University of
South Australia

UniSA STEM

COMP 5076– Design Thinking in the Digital age

Assignment 2: Personal Timetable

Weighting: 40%,

Due Date: See course website

Version	Date	Notes
1.0	24 Sep 23	Initial release

Overview

In this assignment, you are tasked to design and develop a text-based Python program that helps managing a weekly personal timetable. Throughout the process of developing this software program, you should apply and practice the topics learnt through this course.

In this assignment, you will:

1. Write a Python program to a set of requirements;
2. Create a recording demonstrating your program;
3. Demonstrate your understanding of your program during Week 13.

The recordings you submit will not be assessed but will be kept for our records or in the case that there are discrepancies in grades. The demonstrations in Week 13 will be scheduled in during that week and will be assessed.

Timetable

The Python program to develop is for managing a weekly timetable. The timetable should:

- support a seven-day week,
- allow creating, deleting, and updating a scheduled event, and
- each event must have a **title**, when it **starts** and **ends**, and optionally **where** it is held.

For example:

```
Title: Meet project team
When: Friday 10:00am-10:30am
Where: Brightside Cafe
```

Below is an example of a complete timetable showing a schedule for the full week. The table below is provided just to explain the concept of a weekly timetable, while the actual output of your program may depend on your own design.

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
		Gym 7-7:30am	Gym 7-7:30am	Gym 7-7:30am	Gym 7-7:30am	Gym 7-7:30am	
9am							
10am	Badminton CW gym	DS Seminar F1-24				Meet proje..	
						Meet super..	
11am				DS Practical P1-13 11:10-12pm			
12pm					Lunch w K.. 12-1pm		
1pm			CS Lecture Online 1:10-3pm				
2pm						Book club MLK Libr.. 2:30-4pm	
3pm			CS Practical P1-13 3:10-4pm				
4pm							
		Tutoring 7-8:30pm		Tutoring 7-8:30pm			

Requirements

The Python program must adhere to the following requirements.

Functional Requirements (what the program should do):

FR1. The program should start with printing out:
- the title,
- the author's name, and
- UniSA email.

FR2. The program should provide a text-based menu and prompt so that a user can perform various actions to manage the timetable.
FR3. The user should be able to quit the program through the menu.
FR4. User inputs should be validated (e.g., wrong data type, out of range, or not among the provided options) and asked again with proper error messages if invalid.
FR5. A user should be able to **create** a scheduled event. An event must have a title, when it starts and ends, and *optionally* where it is held.
FR6. A user should be able to **update** a scheduled event. Identifying an event for updating should be based on the start time of the event.
FR7. A user should be able to **delete** a scheduled event. Identifying an event for deleting it should be based on the start time of the event.
FR8. The timetable should be able to *at least* support scheduling events during the work hours, 9am – 5pm.
FR9. Scheduling an event should support at least hour level granularity (11am, 3pm, etc.).
FR10. Events should have no overlap in their duration. When creating or updating an event, the start and end time must be checked against other events to ensure there is no overlap. If a user tries to schedule an event that overlaps with another event, an error message should be displayed and ask the user to reschedule the event.
FR11. A user should be able to print the timetable which gives an overview of the schedule for the whole week. This should include as much details as possible, but may abbreviate (e.g., long title or location) or omit certain details (e.g., end time or location) of each event as needed to make them fit into a table formatting.
FR12. A user should be able to print a list of all the events scheduled on a selected day of the week in chronological order. This should provide full details of each event (without any abbreviation or details omitted) on the selected day.
FR13. A user should be able to store (save and load) the timetable data on a file with provided file name.

Non-functional requirements (constraints on the design):

NFR1. Must be written in a single Python script **<u>without using any import statement</u>**.
NFR2. At the top of the Python script should include the following comment (parts in blue must be replaced by your own personal details) :

```
#
# File: filename.py
# Author: Steve Jobs
# Student ID: 12345678
# Email ID: jobst007
# This is my own work as defined by
#  the University's Academic Misconduct Policy.
#
```

NFR3. Text output must be no more than 80 characters wide per line and 30 lines per screen.

