CP1401/CP5639 2025 TR2 Assignment 1

Task:

For this assessment, you are to solve 3 separate problems by writing algorithms in pseudocode and programs in Python 3. This assignment is designed to help you build skills using:

1. **Input, Processing and Output** – Band Calculator
2. **Decision Structures** – Device Time Determinator
3. **Repetition Structures** – Sales Tracker

Do not define any of your own functions or use any code constructs (e.g., lists, files, dictionaries, exceptions) that have not been taught in the subject up to this point. **100%** of what you need to know to complete this assignment successfully is taught in the lectures and practicals. Carefully check the rubric below to understand how you will be assessed. There should be no surprises here.

Each program should be written in a separate Python file with the prescribed file name.
Each file should follow the structure provided in the example below. That is, each file should start with a module docstring comment at the very top containing your own details and your pseudocode, then your solution code should follow. Replace the parts in <brackets>, which are there to show you where to put your own content.

Example for program 1:

```
"""
CP1401 2025 TR2 Assignment 1
Program 1 - Band Calculator
Name: <your name>
Date started: <date>

Pseudocode:

<pseudocode here>
"""

<code here, e.g., >
print("...")
```

Requirements and Expectations:

1. We encourage you to work incrementally on these tasks: focus on completing small parts at a time rather than trying to get everything done at once.
2. Sample output from the programs is provided with each program description.
   Ensure that your programs match these, including wording, formatting, etc. Think of this as helpful guidance as well as training you to pay attention to detail. The sample output is intended to show a full range of situations so you know how the programs should work.
3. You do **not** need to handle incorrect types in user input. E.g., if the user is asked for a number and enters "none" instead of an integer, your program should just crash. That's fine.
4. Make use of named constants as appropriate, e.g., for appropriate things that would otherwise be "magic numbers". See our guide for guidelines about choosing constants.
5. You are expected to include appropriate comments in each of your programs (not just the module docstring). Use # block comments for things that might reasonably need a comment. Use comments as you have been taught and is summarised in our guide.
   Do not include unnecessary "noise" comments.
6. Ensure that your naming follows the style guide.

Program 1 – Band Calculator

Learning outcome focus: Input, Processing, Output
File name: a1_1_band.py

Rando Bando is an agency for musicians, which provides a service to clients who want to book random bands. Clients select the size of the band and how many minutes they want the band to play for. This software then calculates the price for that performance.
Every performance has a call-out fee of $72, then each musician costs $1.95 per minute.
You should write your program so that if these base rates were to change, the programmer would only need to change them in one place (that's what constants are for).

The sample output below shows the currency values displayed with two decimal places, which your program should also do.

Sample Output from 3 different runs:

It should be clear what parts of the samples are user input for all samples in this document.
E.g., in the first example below, the user entered 3 and 115. All the other parts of the sample were printed by the program.

```
Rando Bando!
Let's make good music. Get your quote here...
Number of musicians: 3
Number of minutes  : 115
3 musicians for 115 minutes will cost $744.75
```

```
Rando Bando!
Let's make good music. Get your quote here...
Number of musicians: 45
Number of minutes  : 312
45 musicians for 312 minutes will cost $27450.00
```

This last run shows that there is no error checking.

```
Rando Bando!
Let's make good music. Get your quote here...
Number of musicians: -4
Number of minutes  : 0
-4 musicians for 0 minutes will cost $72.00
```

Program 2 – Device Time Determinator

Learning outcome focus: Decision Structures
File name: a1_2_device_time.py

A youngster's device time is allocated based on their number of music practices and whether they mow the lawn.
If they mow, then device time is allocated at 18 minutes per practice.
There's a special cupcake bonus if the youngster does at least 6 practices.
If they do not mow, then they do not get device time or a cupcake :(
The user should be able to enter "yes" in any capitalisation (see examples below) to indicate that they did mow the lawn.

In comments, after your pseudocode for this question, write a brief explanation of which decision pattern(s) you chose to use and why.

Note: there is no looping or error-checking in this program.

Sample Output from 3 different runs:

```
Device Time Determinator
Number of practices: 8
Did you mow? not really
No device time :(

Device Time Determinator
Number of practices: 3
Did you mow? yes
You get 54 minutes of device time :)

Device Time Determinator
Number of practices: 7
Did you mow? YES
You get 126 minutes of device time :)
And you get a cupcake!
```

Program 3 – Sales Tracker

Learning outcome focus: Repetition Structures
File name: a1_3_sales.py

This program helps a salesperson keep track of sales made per day.
The user will be asked for the number of days to track, then the sales for each of those days.
When this is entered, the program will tell them the total and average sales as well as a message based on comparing their last day's sales to the average. See the sample output below for the exact messages.
Both number of days and each sales amount must be error checked.
There is no maximum number of days, but the minimum is 1.
There is no maximum sales amount, but the minimum is 0.

Suggestion: One good way to approach this question is to design it and get it working without the error-checking, then add the error-checking, one section one at a time.

In comments, after your pseudocode for this question, write a brief explanation of which repetition pattern(s) you chose to use and why.

