code.sprint MT

Preparatory Workbook 2025

Your starting point towards Codesprint Malta

GOVERNMENT OF MALTA
MINISTRY FOR EDUCATION, SPORT
YOUTH, RESEARCH AND INNOVATION

Table of Contents

Welcome to CODESPRINTmt 2

Workbook Guidelines. 3

Grading Distribution 4

Section A 5

- Task 1: Nuclear Power Plant Access Area 5
- Task 2: Lotto Ticket 7
- Task 3: Point-of-Sale System 8
- Task 4: CodeLift: Rise to the Elevator Simulation Challenge! 10

Assessment Criteria 14

Section B 16

- Self-Reflection Exercise 16

Final Presentation Guidelines 17

Welcome to CODESPRINTmt

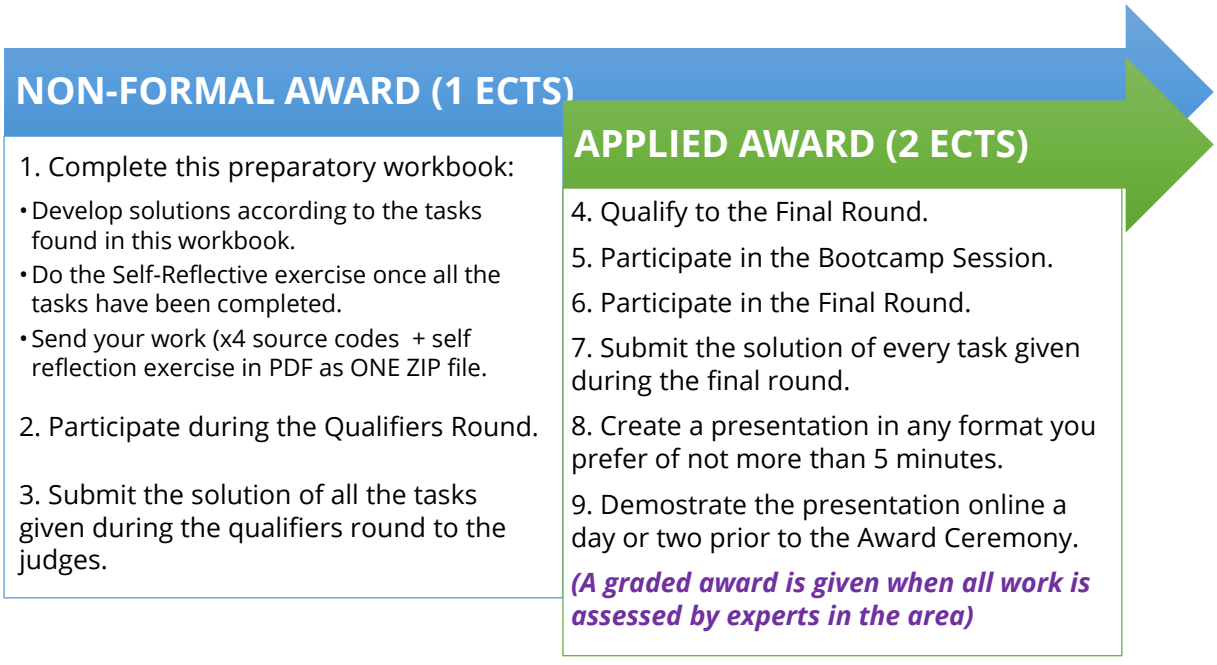
Join **Code.Sprint Malta,** a timed coding challenge that puts your problem-solving / coding skills to the test. Besides the coding fun, you will have the opportunity to meet coding enthusiasts and sharpen your coding competitive skills while having the chance to win amazing prizes.

Code.Sprint^{MT} is divided into two phases: 1) Qualifiers Round, and 2) Finals Round. The top 10 participants from the qualifiers round will pass on to the final round.

During both rounds:

- you will be given a task to complete within a fixed time.
- you are to carry out the tasks on an individual basis.
- a judging panel will give points to your work according to an established criteria rubric.
- a detailed assessment rubric will be given during the contest. Therefore, you will be aware of what the judges are expecting from your work.

