UNIVERSITY OF SOUTHERN QUEENSLAND
CSC5020 – Foundations of Programming (Semester 1, 2021)
Assignment 1 Specification

Days-To-Go

Due Date: 25 March 2021

Weight: 10%

Goals and Topics

The assignment problem is straightforward. All necessary details have been supplied. The solution of the problem will be straight line code which will use the programming concepts and strategies covered in Workshops 1-4. The subgoals are:

- Understanding values, variables, operations, and functions
- Translating simple design into Python code
- The mechanics of editing, interpreting, building and running your program
- Testing your program
- Commenting your source code
- Becoming confident and comfortable with programming in small problems

Your Task

"Days-To-Go" is an interesting and useful feature in website design. It shows up the remaining number of days (or even with hours, minutes, and seconds) towards an event. For example, at anytime you visit the official website of Tokyo 2020 Olympic Game (https://tokyo2020.org/en/), you will see such a feature showing the number of days, hours, minutes and seconds towards the starting date of Tokyo 2020. In this assignment, you are going to write a Python program to implement the "Days-To-Go" feature.

The Concept of Time in JavaScript

In Python, a time is defined as a **Date** Object. Each date object stores its state as a time value, which is a primitive number that encodes a date as seconds since 1 January 1970 00:00:00 UTC. Thus, a date later than 1 January 1970 00:00:00 UTC will have a positive time value, whereas an earlier date will have a negative time value. On the basis of the common timeline (which we all live on), the distance between any two dates can be calculated using their time values in seconds. Figure 1 illustrates the concept, where *C* is a date earlier than 1 January 1970 00:00:00 UTC, and *A* and *B* are later with *B* being further than *A*.

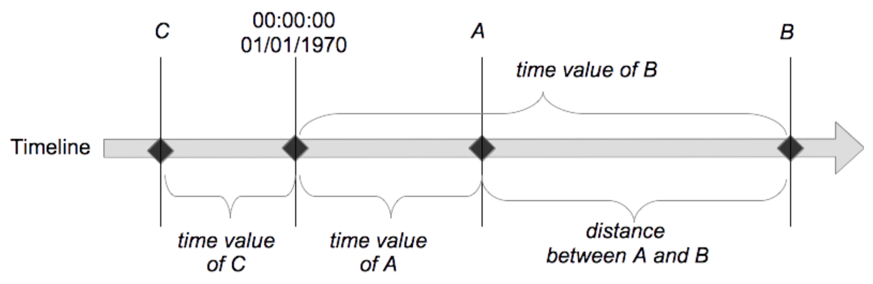

Figure 1: The Difference Between Two **Date** Instances

Functional Requirements

Variables

Within the script section, create variables following professional conventions and initialise them using right values. Some variables have been suggested in the following tables. You should create more when necessary.

Table 1: Variables1

Description	Value
Number of seconds in a day	60*60*24
Number of seconds in an hour	60*60
Number of seconds in a minute	60

Table 2: Variables2

Description	Initialising value description	Type
Event	The name of event	*String*
Year of the event	The year of event	*Number*
Month of the event	The month of event	*Number*
Day of the event	The day of event	*Number*

Calculation

1. Create a **Date** object for the event by using the variables created previously.
    - Use one input function to input the name of event with the hint "Enter the name of event".
    - Use another three input functions to input the year, month, day of the event one by one with the hints "Enter the year of the event", "Enter the month of the event" and "Enter the day of the event". Then transfer them to the Date constructor.

Example of transferring number to the Date constructor and calculate how many seconds of the date since 1 January 1970 00:00:00 UTC:

```
>>> import datetime, time
>>> t = datetime.datetime(2011, 10, 21, 0, 0)
>>> time.mktime(t.timetuple())
1319148000.0
```

- Mind the order of arguments sent to the constructor;

2. Obtain the current time instant, a floating-point number of seconds since "the epoch" (see Exercise 4.1).
3. Calculate the difference between the current time and the event time **(assume it is the starting time 0:00:00 a.m. on the date of the event):**
    - Get the event's time value in seconds as the example above.
    - Deduct the time value of current time by using the value of event time.
4. Calculate the number of days to the event:
    - Divide the time value difference by the number of seconds in a day;
    - Use the floor division to reduce the result number to an integer.
5. Calculate the number of hours, minutes, and seconds in the remaining time value:
    - Mod the time value difference by the number of seconds in a day;
    - Divide the mod result by the number of seconds in an hour;
    - Use the floor division to reduce the number to an integer for the number of hours;
    - Repeat three steps above to calculate the number of minutes and seconds. You may need to update the calculating formula accordingly.

