".
To return to main menu, enter "return".

How many tickets do you want to buy?
Business class (40 seats available): 2
Economy class (160 seats available):
```

</td>
<td>

Auto Seat Selection Page

If the user chooses to select seats automatically, this auto seat selection page will be prompted. User is required to key in desired number of business class seats and economy class seats.

</td>
</tr>
</table>

	**Error** If the user key in an invalid input, an error message is prompted. The system will keep asking user to key in until the inputs are correct.
	**Passenger Name Page** After the user finish choosing the seats, the passenger name page will be prompted. User is required to key in the passenger's name for each seats selected from the seat selection page.
	**Confirmation Page** After the user finishes entering the passengers' names, the confirmation page will be prompted. User is required to check whether the details are correct.

```
Main Menu > Purchase Ticket > Route > Date > Time >
 Method Selection > Seats > Names > Confirmation >
Print Ticket

    +----------------+--------------------------+
    |          Route | Penang to Langkawi       |
    |           Date | 30 Dec 2018 (Sun)        |
    |           Time | 1pm                      |
    |       Ferry ID | 005                      |
    |        Seat No | A3                       |
    | Passenger Name | Eugene                   |
    |   Purchased At | 2018-12-30 20:12:04      |
    +----------------+--------------------------+

    +----------------+--------------------------+
    |          Route | Penang to Langkawi       |
    |           Date | 30 Dec 2018 (Sun)        |
    |           Time | 1pm                      |
    |       Ferry ID | 005                      |
    |        Seat No | A4                       |
    | Passenger Name | Tan                      |
    |   Purchased At | 2018-12-30 20:12:04      |
    +----------------+--------------------------+

    +----------------+--------------------------+
    |          Route | Penang to Langkawi       |
    |           Date | 30 Dec 2018 (Sun)        |
    |           Time | 1pm                      |
    |       Ferry ID | 005                      |
    |        Seat No | C2                       |
    | Passenger Name | Sammy                    |
    |   Purchased At | 2018-12-30 20:12:04      |
    +----------------+--------------------------+


Main Menu

P: Purchase Ticket
V: View Seating Arrangement
S: Search Passenger
Q: Quit System

Select an option:
```

Printing Page

After user had confirmed all the details are corrected, this printing page will be prompted, and the customer's ticket will be printed out. The system will automatically return to main menu.

View Seating Arrangement

```
Main Menu > View Seating Arrangement > Date

0: 2018-12-30
1: 2018-12-31
2: 2019-01-01
3: 2019-01-02
4: 2019-01-03
5: 2019-01-04
6: 2019-01-05
7: 2019-01-06
B: Back
R: Return to Main Menu

Select a date:
```

Date Selection Page

After the user input V in the main menu, the date selection page will be prompted. User is required to choose which date to view the seating arrangement.

```
Main Menu > View Seating Arrangement > Date > Time

0: 10am
1: 11am
2: 12pm
3: 1pm
4: 2pm
5: 3pm
6: 4pm
7: 5pm
B: Back
R: Return to Main Menu

Select a time:
```

Time Selection Page

After the user input the date, the time selection page will be prompted. User is required to choose which trip time to view the seating arrangement.

```
Main Menu > View Seating Arrangement > Date > Time
> Ferry

0: 001
1: 002
2: 003
3: 004
4: 005
5: 006
6: 007
7: 008
B: Back
R: Return to Main Menu

Select a ferry:
```

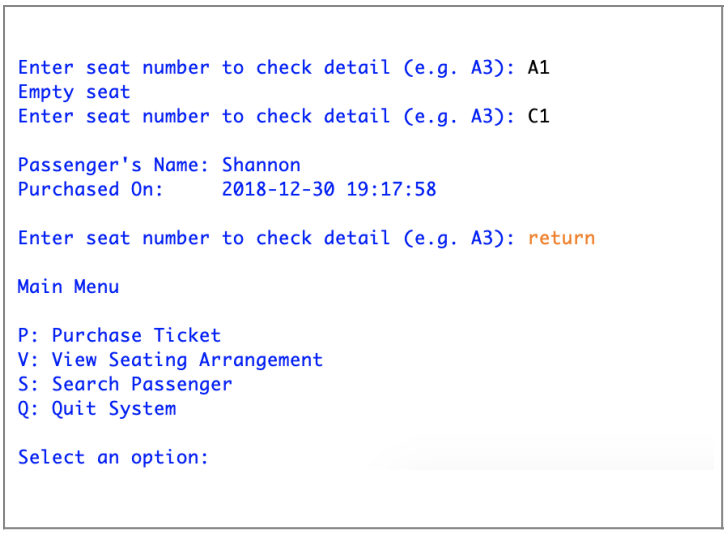
Ferry ID Selection Page

After the user input the trip time, the ferry ID selection page will be prompted. User is required to choose which ferry ID's seating arrangement to view.

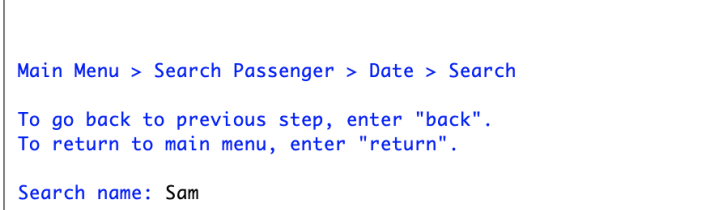
```
Main Menu > View Seating Arrangement > Date > Time > Ferry >
Seat Number

