SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

10.014 – CTD 1D Project

F03 Team 02

Professor: Matthieu De Mari and Liu Jun

S/N	Name	Student ID
1	Austin Isaac	1007099
2	Chew Yu Qiao, Michelle	1007222
3	Chu Mei Qi	1006943
4	Everlyn Lim Jia Qi	1007032
5	Tan Hyu Seong	1006861
6	Tan Soon Kang, William	1007177

Description

Target Audience and Scenario

This game was created to help potential Singapore University of Technology and Design (SUTD) students with their university selection by allowing them to experience the life of a Term 1 Freshmore student. We feel that this is especially important in SUTD's context as, given this school's smaller cohort size, most pre-university students do not personally know any current SUTD students or alumni. We hope that this game will be informative, while also creating a fun experience that piques their interest to come to SUTD.

Description of the Game

This game is a text-based Role-Playing Game (RPG) that calls for user inputs to produce different outcomes. The game simulates the activities that SUTD Term 1 Freshmore students might experience to help players get a better understanding of the required school-life balance. The player will go through 1 week in SUTD, and the primary objective would be to pass the final 'exam' by obtaining a 'Mastery' score of 5 and above. The player should also aim to maximize their other scores, 'Happiness' and 'Health', and would lose the game if they obtained 0 points for either of these scores.

Documentation

Libraries, Lists, and Dictionaries

This game requires us to import the underlying operating system, random library and copy module. Lists of selected actions are stated and sorted by the different iterations of the day and type of actions. Day count (day_count) is also stated here to be 1, increasing with a step of 1 until a maximum of 5.

The 'total_scores' dictionary stores the players accumulative statistics and is updated throughout the game. The game begins with {'Mastery': 0.0, 'Happiness': 5.0, 'Health': 5.0}.

The 'day_scores' dictionary tallies the players statistics per day. The game beings with {'Mastery': 0.0, 'Happiness': 0.0, 'Health': 0.0}. The dictionary returns to its original computation after each day.

Functions used

1. generate_message(period, day_count)

   This function generates a message after every period (‘Morning’, ‘Afternoon’, ‘Evening’, ‘Night’) for each day. The input arguments are period and day_count.

2. generate_choices(period)

   This function takes in period from the for loop as an argument and returns a dictionary containing the choices of activities for a particular period of the day. For example, the choices for every morning would be either to go to class, study or a random action which would increase the player’s Happiness or Health score.

3. music_or_no()

   This function allows the player to choose if they would like music or not for the given day.

4. generate_music_choices()

   This function allows the player to choose the genre of music they would like to listen to for the day. They are also able to choose the song they would like to listen to through a curated song library, much like a Spotify playlist. The function then calls up the song through the YouTube link by starting Google chrome though os.system.

5. go_to_class()

   This function is activated when the user chooses the “Go to Class” option. The function uses the random library to decide whether the student obtains a good, average, or bad professor teaching the class. Based on the outcome, the function returns a score which adjusts the players statistics accordingly.

6. study()

   This function allows the player to choose how they would like to study from a set of choices in a dictionary called study_types and return the respective scores for each choice. For example, the player chose to study alone, so they would get a ‘Mastery’ score of 2, ‘Happiness’ of 0 and ‘Health’ of –1. The score comes in a tuple.

7. choose_action(random_choices)

   This function takes in the player’s choice of action for each period of the day and returns the action and its corresponding scores in a tuple. The random_choices argument allows the code to randomise a Happiness or Health activities in addition to the fixed activities that we have implemented.

8. generate_dilemma(period, choices)

   This function uses the random library, creating a chance for the player to obtain 2 additional choices during any period of the day. The 'period' argument affects the type of choices given, while the 'choices' argument ensures that the choices given before this dilemma function is not duplicated.

9. choose_dilemma(dilemma_choices)

   This function computes and returns the score obtained based on the player's decision when the generate_dilemma(period, choices) function is activated.

10. update_scores(choice_score, dilemma_score, day_scores, total_scores)

    This function tallies the score for each period of the day after all choices are made, and updates day_scores and total_scores accordingly.

11. display_scores(day_count, day_scores, total_scores)

    This function takes in the argument of day_count, day_scores and total_scores and returns the display of the points earned for the day.

12. grade_mastery(meter_value)

    This function takes in the argument of meter_value and returns the grade of the player's Mastery based on their total Mastery score after the full 5 days.

13. grade_happiness(meter_value)

    This function takes in the argument of meter_value and returns the grade of the player's Happiness based on their total Happiness score after the full 5 days.

14. grade_health(meter_value)

    This function takes in the argument of meter_value and returns the grade of the player's Health based on their total Health score after the full 5 days.

15. score_warnings(total_scores, day_count)

    This function prints warnings when the player's score values are low. This is done when their 'Happiness' or 'Health' score is between 0 and 2 for any day, and when their 'Mastery' score is below 2 from day 3 to day 5.

16. score_cap(total_scores)

    To adjust the game's difficulty, this function was created to not allow the player scores to exceed 10 for each statistic. It caps the maximum attainable scores at 10 and if the player's Happiness or Health scores reaches 0, they are out of the game.

17. game_main()

    This is the main function that runs the game. The code is as follows:

    a. The function prints the starting instructions for the player, explaining how the game should be played.
    b. All previous functions computed are called within this function to run the game.
    c. Should the player hit a ‘Happiness’ or ‘Health’ score of 0 before 5 days, they lose the game, and the code is broken.
    d. The players final statistics are summarised after the 5 days. The player is informed on whether they have or have not passed the ‘final exam’. Comments are provided based on their final total_scores.