ECM1421 System Development 1
Programming Exercise 1

Hand-out date: 16th January 2023
Hand-in date: 24th February 2023
Feedback: 17th March 2023

This assignment is worth 30% of the marks for ECM1421. This is an individual assessment. Your attention is drawn to the university's regulations on plagiarism.

Submission method: Anonymous submission via ELE.

Submission format: Your submission will be of **one zip** file.
Submissions should be anonymous. You should title each submission with your Student ID number (i.e. something like '7200nnnnnn'), then the assignment title, but do not include your name in any part of the document.

Submission format details:

Submit a **zip file** containing:

PDF file	to include your algorithms/flowchart from 1a(ii),	1a (ii),
Python code files	woodenbox1.py	1a (i)
	woodenbox2.py	1b
	ticketit_prep.py	2a
	ratings_analysis.py	2b (i)
	monthly_analysis.py	2c
Data files	ticketit_prep.dat	2a
	seg_ratings.csv	2b (i)
	month_genre.csv	2c
Chart image	seg_ratings.jpg	2b (ii)

Do not include any other material in your submission.

The assignment consists of 2 tasks as detailed below.

Assume that data files will be in the same location as program files throughout this coursework.

In this coursework, **do not import any Python modules**, except 'sys' if required.

Important:
Please check your submitted work against the requirements listed above. Marks cannot be given for work that is not submitted or is formatted so as to be illegible.

Task 1 Wooden Tray Manufacture
Current Scenario

A business purchases lengths of timber of dimension 1.8 metres x 20mm x 72 mm.

Their requirement is to construct an open top seed tray or box from each piece of timber, with no waste (ignore waste from the saw cut).

The base of the tray will consist of 2 lengths of timber. The base will be overlapped by the sides and ends (4 pieces of timber). The sides will overlap the ends.

Completed Box – in use

View of base from below.

Note design of base

Total length of timber is 1800 mm.

The 2 ends are 2 x 144 mm = 288mm .

This leaves 1800 – 288 = 1512mm for the remaining 4 pieces.

If the base pieces are x mm then,

$2x + 2(x + 40) = 1512$

$4x + 80 = 1512$

$4x = 1432$

$X = 358$ mm

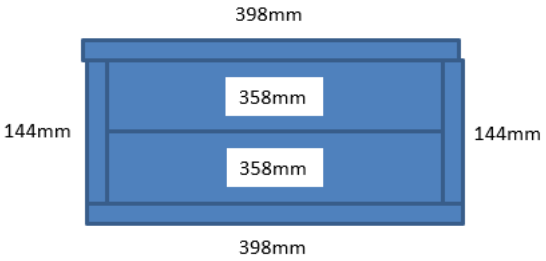
Our cutting list is therefore:

Base : 2 x 358 mm = 716 mm
Sides: 2 x 398 mm = 796 mm
Ends: 2 x 144 mm = 288 mm
Total: = 1800 mm

Here are the calculations made by the business to determine how to cut up one 1.8 metre length of timber – the 'cutting list.'

(We ignore any waste from the saw cut.)

Here is the base of the resulting box.



New Business Requirement

The business is now going to receive timber in a variety of lengths, ranging from the current 1.8 metres to 6 metres (inclusive) in increments of 0.3 metres, with width and thickness unchanged. It is going to continue to make boxes to the same design but with a varying length base.

A Python program is required to produce a cutting list, given the length of piece of timber received, which can now vary.

Assignment Submission Requirement

Task 1 a.

(i) You are required to write the Python program, '**woodenbox1.py**' that will accept user input of the length of timber in metres to 1 decimal place. Using a valid user input, the program will output a cutting list for base, sides and ends in similar format to that detailed on the previous page. Program details are given below. **(30 marks)**

(ii) Before you write your program, you should produce the algorithm to be used by the program in a flowchart format. (Calculation details not required).(**5 marks)**

Program requirement details '**woodenbox1.py'**:

On initialisation the program will prompt the user for an input as follows:

```
Enter length of timber (metres – to 1 dp) :
```

If the user wishes to quit the program at this point, they should enter '`quit`'. The program will then exit without any output.

If the user does not enter a numeric value, then the error message '`A numeric value is required`' should be shown and the prompt displayed again

The user should enter a valid length of timber in metres to 1 dp. If the user input does not have 1 decimal place, then the error message '`Enter length in metres to 1 dp`' should be shown and the prompt displayed again.