Moreover, Code.Sprint^{MT} is **accredited at MQF Level 3,** thus offering you the opportunity to top up your list of academic achievements. You can be awarded either a **1 ECTS Non-Formal MQF3 Award** (not graded) or a **2 ECTS Applied MQF3 Award** which is graded as explained in section Grading Distribution.

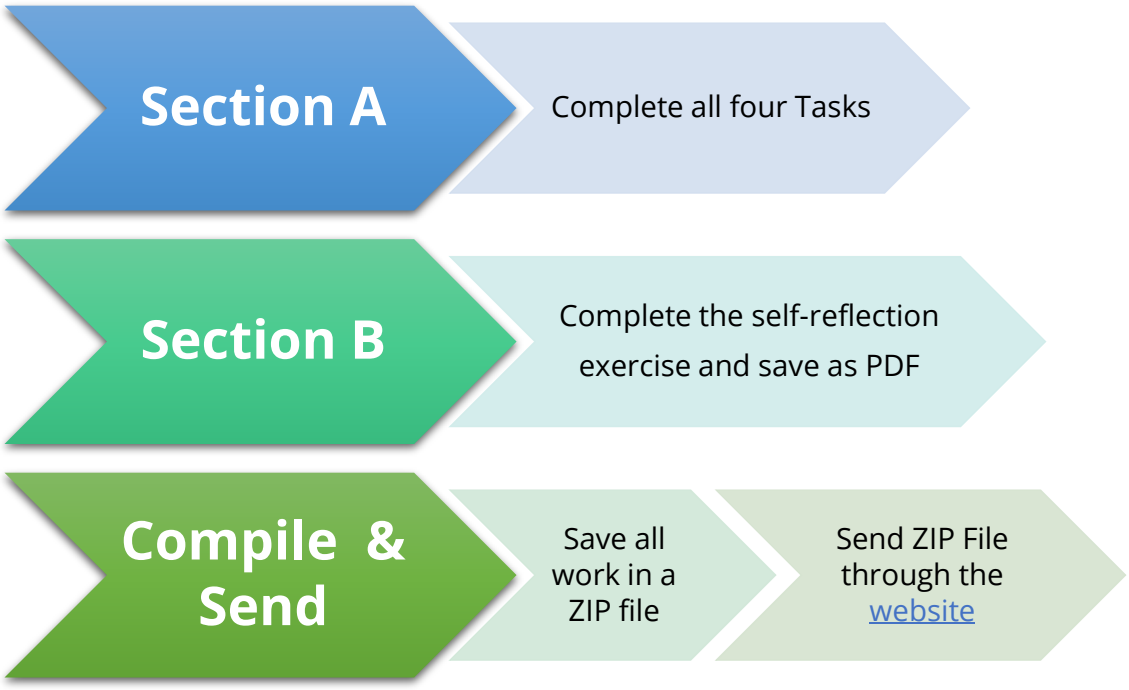


Workbook Guidelines.

This workbook includes:

- Four tasks in Section A
- Assessment Criteria for every task
- Self-Reflection Exercise in Section B
- Guidelines for the Final Presentation *(requirement for the 2 ECTS Applied MQF3 Award)*

Participants are expected to complete all the tasks presented in section A. Besides the development of the source code, participants are also expected to complete the self-reflection exercise in Section B.



ALL WORK MUST BE SAVED IN **ONE ZIP FILE** AND SENT BY THE CLOSING DATE AS INDICATED IN ACCREDITATION SECTION IN THE WEBSITE.