NFR4. Loop structures must be well constructed and *never* use `break` statement.

NFR5. Sufficient documentation of the code must be practiced by using docstrings and comments. Each function must have a docstring. Each significant block of code should have a high-level description of what it is doing. Variables must have a comment explaining its purpose when it is first introduced.

NFR6. Variables and functions must have meaningful names that reflect their purpose. The only place a single character name is allowed is where a variable is defined for iteration in a loop.

NFR7. Your code should follow PEP 8 – Style Guide for Python Code available at https://peps.python.org/pep-0008/. Especially, variable and function names must use `under_score` naming style consistently.

NFR8. Your code must be structured in a procedural programming paradigm (i.e., structuring your code as a collection of functions, no class definition is allowed). Using any built-in functions that are not covered in this course is not recommended, but if it is necessary, it must have comments explaining what it means and why it is necessary.

NFR9. No global variable must be allowed. All variables should be local to a function.

NFR10. No global level statement must be allowed other than calling the `main()` function which must be the starting point of your program.

NFR11. The program must be robust and should not crash in any case.

NFR12. The program should be user friendly so it could be easily understood how to use it even for a first-time user.

Advanced Features:

- The program could support scheduling events outside the work hours. It should include them at relevant places when printing the timetable for the whole week.
- Scheduling an event could support minute level granularity (10:10am, 2:45pm, etc.).
- The program could provide an option for the user to choose the start day of the week (e.g., the first day of the week could be either Sunday or Monday). This option should be applied when printing the timetable for the whole week or printing the search results.
- A user could be able to search for an event based on keywords in their title or location. The keyword search should be case insensitive (i.e., both ‘lect’ and ‘LeCt’ should have the same search results). The search result should print out a list of all the events that matches the search condition, sorted in chronological order and with their full details.
- The event which user wants to delete or update could be identified using a keyword search, in addition to its start time (as it was prescribed as a functional requirement).

Video Presentation

A video recording (no longer than 10 minutes) of yourself demonstrating your program must be included in your submission.

You must first state your name then give verbal explanation while demonstrating your program showing how each feature works. The video must show your own face together with the computer screen (you can do this by recording on Zoom while sharing your computer screen.) The video must clearly show the output on the screen so the text information would be legible.

The video must also include explanation of the overall structure of your Python program by providing a brief description of each function. It must also include a description of the file structure used, explaining how the timetable information is stored in a file.

Live Demonstrations

You will demonstrate your understanding of your code in Week 13. This is to assess how well you understand the code you have submitted. Based on your demonstration, your marks may be moderated.

These demonstrations will be scheduled in Week 13 and may take place outside of your practical class.

Submission

You must submit your assignment through the LearnOnline course webpage by the due date as indicated on the submission link. Submission of your assignment must include:

1) a Python program file (.py file)
2) a presentation video clip (maximum 10 minutes in .mp4 file)

Late Submission

Late submissions will be penalised by scaling the marks by 50% unless with pre-approved extension. Application for extension should be lodged through the course website and it requires a document proving acceptable reasons for extension (e.g., medical certificate, or a letter from your work supervisor).

Academic Misconduct

Students must be aware of the academic misconduct guidelines available from the University of South Australia website. Deliberate academic misconduct such as plagiarism is subject to penalties. Information about Academic integrity can be found in Section 9 of the Assessment policies and procedures manual at:

https://i.unisa.edu.au/policies-and-procedures/codes/assessment-policies/

Marking Criteria

Criteria	**Max**	**Mark**	**Comment**
Python Program			
Non-functional requirements	10		
Functional requirements	15		
Advanced features	10		
Sub Total	35		
In class demonstration	5		
Total	40		
Penalties			
Missing student details in submissions.	-10		
Failure to explain program and make changes during live demonstration.	Up to -20		

* Late submissions will be penalised by scaling down to 50% of the total mark.