If an otherwise invalid numeric value is entered, then a suitable error message should be output, and the prompt displayed again. Suitable error messages are:

```
Length is too short.
Length is too long.
Invalid value for length.
```

The entry of a valid input will result in a display, showing the cutting list, with the format below:

```
Cutting List for 1.8 metre length:
Bases : 2 x 358 mm =  716 mm
Sides: 2 x 398 mm  =  796 mm
Ends: 2 x 144 mm  =  288 mm
Total:            = 1800 mm
```

Lengths of components should be rounded down to nearest millimetre, so that total length doesn't exceed length input.

The program will then exit without any further message.

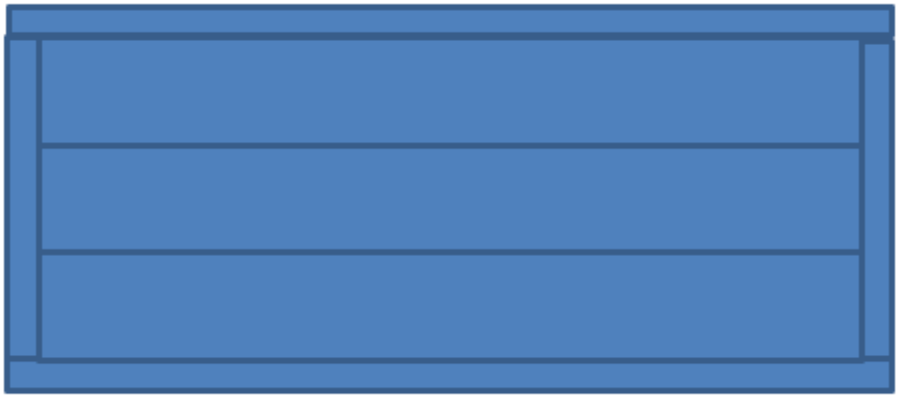
Note: Do not import any Python modules, except 'sys' if required.

Modified Requirements.

Using your program **woodenbox1.py** and entering a length of 6.0 metres you will find that the resulting length of box is well over a metre in length. In practice this length is found to result in a weak box.

A long, narrow box is weak

As a result, we must implement a design requirement that the length of **each** of the base slats should not be more than 6 times their width (i.e., a long tray would require more than 2 base slats).



Modified design with more than 2 base pieces

1. b. Take a copy of your Python code **woodenbox1.py** and name it '**woodenbox2.py**' Modify **woodenbox2.py** in the light of this new design requirement. (**5 marks)**

Task 2 -Ticket Booking Service

Background

TicketIt is a software application that provides a ticket booking service for the entertainments industry. TicketIt has implemented a cinema seat booking system for client cinemas using a SQL Server relational database. As at 31.12.2022 this system has been running for just over a year and has processed over 1 million bookings from 73000 registered customers. As an extension to the services that it provides to its clients, TicketIt intends to provide data analysis support to enable them to identify market trends revealed by TicketIt bookings data.

Sample data has been extracted from the SQL Server database by simple download of database tables into flat files, which are provided to you. The table below gives details of the structure of your data files. All files, unless detailed below, have an 'id' column that is a unique primary key.

Because the data has been extracted from a normalised relational database with foreign key constraints, we can assume that the relevant relational links between these flat files are in place.

Data

File	**Details**
booking	Id, customer id, date, total price (£), film showing id
customer	Id, first name, last name, socio-economic group, date of birth, contact id, address id.
films	Note that a tab-delimited text file is provided here. Columns are id, language, title, plot, release date and rating (/10). Note that the 'plot' column contains much data.
film film genres	A 'cross-reference' between film and film genres, as a film can belong to more than one genre. Columns are film id and genre id.
film_genre	Simple 'id' and 'name' structure.
film_showing	Id, venue (cinema) id, location, film id, date, seats available, fully booked?, seat price (£)

Some files are too big to be handled correctly by Excel, so do not attempt to re-save them in Excel. Other simple applications such as MS Notepad or one of the many coding text editors will probably enable you to view the complete files, should this prove useful.

Note that cleaning of this test data has not been completely successful in the case of the Film entity. We will ignore this during this assignment but note that Film Titles are not unique.

Task 2a Data Preparation

Data normalisation, whilst enabling us to maintain data integrity within a relational database environment, is not providing a good base for data analysis. Your first task is to de-normalise the data.

Write a Python program '**ticketit_prep.py'** that reads the data files and creates **one flat file, 'ticketit_prep.dat' with following columns:**

id	cust_id	bkg_date	cust_seg	film_id	film_rating	genre
(from bookings)			(from customer)	(from film)		(from film film genre)

Note that this will generate more than one line of data for many bookings, as a single film can relate to many genres.