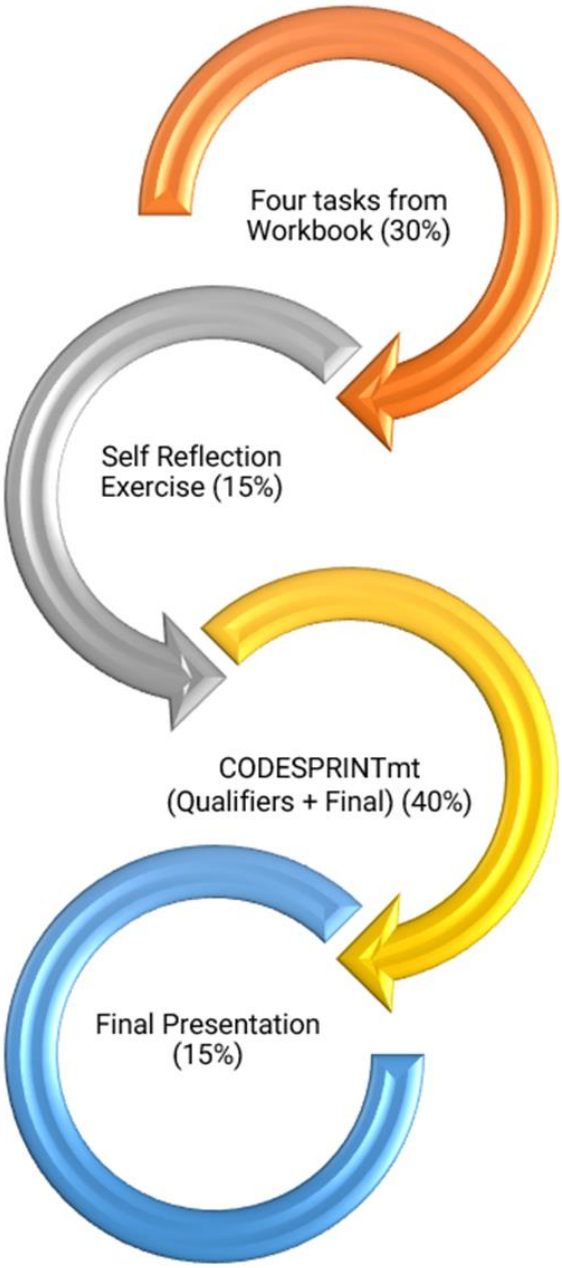
The Zip filename should be your name_surname_school.pdf
e.g. alexia_brincat_stignatiuscollege.zip

Grading Distribution

The 2 ECTS Applied MQF 3 Certification is graded as follows:

0 - 39	40 - 59	60 - 89	90 - 100
FAIL	PASS	MERIT	DISTINCTION

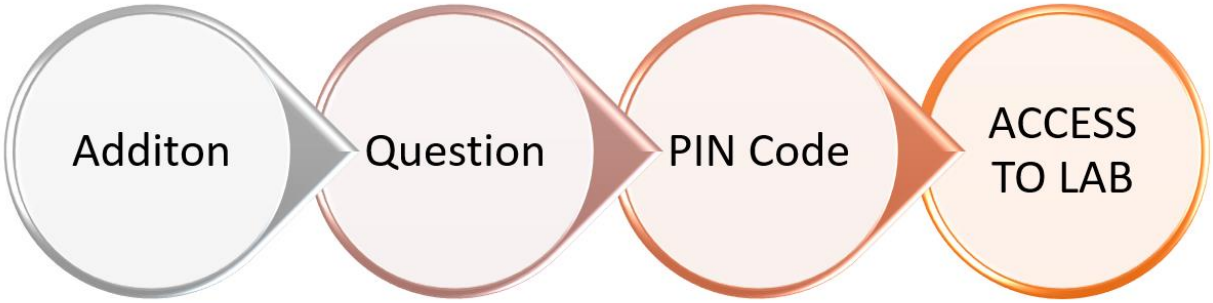
The marks from each phase of the event contribute to a global mark. The distribution of marks is as follows:



Section A

Task 1: Nuclear Power Plant Access Area

A scientist can access the Nuclear Powerplant lab by going through a three-tiered validation protocol as in the diagram below:



Step 1: Addition

The scientist must answer correctly a randomly generated addition problem with two numbers. The values to be randomised are between 0 and 9; e.g. 3+7 = ? where 3 and 7 are randomly generated.

Step 2: Question

The scientist must answer correctly a multiple-choice-answer question which is randomised from a list of three pre-set questions. The pre-set questions & answers are listed below; the correct answers are enclosed in a blue box.

