CP1401/CP5639 2023 TR3 Assignment 1

Task:

For this assessment, you are to plan (using pseudocode) and then implement 3 separate programs in Python 3. This assignment is designed to help you build skills using:

1. Pay Calculator: **Input, Processing and Output**
2. Device Time Determinator: **Decision structures**
3. Sleep Tracker: **Repetition structures**

Do not define any of your own functions or use any code constructs (e.g., lists, files) 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.

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 details and work.

Example for program 1:

```
"""
CP1401 2023 TR3 Assignment 1
Program 1 - Pay Calculator
Student 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 spacing, spelling, 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 the number of minutes 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 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 on their own line 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. Check the rubric below carefully to understand how you will be assessed. There should be no surprises here – this is about following the best practices we have taught in the subject.

Program 1 – Pay Calculator

Learning outcome focus: Input, Processing, Output
File name: a1_1_pay_calculator.py

At Experience Counts, workers get paid based on their level of experience.
Write a program that asks an employee for their number of hours worked and experience level, then displays how much their total pay will be.
There is no limit to an employee's experience level and no error-checking is required.
The base pay for experience level 0 is $45.00
Each level of experience gives 5% more pay, so experience level 1 is $47.25; experience level 3 is 15% more than the base, or $51.75, etc. You should write your program so that if the base pay were to change, the programmer would only need to change it 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 10 and 3. All of the other parts of the sample were printed by the program.

```
Experience Counts Pay Calculator
Number of hours worked: 10
      Experience level: 3
Based on your experience level (3):
Your hourly pay rate is $51.75
Your total pay is $517.50

Experience Counts Pay Calculator
Number of hours worked: 40
      Experience level: 99
Based on your experience level (99):
Your hourly pay rate is $267.75
Your total pay is $10710.00

Experience Counts Pay Calculator
Number of hours worked: 0
      Experience level: 52
Based on your experience level (52):
Your hourly pay rate is $162.00
Your total pay is $0.00
```

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 15 minutes per practice.
There's a special cupcake bonus if the youngster does at least 7 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.

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? no
No device time :(

Device Time Determinator
Number of practices: 3
Did you mow? yes
You get 45 minutes of device time :)

Device Time Determinator
Number of practices: 7
Did you mow? YES
You get 105 minutes of device time :)
And you get a cupcake!
```

Program 3 – Sleep Tracker

Learning outcome focus: Repetition Structures
File name: a1_3_sleep_tracker.py

A “sleep debt” represents the difference between a person’s desirable amount of sleep and how long they actually sleep for. Write a program that prompts the user to enter how many hours they slept each day over a work-week period of 5 days, then informs them of their sleep debt status.
Using 8 hours per day as the desirable amount of sleep, determine their sleep debt by calculating the total actual hours of sleep and subtracting that from the total desirable hours of sleep.
Display results messages as demonstrated below.
Note that in this program, you must use an error-checking loop to ensure that the user's inputs are within the range 0-24 inclusive.
As with the other programs, you should think about using CONSTANTS to make it easy (in one place) to change the program, such as calculating sleep debt for a period of 7 days instead of 5.

With (next to) your pseudocode for this question, include a brief justification/explanation of which repetition pattern(s) you chose to use and why.

Sample Output from 2 different runs:

```
Sleep Tracker
Night 1 hours sleep: 7.5
Night 2 hours sleep: -3
Invalid number of hours.
Night 2 hours sleep: 25
Invalid number of hours.
Night 2 hours sleep: 0
Night 3 hours sleep: 8.75
Night 4 hours sleep: 6
Night 5 hours sleep: 7

Recommended total sleep is: 40
Your total hours of sleep : 29.25
Your sleep debt over this time is: 10.75
```

```
Sleep Tracker
Night 1 hours sleep: 8
Night 2 hours sleep: 8
Night 3 hours sleep: 8
Night 4 hours sleep: 8
Night 5 hours sleep: 8

Recommended total sleep is: 40
Your total hours of sleep : 40.0
You are getting enough sleep. Keep it up!
```

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.