Data Cleaning

Some data cleaning my be required as we prepare our data. See the 3 notes below.

1. **Empty columns?** We need to consider our assumptions concerning missing or default data. We are assuming that none of our relevant data is NULL. Customer SEG is always one of the 4 valid values (AB, C1, C2, DE).
2. **Default Values**. A quick examination of Films data shows that over 2,000 films appear to have a zero rating. We will assume that this is because they have not yet been rated. Therefore, we will default the value for the rating of these films to 5.5 in our analysis.
3. **No genre**. There appear to be over 2000 films that have no identified genre. This may be the only circumstance that produces a NULL value in your output data. All other columns of your '**ticketit_prep.dat**' file should be non-null. You should ensure that your code deals appropriately with such a film.

Your output file '**ticketit_prep.dat'** will have a header row with the column headings shown above. Data will be output as comma separated values. Records will be ordered by booking_id.
(23 marks)

Task 2b Ratings Analysis – Slice by Ratings

Are the film-going habits of the various socio-economic groups affected differently by film ratings?

The aim of this task is to suggest an answer to this visually.

i. Summarise the data.

Using your '**ticketit_prep.dat'** as a data source, write a Python program '**ratings_analysis.py'** to summarise the data for bookings and film ratings. Read your data source and accumulate the number of bookings for each film rating (0 - 10) for each SEG (AB, C1, C2, DE). Use your accumulated data to write a csv output file as follows:

- The name of the file is **seg_ratings.csv**.
- The file is in csv format (with comma separators).
- The file has 5 lines and 11 columns. The top line is a header line with format:
  - **seg,0,1,2,3,4,5,6,7,8,9,10**
- The first column consists of the labels **seg, AB, C1, C2 and DE**
- The 4 lines that are not header lines hold the relevant seg/ratings data.

When viewed in Excel file '**seg_ratings.csv**' will look like this (plus your data)

seg	0	1	2	3	4	5	6	7	8	9	10
AB											
C1											
C2											
DE											

Rounding convention for grouping ratings data:
Your program should group your ratings data by accumulating to the nearest integer and rounding-up at the boundary point so, for example

Data point	Accumulates values of…
0	0- 0.49
1	0.5 – 1.49
…	
10	9.5 - 10

(16 marks)

ii. Ratings Visualisation: Creating an Excel chart

Open your '**seg_ratings.csv'** file and save in Excel workbook format (.xlsx).

You are required to communicate your analysis results visually by creating an Excel chart.
(**Hint**: Select the data area and use Excel command <Insert> <Chart>, selecting chart type <Line><Stacked Line>.)

Format the chart to include the following features:

When complete, save the chart (only) as '**seg_ratings'** in jpeg format for submission.
(8 marks)

Excel chart guidance
https://support.microsoft.com/en-us/office/create-a-chart-from-start-to-finish-0baf399e-dd61-4e18-8a73-b3fd5d5680c2
https://www.ablebits.com/office-addins-blog/2015/10/29/excel-charts-title-axis-legend/

Task 2c Ratings Analysis – Dice by Month and Genre

How do film going habits change through the year?

Write a Python program '**monthly_analysis.py**' to read your '**ticket_prep.dat**' file and analyse monthly film bookings by genre. Process only those records with a booking date in 2022.

Write results to a comma-separated values file, '**month_genre.csv'**.

There are 12 months in the year, which will be the columns in the output file, with appropriate headings. There are 20 film genres. Your output file should have the layout below:

month	jan	feb	mar	apr	may	jun	jul	aug	sep	oct	nov	dec
1												
2												
3												
4												
5												
6												
7												
8												
9												
10												
11												
12												
13												
14												
15												
16												
17												
18												
19												
20												

(13 marks)

Useful technical references:

Task 1:

User Input:

https://www.w3schools.com/python/python_user_input.asp

Functions:

https://www.w3schools.com/python/python_functions.asp

While loops:

https://www.w3schools.com/python/python_while_loops.asp

If…Else

https://www.w3schools.com/python/python_conditions.asp

Decimal and floating-point arithmetic

https://docs.python.org/3/library/decimal.html

https://www.tutorialspoint.com/decimal-fixed-point-and-floating-point-arithmetic-in-python

Try…Except

https://www.w3schools.com/python/python_try_except.asp

Algorithms:

https://www.tutorialspoint.com/programming_methodologies/programming_methodologies_writing_the_algorithm.htm

Task 2:

https://realpython.com/read-write-files-python/#reading-and-writing-opened-files

https://www.w3schools.com/python/python_file_handling.asp