Pre-set Questions

Question 1:

Which element is used as fuel in a nuclear power stations?

A: Water	B: Gas	C: Uranium

Question 2:

Which country uses the most nuclear power?

A: The United States	B: Russia	C: France

Question 3:

Which country opened the first nuclear power plant in 1954 known as 'Atom Mirny'?

A: North Korea	B: The Soviet Union	C: Japan

Step 3: PIN Code

The scientist must enter a four-numbered PIN Code which is the constant value 6502.

Log-In Access Flow

From one validation step to another the program will not provide any feedback to the scientist. At the end of the validation process, the scientist will be granted or denied access to the lab by displaying a message accordingly.

Name your program **task1_name.py**
(e.g. task1_ john.py)

Task 2: Lotto Ticket

The lottery is a game in which players pay for a ticket, select a group of numbers and win prizes based on how they match the drawn results.

Write a program that simulates a lottery system according to rules below:

- The lottery system has a set of numbers available which is from 1 to 45.
- The lottery prize is that of €500,000.
- Five lottery numbers are randomly drawn and are not visible to the user.
  Proper validation is required to avoid duplicate numbers.
- The user is asked to purchase a lottery ticket.
- A lottery ticket is made up of five non-duplicate numbers.
  Proper validation is required to avoid non-valid and duplicate numbers.
- According to the numbers guessed, a prize is won:
  - With three numbers guessed, the user wins 10% of the lottery prize.
  - With four numbers guessed, the user wins 25% of the lottery prize.
  - With five numbers guessed, the user wins lottery prize in full.

The result screen should display the numbers drawn, the number of correct guesses, and the prize won (if applicable).

Name your program **task2_name.py**
(e.g. task2_ john.py)

Task 3: Point-of-Sale System

POS systems have replaced most traditional cash registers due to their ability to connect to the retailer's main database system. Having all data stored and accessible within one system makes daily operations more efficient and more profitable.

Write a program that simulates a POS system. This system has a Log-In Screen and a Main Menu as shown below:

Log In Screen	
Option	**Description**
Cashier Log In	A cashier must be logged-in to proceed to the Main Menu. *(Details of the registered cashiers is shown in **Table 2**)*
Exit	Terminates the program.

Main Menu	
Option	**Description**
Enter new transaction	Allows the cashier to enter the items that the client wants to buy. *(A list of the stock items is shown in **Table 1**)*
Issue Receipt	This option displays the receipt of the last transaction carried out. *(A sample is shown in **Figure 1**)*
Display Stock List	This option displays the list of items that a client can buy. *(A list of the stock items is shown in **Table 1**)*
Cashier Sign Out	Allows a logged-in cashier to sign out and return to the Log-In Screen. *(Details of the current registered cashiers is shown in **Table 2**)*

Stock List	
Items	Price
Printer	€67.99
Monitor	€138.00
Keyboard	€12.50
Graphics Card	€114.99
Soundbar	€249.00
Hard Disk	€66.95
Headset	€17.55
Smartwatch	€135.00
Camcorder	€329.00
Drone	€449.99

Table 1

```
-----------------------------------
 << EASY-SAVE SUPERMARKET >>
 102, Flower Street, Filfla
***********************************
 Sun Jan 20 11:48:45 CET 2019
 Receipt: 66432020
 Cashier: mallia.amy
***********************************
 MONITOR                 Eur 138.00
 HEADSET                 Eur 17.55
 SOUNBAR                 Eur 249.00


 SUBTOTAL:               EUR 404.55
 VAT:                    Eur 72.82
 TOTAL:                  Eur 477.37

***********************************
          FISCAL RECEIPT
            THANK  YOU
------------ CUT-HERE --------------
```

Figure 1: Sample Generated Receipt

Registered Cashiers	
Username	Password
borg.steve	Borg87max
zammit.rita	RitPopSing!
agius.john	ToyotaBeSt
vella.carlos.02	Rock!n!Roll
mallia.amy	$Gaga$Lady$