Presentation

A sample output is presented as below when running the "Days-To-Day" program.

"Christmas is on 25/12/2021 (XXXX days, XXXX hours, XXXX minutes and XXXX seconds later)."

Note that

- the information should be displayed using the **print** function;
- wherever possible you should use variables in expressions instead of explicit values (e.g., literals and numbers), for example, using the variable created for the name of event instead of a string value of "Christmas"; "XXXX" are the results from your calculations.

Testing

Test your program by entering the date of Christmas in 2021 or Tokyo 2020 Olympic Game to see whether the outcome is correct or not.

Non-functional Requirements

Structure of the Source Code

- All code should appear in an *.ipynb* file.

Comments

- You are required to add at least three comments to the source code.
- Do not comment on every single line, instead, comment on blocks of code with a common purpose.
- Do not simply translate the syntax into English for comments, instead, describe the purpose of blocks of code.

Submission

What You Need to Submit – Three Files

For a complete submission, you need to submit three files as specified below.
The assignment submission system will accept only the files with extensions specified in this section.

1. *Statement of Completeness* in a file saved in *.doc, .docx* or *.odt* format in 200-300 of your own words describes:

   - **The state of your assignment**, such as, any known functionality that has not been implemented, etc. (It is expected that most people will implement all the functionality of this assignment).
   - **Problems encountered**, such as, any problems that you encountered during the assignment work and how you dealt with them.
   - **Reflection**, such as, any lessons learnt in doing the assignment and suggestions to future programming work.

2. *The program* in a file saved with an *.ipynb* extension contains the source code implemented following the functional and non-functional requirements.

3. *The program* in a file saved with a *.doc/.docx/.odt* extension contains the exact same source code in the *.ipynb* file. You can just download the *.ipynb* as the *.py* file, then copy the code in the *.py* file and paste it to the *.doc/.docx/.odt* file. If you don't submit your code in **both** *.doc/.docx/.odt* and *.ipynb* files, or *the code* in *.doc/.docx/.odt* and *.ipynb* files are **different**. You will get a penalty (up to 5 marks).

Marking Criteria

The assignment will be marked out of 10. Table 3 presents the marking criteria. If all criteria are satisfied you will receive 10 marks. If not all criteria are met, part marks may be given. Check your own submission against these criteria before you submit.

Table 3: Marking Criteria

ID	**REQUIREMENTS**	MARK
	Statement of Completeness	
1	The statement is in appropriate length of 200-300 of student's own words	0.5
2	The "State of assignment" reflects the true state of completeness	0.5
3	The "Problems encountered" discusses problems and dealing strategies	0.5
4	The "reflection" discusses learnt lessons and reasonable suggestions	0.5
	Subtotal	*2*
	Functional Requirements	
5	The program is running without any syntax errors	0.5
6	All variables are used appropriately, and identifiers of variables are following professional conventions	0.5
7	The input function for entering the event name is correct	0.5
8	The input function for entering the year is correct	0.5
9	The input function for entering the month is correct	0.5
10	The input function for entering the day is correct	0.5
11	Transferring inputs to the Date constructor correctly	0.5
12	The event time and current time instant, a floating-point number of seconds since "the epoch" are calculated correctly	0.5
13	The difference between two times (the date of the event and now) is calculated using an appropriate strategy	0.5
14	The number of days to the date of the event is calculated correctly	0.5
15	The number of hours to the date of the event is calculated correctly	0.5

16	The number of minutes to the date of the event is calculated correctly	0.5
17	The number of seconds to the date of the event is calculated correctly	0.5
18	The calculating result is displaying appropriately	0.5
	Subtotal	*7*
	Non-functional Requirements	
19	Variables are used in calculation and expression instead of explicit values	0.5
20	At least three comments are added to describe the purpose of blocks of code	0.5
	Subtotal	*1*
	TOTAL	**10**

Suggested Strategy

You have three weeks to finish the assignment. Plan to complete it on time. Do not write all of the code in one sitting and expect that everything will be working smoothly like a magic.

First week Read assignment specification, clarify anything unclear by putting a post on the Assignment 1 Forum, think about how to do it, how to test it, devise high-level algorithms for each independent part of the assignment. Begin to type program (with comments), in incremental stages. Seek help on assignment forum if needed.

Second week Re-read the specification, continue to refine design of various sections to code, bring up any problems to the assignment forum if necessary. Finish initial coding.

Third week Fully test the program; have another review on the source code; re-read the specification (especially marking criteria) to make sure you have done exactly what is required. Document the "Statement of Completeness".