   ************************************************
   * Ferry 001   Date: 30 Dec 2018 (Sun)   Time: 12pm *
   ************************************************
   *    1    *    2    *    3    *    4    *    5    *
   ************************************************
   * BUSINESS                                       *
   ************************************************
A  *    0    *    0    *    0    *    1    *    1    *
   ************************************************
B  *    0    *    0    *    0    *    0    *    0    *
   ************************************************
   * ECONOMY                                        *
   ************************************************
C  *    1    *    1    *    0    *    0    *    0    *
   ************************************************
D  *    0    *    0    *    1    *    1    *    1    *
   ************************************************
E  *    0    *    0    *    0    *    0    *    0    *
   ************************************************
F  *    0    *    0    *    0    *    0    *    0    *
   ************************************************
G  *    1    *    1    *    1    *    0    *    0    *
   ************************************************
H  *    0    *    0    *    0    *    0    *    0    *
   ************************************************
I  *    0    *    0    *    0    *    0    *    1    *
   ************************************************
J  *    0    *    0    *    0    *    0    *    0    *
   ************************************************

To go back to previous step, enter "back".
To return to main menu, enter "return".

Enter seat number to check detail (e.g. A3):
```

Seat Arrangement Page

After the user select the ferry ID, the seating arrangement of that particular ferry ID will be shown. User can either key in the seat number to check passenger's details or go back to previous back or return to main menu.

	If the user input an empty seat, a proper message such as “Empty seat” will be shown. If the user input an occupied seat, the purchase details of that passenger will be shown. User can return to main menu by keying in `return`

Search Passenger

	**Date Selection Page** After the user input S in the main menu, the date selection page will be prompted. User is required to choose which date to search.
	**Name Search Page** After the user choose a date, the name search page will be prompted. User is required to enter the name for searching.

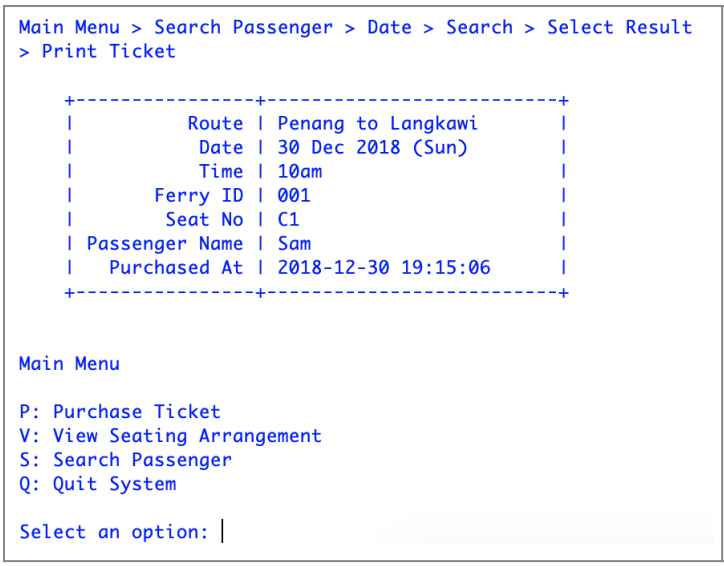
```
Main Menu > Search Passenger > Date > Search > Select Result

0:      Name          : Sam
        Route         : Penang to Langkawi
        Date and Time : 30 Dec 2018 (Sun) 10am
        Ferry ID      : 001
        Seat Number   : C1
        Purchased On  : 2018-12-30 19:15:06

1:      Name          : Sam
        Route         : Penang to Langkawi
        Date and Time : 30 Dec 2018 (Sun) 12pm
        Ferry ID      : 001
        Seat Number   : D5
        Purchased On  : 2018-12-30 19:17:58

2:      Name          : Sam
        Route         : Penang to Langkawi
        Date and Time : 30 Dec 2018 (Sun) 12pm
        Ferry ID      : 001
        Seat Number   : G1
        Purchased On  : 2018-12-30 19:50:51

B: Back
R: Return to Main Menu

Select a result: 0

Main Menu > Search Passenger > Date > Search > Select Result
> Print Ticket

    +----------------+--------------------------+
    |          Route | Penang to Langkawi       |
    |           Date | 30 Dec 2018 (Sun)        |
    |           Time | 10am                     |
    |       Ferry ID | 001                      |
    |        Seat No | C1                       |
    | Passenger Name | Sam                      |
    |   Purchased At | 2018-12-30 19:15:06      |
    +----------------+--------------------------+


Main Menu

P: Purchase Ticket
V: View Seating Arrangement
S: Search Passenger
Q: Quit System

Select an option:
```

Select Result Page

After the user enter the name, if the name can be found in the passenger list, the details of that particular name will be shown. User can choose to reprint the tickets or back to previous page or return to main menu.