Table 2

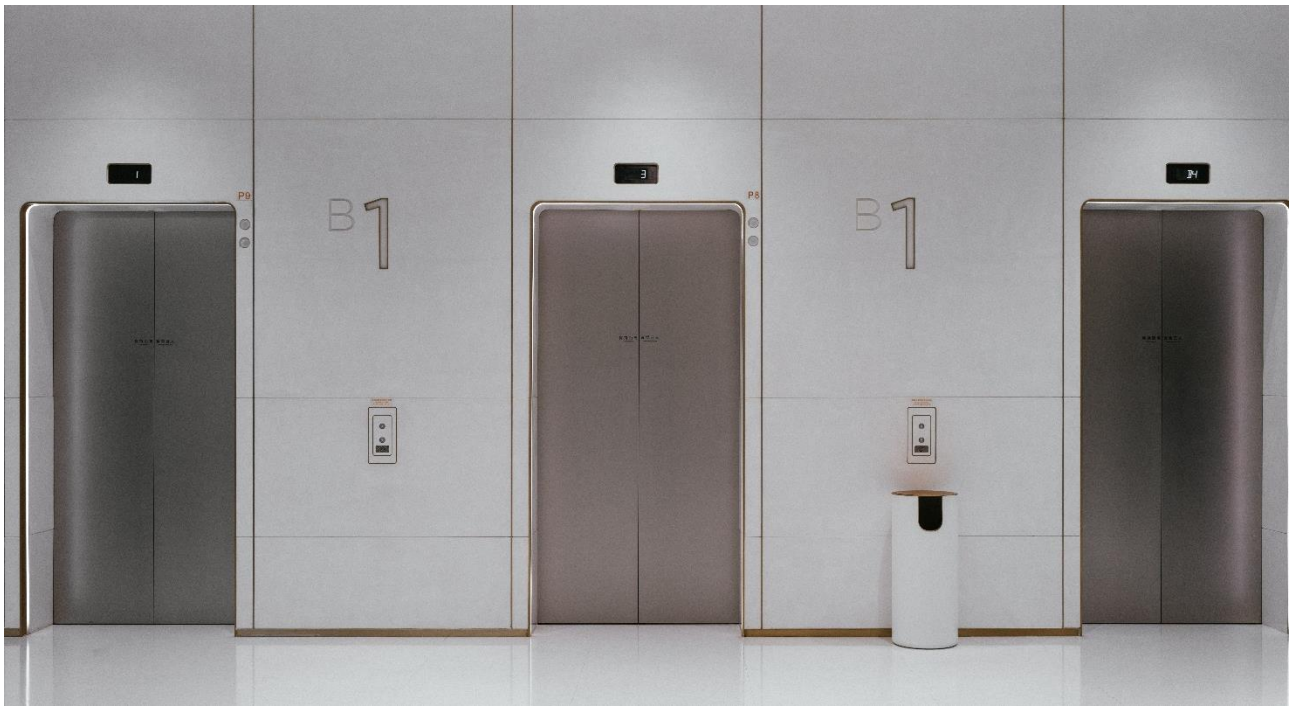
Name your program
task3_name.py
(e.g. task3_ john.py)

Code Hint: The code below displays the current date and time.

```
import datetime

# Get the current date and time
current_datetime = datetime.datetime.now()

# Format the date and time (e.g., DD/MM/YYYY HH:MM:SS)
formatted_datetime = current_datetime.strftime("%d/%m/%Y
%H:%M:%S")

# Display the formatted date and time
print("Receipt Date and Time:", formatted_datetime)
```

Task 4: CodeLift: Rise to the Elevator Simulation Challenge!

In the bustling world we live in, elevators are essential for vertical transportation. To ensure their smooth operation, sophisticated software systems govern their movement, safety features, and user interactions.



In CodeLift, your task is to simulate an elevator software using the Python programming language. Design a text-based program that simulates a hotel elevator system to transport passengers between the following floors, considering a maximum passengers capacity limit of four (4) passengers:

- Garage Floors (-2 and -1): Entry/exit for hotel guests.
- Ground Floor (Reception): Hotel access point.
- Guest Room Floors (1-5): Transport to guest rooms.

