CP1401/CP1801/CP5639 2021 SP51 Assignment 1

Task:

For this assessment, you are to plan and then code 3 separate console-based programs in Python 3. This assignment is designed to help you build skills using:

1. Pay Bill: Input, Processing and Output
2. Simple Tax: Decision structures
3. Celsius Calculator: Repetition structures

Do not define any of your own functions or use any code constructs that have not been taught in this subject. 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 below by example. 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/CP5639 Assignment 1
Program 1 - Pay Bill Calculator
Student Name: <your name>
Date started: <date>

Pseudocode:

<pseudocode here>
"""
print("Pay Bill Calculator")
```

Coding Requirements and Suggestions:

- We suggest you work incrementally on these tasks: focus on completing small parts rather than trying to get everything working at once.
- You do **not** need to handle incorrect types in user input. E.g. if the user is asked for the number of hours and enters "none" instead of an integer, your program should just crash. That's fine.
- Sample output from the programs are 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. It should be obvious what parts of the samples are user input.
- You are expected to include useful 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. Do not include unnecessary or many comments as these are just "noise" and make your program harder to read.
- 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 class.

Program 1 - Bill Calculator:

Learning outcome focus: Input, Processing, Output
File name: a1_bill.py

A restaurant is offering meals at 50% discount. A service charge (10%) and GST (7%) apply to the discounted cost. While the service charge applies to the discounted price, note that the GST calculation is based on the total of the discounted amount and the service charge. Write a program that reads in the cost of the meal and displays a detailed receipt. An example is as follows:

The sample output below shows the currency values displayed with two decimal places. This can be achieved using string formatting, like:

```
print("Money be like ${:.2f}".format(value))
```

Sample Output:

```
Enter meal amount ($): 120

Receipt
Cost of meal:   $120.00
50% discount:   $ 60.00
Service charge: $  6.00
GST:            $  4.62
Total amount:   $ 70.62
```

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

Program 2 – Simple Tax:

Learning outcome focus: Decision Structures
File name: a1_simpletax.py

The Python Party wins the election to form the government. It introduces a simpler tax system that works like this:

- If you earn under $10,000, you pay no tax
- If you earn between $10,000 and $20,000, you pay 5% tax on the total amount
- If you earn over $20,000, you pay 10% tax on the total amount

Write a program that prompts user to enter annual salary and displays the tax amount to be paid.

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

Sample Output from 3 different runs:

Run 1

```
Welcome to the Tax System
Enter salary amount: $9000
You do not need to pay any tax.
```

Run 2

```
Welcome to the Tax System
Enter salary amount: $10000
Please pay $500 for your tax.
```

Run 3

```
Welcome to the Tax System
Enter salary amount: $30000
Please pay $3000 for your tax.
```

Program 3 – Celsius to Fahrenheit Calculator:

Learning outcome focus: Repetition Structures
File name: a1_fah_calc.py

Write a Celsius converter program that displays a table of conversions between 2 input celsius values (e.g. 0 and 100) in steps of a 3rd input value. You are expected to search the formula for the conversion from celsius to fahrenheit. The program should error-check the second input to make sure it is greater than the first (loop until it does). The step size must not be negative. It should display the results in well-spaced columns.

Sample output: from 3 different runs:

Run 1

```
Celsius Start: 10
Celsius End  : 5
Error: end value must be bigger than start value!
```

Run 2

```
Celsius Start: 0
Celsius End  : 100
Step size    : -10
Error: step size cannot be negative!
```

Run 3

```
Celsius Start: 0
Celsius End  : 100
Step size    : 10
--------------------
Celsius  Fahrenheit
  0.0        32.0
 10.0        50.0
 20.0        68.0
 30.0        86.0
 40.0       104.0
 50.0       122.0
 60.0       140.0
 70.0       158.0
 80.0       176.0
 90.0       194.0
100.0       212.0
--------------------
```

Submission:

Submit 3 separate Python files, named as in the instructions. You are to zip these three Python 3 code files (*.py) into one file. Name this file FirstnameLastname.zip (e.g. if your name were Darth Vader, the filename would be DarthVader.zip). Submit this zip file to LearnJCU.

Upload your zip file 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.

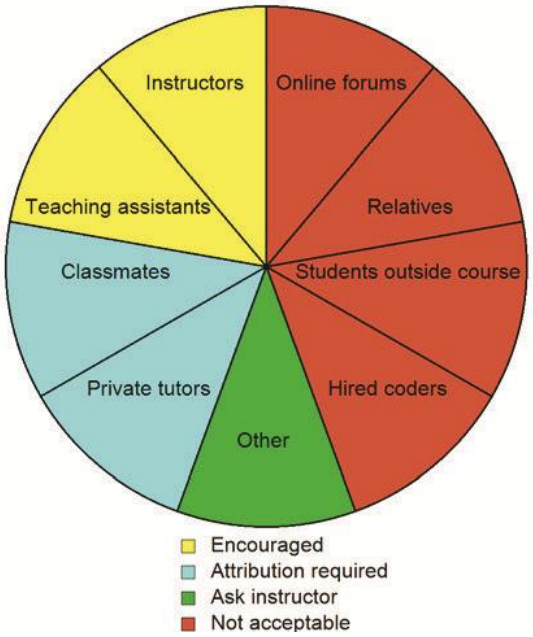
Integrity:

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.

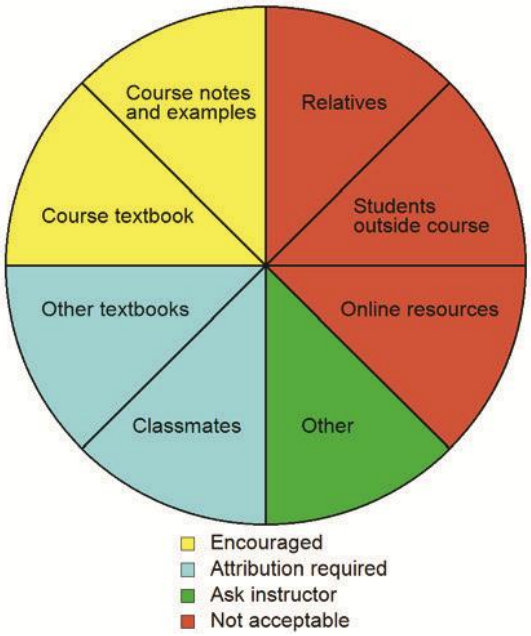
The goals of this assignment include helping you gain understanding of fundamental programming concepts and skills, and future subjects will build on this learning. Therefore, it is important that you develop these skills to a high level by completing the work and gaining the understanding yourself. You may discuss the assignment with other students and get 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 #cp1401 in Slack, or get **specific** assistance with your own work by talking with your lecturer or tutor.

The subject materials (lecture notes, 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. Stack Overflow or other forums) 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 class in order to produce high quality work. Do not just focus on getting your code working. This assessment rubric provides you with the characteristics of exemplary to very limited work in relation to task criteria.
This rubric will be used separately for each of the 3 programs.

Criteria	Exemplary (9, 10)	Good (7, 8)	Satisfactory (5, 6)	Limited (2, 3, 4)	Very Limited (0, 1)
Algorithm	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 (worth double)	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 (including all formatting)	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	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 (worth double)	Appropriate and efficient code use, including good logical choices for calculations, selections and loops.		Mostly appropriate code use but with definite problems, e.g. unnecessary code, poor choice of selections or loops.		Many significant problems with code use.
Formatting	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.
Commenting	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.