Sample Output from 3 different runs (the first shows all the error-checking)**:**

```
Sales Tracker
Number of days: -1
Error. Number of days must be >= 1
Number of days: 0
Error. Number of days must be >= 1
Number of days: 3
Day 1 sales: -2
Error. Sales must be >= 0
Day 1 sales: -3.2
Error. Sales must be >= 0
Day 1 sales: 0
Day 2 sales: 5.6
Day 3 sales: 12.82
Total sales = $18.42 for 3 days. Average sales = $6.14 per day.
Great momentum!

Sales Tracker
Number of days: 2
Day 1 sales: 902.54
Day 2 sales: 801
Total sales = $1703.54 for 2 days. Average sales = $851.77 per day.
Pick up the pace.

Sales Tracker
Number of days: 3
Day 1 sales: 9
Day 2 sales: 9
Day 3 sales: 9
Total sales = $27.00 for 3 days. Average sales = $9.00 per day.
Great momentum!
```

Submission:

Submit 3 separate Python files, named as in the instructions. **DO NOT ZIP/COMPRESS YOUR FILES**. Upload your 3 separate .py files on LearnJCU (under Assessments as instructed).
Submit your assignment by the date and time specified on LearnJCU. Submissions received after this date will incur late penalties as described in the subject outline.

Integrity:

The work you submit for this assignment must be your own. Submissions that are detected to be too similar to that of another student or other work (e.g., online code or generated by AI) will be dealt with according to the College procedures for handling plagiarism and may result in serious penalties.

The goals of this assignment include helping you gain understanding of fundamental programming concepts and skills, and future subjects will build on this learning. Therefore, it is important that you develop these skills to a high level by completing the work and gaining the understanding yourself. You may discuss the assignment with other students and get general assistance from your peers, but you may not do any part of anyone else's work for them, and you may not get anyone else to do any part of your work. **Note that this means you should never give a copy of your work to anyone or accept a copy of anyone else's work, including looking at another student's work or having a classmate look at your work.**
If you require assistance with the assignment, please ask **general** questions as your lecturer has instructed, or get **specific** assistance with your own work by talking with your lecturer or tutor.

The subject materials (lectures, practicals and other guides provided in the subject) contain all the information you need for this assignment. You should not use online resources (e.g., Google, Stack Overflow, GenAI like ChatGPT, Copilots, etc.) to find resources or assistance as this would limit your learning and would mean that you would not achieve the goals of the assignment - mastering fundamental programming concepts and skills.

Marking Scheme:

Ensure that you follow the processes and guidelines taught in the subject to produce high quality work. Do not just focus on getting your code working. This assessment rubric will be applied as an average across all 3 questions for this assignment. It provides you with the characteristics of exemplary to very limited work in relation to task criteria, covering the outcomes:

- SLO1 - apply problem-solving techniques to develop algorithms in the IT context
- SLO2 - apply basic programming concepts to develop solutions

Criteria	Exemplary (9, 10)	Good (7, 8)	Satisfactory (5, 6)	Limited (1-4)	Very Limited (0)
Algorithm SLO1 20%	Clear, well-formatted, consistent and accurate pseudocode that completely and correctly solves the problem.	Exhibits aspects of exemplary (left) and satisfactory (right)	Some but not many problems with algorithm (e.g., incomplete solution, inconsistent use of terms, inaccurate formatting).	Exhibits aspects of satisfactory (left) and very limited (right)	Many problems or algorithm not done.
Correctness SLO2 20%	Program works correctly for all functionality required.		Program mostly works correctly for most functionality, but there is/are some required aspects missing or that have problems.		Program works incorrectly for all functionality required.
Similarity to sample output SLO2 10%	All outputs match sample output perfectly, or only one minor difference, e.g., wording, spacing.		Multiple differences (e.g., typos, spacing, formatting) in program output compared to sample output.		No reasonable attempt made to match sample output. Very many differences.
Identifier naming SLO2 15%	All variable and constant names are appropriate, meaningful, consistent, following style guide.		Multiple variable or constant names are not appropriate, meaningful or consistent, not following style guide.		Many variable or constant names are not appropriate, meaningful or consistent, not following style guide.
Use of code constructs SLO1, SLO2 20%	Appropriate code constructs, correct pattern for the problem (best tool for the job), as taught in the subject.		Mostly appropriate code use but with definite problems, e.g., unnecessary code, poor choice of decision or repetition patterns.		Many significant problems with code use. **Any use of while True.**
Commenting SLO2 10%	Helpful block/inline comments and module docstring contains all program details, no 'noise' comments.		Comments contain noise, unhelpful comments, or missing details in module docstring, or missing block/inline comments.		Commenting is very poor either through having too many comments (noise) or too few comments.
Formatting SLO2 5%	All formatting meets PEP8 standard, including indentation, horizontal spacing and consistent vertical line spacing. PyCharm shows no formatting warnings.		Multiple problems with formatting reduce readability of code. PyCharm shows formatting warnings.		Readability is poor due to formatting problems. PyCharm shows many formatting warnings.