Functionality #1: Elevator Control Interface

1. The control panel prompts to enter the number of passengers who are entering the elevator whenever the elevator door opens. Inputting [X] indicates that no passengers are getting on the lift.
2. Upon entering the elevator, passengers will be asked to input their desired floor. The available floor options are: [-2] [-1] [0] [1] [2] [3] [4] [5].

Functionality #2: Floor Movement

1. The elevator moves between floors based on passengers' choice. The lift should follow the sequence of floors entered by the passengers.

   For example, if passenger 1 enters [3], passenger 2 enter [-2], and passenger 3 enters [4], the elevator will start by going to floor 3, then to floor -2 and then to floor 4.

2. When the elevator stops on a floor chosen by the passenger, it is assumed that the respective passenger/s exit the elevator automatically.

   For instance, let's imagine there are three passengers inside the elevator with the following floor choices: passengers 1 and 2 select floor [4], while passenger 3 chooses floor [2]. As the elevator reaches floor 4, passengers 1 and 2 will automatically exit the lift, resulting in an updated passenger count of one.

3. If there are no passengers entering the lift, i.e. if the user presses [X], the elevator proceeds to the next floor if there are still passengers in it.

4. When there are no passengers in the elevator, the elevator remains waiting at the last floor it stopped on, awaiting further instructions.

5. Each floor transition takes 1 second to complete.

Code Hint: the below code displays the string 'Before Delay', then it waits for 2 seconds and then displays the string 'After Delay'. This sample code can be used to create a visual of the elevator movement.

```
import time

print('Before Delay)
time.sleep(2)
print('After Delay')
```

Functionality #3: Lift Capacity

1. Track the number of passengers currently inside the lift.
2. Limit the number of passengers in the lift to a maximum of four (4) at any point in time.

Functionality #4: Validation Processes

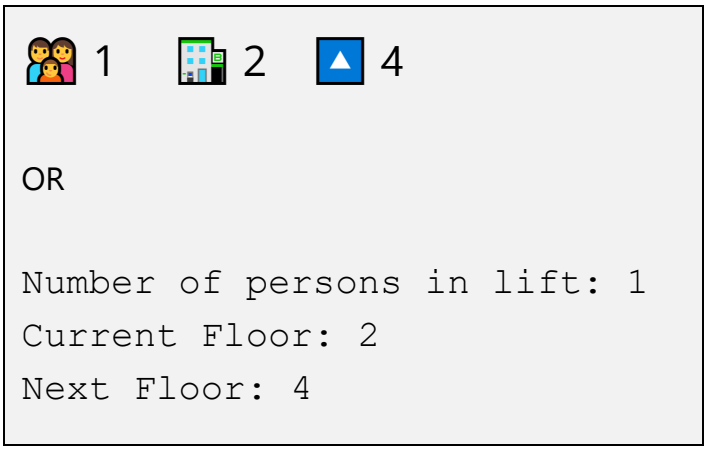
1. Validation is required on the instructions inputted by the user:
   - Number of passengers entering the elevator or [X] for no passengers.
   - Passenger's floor choice when entering the elevator.
2. User instructions are not case sensitive.
3. A warning message should be prompted for invalid instructions.
4. Validation is required to avoid any possible runtime error.
5. Handle instances when exceeding the elevator's capacity.
6. Provide appropriate error messages and allow users to correct their input.

Functionality #5: User Interface.

Create a text-based interface that displays the current state of the elevator system and allows users to interact with it. Refer to screenshot 1, in page 13, for a sample user interface.