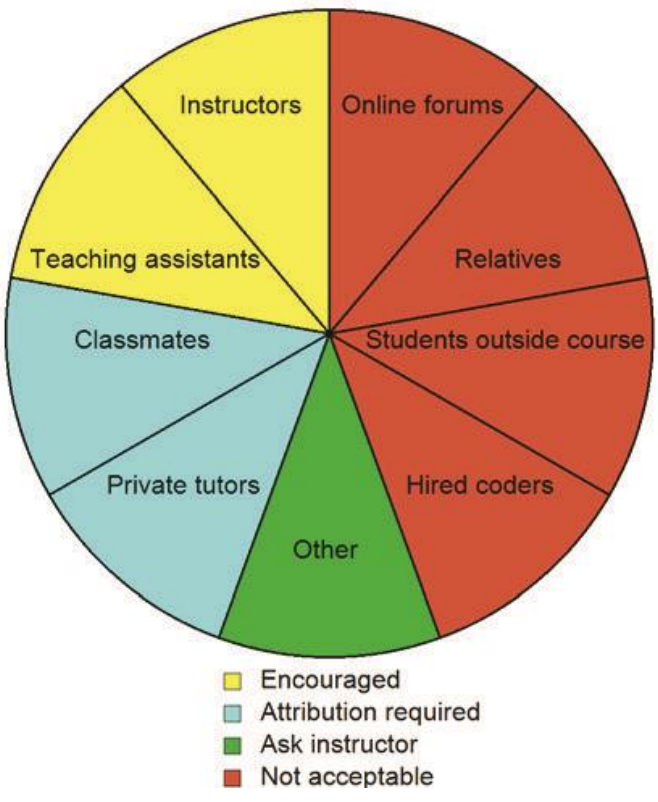
Integrity:

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., code found online) will be dealt with according to the College procedures for handling plagiarism and may result in serious penalties.

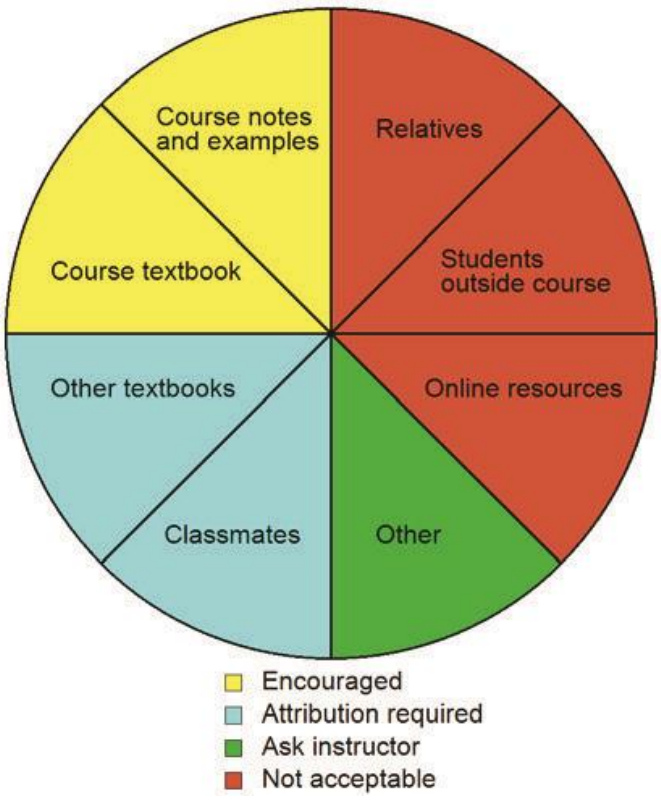
The goals of this assignment include helping you gain understanding of fundamental programming concepts and skills, and future subjects will build on this learning. Therefore, it is important that you develop these skills to a high level by completing the work and gaining the understanding yourself. 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 in the subject channel in Slack, or get **specific** assistance with your own work by talking with your lecturer or tutor.

The subject materials (lectures, practicals, textbook and other guides provided in the subject) contain all of the information you need for this particular assignment. You should not use online resources (e.g., Google, Stack Overflow, ChatGPT, 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.

Assistance: Who can you get help from?
Use this diagram to determine from whom you may seek help with your programs.



Resources: Where can you get code from?
Use this diagram to determine where you may find code to use in your programs.



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 and consistent.		Multiple variable or constant names are not appropriate, meaningful or consistent.		Many variable or constant names are not appropriate, meaningful or consistent.
Use of code constructs SLO1, SLO2 20%	Appropriate code constructs, correct pattern for the problem (right 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.
Commenting SLO2 10%	Helpful block/inline comments and top docstring contains all program details, no 'noise' comments.		Comments contain some noise (too many/unhelpful comments) or some missing program details in top docstring or some inappropriate 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.