```
Main Menu > Search Passenger > Date > Search

To go back to previous step, enter "back".
To return to main menu, enter "return".

Search name: Henry

Main Menu > Search Passenger > Date > Search > Select Result

B: Back
R: Return to Main Menu

Select a result:
```

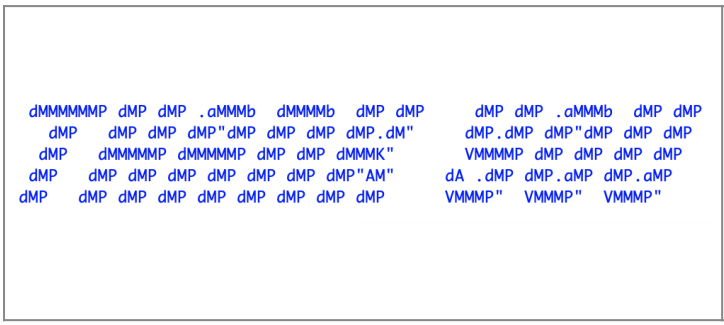
If the name searched is not in the passenger list, nothing will be shown.



Printing Page

If the user chooses to reprint the ticket, the printing page will be prompted, the ticket will be printed out. The system will automatically return to main menu.

Quit Page



When the user input Q in the main menu or the system want to quit itself due to exceeding 3 trail limits, the quit page will be prompted indicating that the system has been shut down.

Error

Main Menu > Purchase Ticket > Route P: Penang to Langkawi L: Langkawi to Penang R: Return to Main Menu Select a route: j Invalid input Select a route: k Invalid input Select a route: l Invalid input Exceeded trial limit. Return to main menu. Main Menu P: Purchase Ticket V: View Seating Arrangement S: Search Passenger Q: Quit System Select an option:	3 trials will be given if the user key in the wrong input in the following page: 1. Route Selection Page 2. Date Selection Page 3. Time Selection Page 4. Method Selection Page 5. FerryID selection page 6. Select Result Page If the user still key in the wrong input after 3 times, the system will automatically return to main menu.

Additional Python Features Used

Language Features

List comprehension

List comprehension is a concise way to create a list. The basic syntax is

```
x = [expression for item in list if condition]
```

It is equivalent to

```
x = []
for item in list:
    if condition:
        x.append(expression)
```

For example, to create a list of 10 squared numbers start from zero:

```
squared_numbers = [x**2 for i in range(10)]
```

Module

In Python, function definitions can be put inside a file. That file is called a module. Module can be imported to be used in a script. For example, a module named `my_module.py` has the following content

```
def add(x, y):
    return x + y
```

`my_module.py` can be imported to be used.

```
import my_module

print(my_module.add(5, 7))
```

Module helps programmers to organize their code for higher reusability.

Dictionary

Dictionary is a built-in data type of Python like list. It is called associative arrays in other languages. To access an element of a list, we need an index. To access an element of a dict, we need a key. Dictionary key can be any immutable type.

Dictionary can be created like this:

```
x = {'jack': 4098, 'sape': 4139, 'susan': 4359}
```

And can be accessed like this:

```
print(x['jack'])
```

String formatting

There is a `str.format()` method for string formatting.

```
'Hello, {}'.format('Lisa') # evaluate to 'Hello, Lisa'
```

It has several formatting syntax, such as specify width and alignment.

```
'| {: <5} | {: ^5} | {: >5} |'.format('a', 'b', 'c')
# evaluate to '| a     |   b   |     c |'
```

Built-in Functions

`callable(x)` test whether `x` is a function.

`isinstance(x, type)` test whether `x` is type of `type`.

`enumerate(l)` return a list of tuple of `i` and `x`, where `i` is index, and `x` is `l[i]`. Normally this function is used in `for` loop.

`sum(l)` return summation of all elements of list `l`.

Libraries

`time` module provides various time-related functions. `time.time()` will return number of seconds since epoch, also known as timestamp. `time.localtime(timestamp)` will convert a timestamp to time structure. `time.strftime(format, time_structure)` will convert a time structure to a string according to format given.

`os` module provides a portable way of using operating system dependent functionality. `os.mkdir(directory, mode)` can be used to create a folder.

`os.path` module provides some useful function on pathnames. `os.path.isdir(x)` check whether x is a directory. `os.path.join(a, b, ...)` will join one or more path component using operating system specific path separator.

`datetime` module supply classes for manipulating dates and times. `datetime.date.today()` will return today's date.

`json` module provide functions to parse variable from (`json.load(f)`) and to (`json.dump(x, f)`) string JSON data format.

References

Gillespie, P. (2018) *Text to ASCII art generator (TAAG).* [Online]. Available from: http://patorjk.com/software/taag/ [Accessed: 28 December 2018]

Python (2018) *The Python standard library.* [Online]. Available from: https://docs.python.org/3/library/index.html [Accessed: 30 December 2018]

Python (2018) *The Python tutorial.* [Online]. Available from: https://docs.python.org/3/tutorial/index.html [Accessed: 30 December 2018]