Here are some suggestions for the user interface:

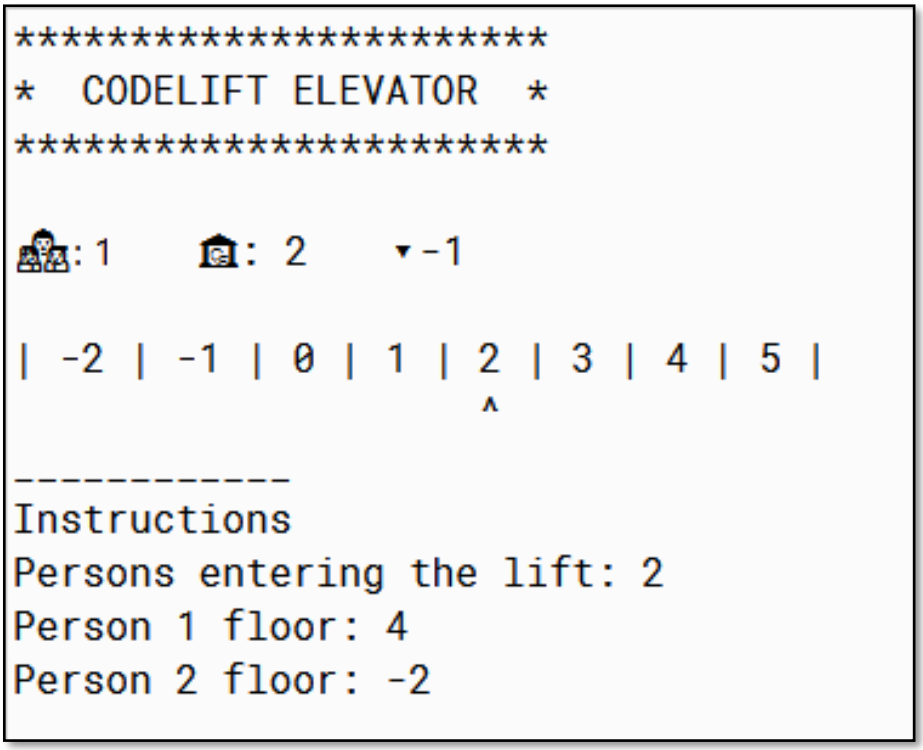
1. **Display the Current State**: Show the number of passengers inside, current floor of the elevator, and the next floor level movement. For example:



2. **Elevator Movement Visualization:** Create a visual representation of the elevator moving between floors. In the sample visualisation shown below the lift is represented using the '^' sign.

```
| -2 | -1 | 0 | 1 | 2 | 3 | 4 | 5 |
                       ^
```

Make sure that this visualisation is updated during the elevator movement, considering 1 second of delay from one floor to another.



Screenshot 1: Sample user-interface

Name your program **task4_name.py**
(e.g. task4_ john.py)

Assessment Criteria

Task1: Nuclear Power Plant Access Area					
Program Functionality	User Friendly Interface	Code Efficiency	Proper use of In-line Text *(Comments)*	Use of proper Conventions *(Snake Case, meaningful variable names etc.)*	Name of Folder & Class/es
Constant Declaration & Initialisation	User Input	Suitable Prompts / Messages displayed	Proper Randomisation of Addition Problem	Proper Randomisation of Questions	Proper Validation of User's Answer *(A, B or C only)*
Proper Validation of PIN Code *(4 Digits Only)*	Extra Features *(not listed in the task)*	**Maximum Score: 26 + 2 for every extra feature**			
0 – Not Satisfactorily | 1- Partly Satisfactorily | 2- Entirely Satisfactorily					

Task 2: Lotto System				
Program Functionality	User Friendly Interface	Code Efficiency	Proper use of In-line Text *(Comments)*	Use of Proper Conventions *(Snake Case, meaningful variable names etc.)*
Name of Folder & Class/es	User Input	Suitable Prompts / Messages displayed	Numbers Drawn Automatically *(5 numbers)*	Validation *(non-duplicates & valid numbers)*
Arithmetic Calculations *(total numbers guessed & prize won)*	Proper Use of Data Structure *(such as Arrays)*	Extra Features *(not listed in the task)*	**Maximum Score: 24 + 2 for every extra feature**	
0 – Not Satisfactorily | 1- Partly Satisfactorily | 2- Entirely Satisfactorily				

Task 3: Point of Sale System

Program Functionality	User Friendly Interface	Code Efficiency	Proper use of In-line Text (Comments)	Use of Proper Conventions (Snake Case, meaningful variable names etc.)
Name of Folder & Class/es	User Input	Suitable Prompts / Messages displayed	Options Validation (Login Screen, Main Menu & New Transaction)	Functionality Validation (Issuing of Receipt)
Ignoring Case Sensitivity (when searching for username & item)	Proper Use of Data Structure (such as Arrays)	Searching of Records (Stock & Cashiers)	Arithmetic Calculations (Subtotal, VAT & Total)	Generating Receipt Number (8-digit Number)
Display Receipt (simulating a real receipt as much as possible)	Display list of items in stock (including formatting)	Extra Features (not listed in the task)	**Maximum Score: 34 + 2 for every extra feature**	

0 – Not Satisfactorily | 1- Partly Satisfactorily | 2- Entirely Satisfactorily

Task 4: CodeLift

Overall Program Functionality	User-Friendly Interface	Proper use of Comments	Proper Conventions (Snake Case, meaningful var names etc.)	Name of Folder & Class/es	User Input	Suitable Prompts / Messages displayed
Update of Elevator Status	Update Elevator Movement Visualisation	Animate Elevator movement between floors	Elevator movement follows passengers' floor sequence	Proper use of Data Structures	**Maximum Score: 38 + 2 for every extra feature.**	
Validations					Modular Code	Code Efficiency
Number of passengers entering elevator or [x] for none	Floor Number.	Elevator Capacity Limit	Avoid Runtime errors	Instruction Case Sensitivity		

0 – Not Satisfactorily | 1- Partly Satisfactorily | 2- Entirely Satisfactorily

Section B

Self-Reflection Exercise

The following ten questions will guide your reflection on the work that you did and the effort that you invested in preparation for Code.Sprint.

Answer all questions:

1. To what extent did you manage to complete the tasks? Rate 1-5.
   [1 being all tasks were not successfully finished and 5 being all tasks were successfully finished]
2. What did you learn from completing these tasks?
3. What did you like most about these tasks? Why?
4. What difficulties did you encounter in tackling these tasks? How did you solve them?
5. Which tasks / part of tasks did you find most challenging?
6. What level of support from your mentor did you seek in completing the tasks to the level you did?
7. What other resources have you used in completing this workbook?
8. Mention some testing procedures that you carried out for each task.
9. Suggest further improvements to your work.
10. What did you learn from completing this workbook?

Each question is awarded 10 marks, of which:

- 7 marks are given for the content, and
- 3 marks for using a reflective rather than a descriptive approach.

Final Presentation Guidelines

Prepare a presentation to briefly outline your Code.Sprint journey including:

- An introduction of:
  - yourself,
  - your coding backgrounds,
  - your past Code.Sprint experiences (if any).

- The problem-solving strategies that you employed in the different tasks.
  - If any, identify situations where you have found more than one solution to tackle part/full task and on what grounds did you choose one strategy over another.

- The difficulties you encountered, and which were:
  - solved and how did you overcome your difficulties,
  - not solved.

- The lessons you have learned from this experience.
- Your suggestions to future candidates.
- Your suggestions to Code.Sprint organisers.

Your presentation will be assessed as follows:

40%	30%	15%	10%	5%
Showing a reflective approach, not just demonstrating a descriptive journal of your experience.	Using at least three different sources of support, such as mentor's support and online sources.	Answering all above questions.	Completing your presentation in maximum of five minutes.	Showing a sense of originality in your presentation.

Your presentation:

- can take any format that you prefer such as a slide-show, word-processed document, a video, a website or any other preferred presentation method,
- should not exceed five minutes, should be presented during online a day or two before the Award Ceremony. *

** You will receive instructions via email after the Final Round of the competition.*

NOTES

Contact Us:

www.codesprintmalta.edu.mt

Code Sprint Malta

A project created by the Computing Department,
within the Directorate for STEM & Vocational Programmes (DSVP)

GOVERNMENT OF MALTA
MINISTRY FOR EDUCATION, SPORT
YOUTH, RESEARCH AND INNOVATION