CP1200 Assignment 1 - Part 2 - Catering Calculator 2.0

JAMES COOK UNIVERSITY AUSTRALIA

Based on your first assignment, you are to plan and then code the next version of the CP1200 catering calculator, as described by the information and sample output below. Use what you have learned in class, including functions, but do not write any classes.

The program is similar to part 1, but with added options, as shown below. The user enters the number of adults and children and then chooses from packages. There is a 1 in 10 random chance of getting a discount (adults at kids' prices) no matter what catering package is chosen. They can add new packages, and save and load packages to/from a text file called packages.txt. Loading a new file replaces any packages already in memory (added or loaded). Adding a package does not save it to the file unless the user chooses to save, and saving replaces everything in the file with whatever packages are in memory.

An example file (as used for the sample output below) is provided for you on LearnJCU and you must use this format. Each line contains a package name, price for adults and price for children separated by commas (no spaces) and a single new line character. You can assume that the packages.txt file exists and its contents are valid.

Error checking in this version is more comprehensive and your program should not crash for any input. Integer and float inputs must be valid numbers. Check the sample output to see specific examples of limitations on the values for various inputs (e.g. the package choice has to be a valid package number, and prices must not be <= 0). The package name must fit within 16 characters, and when packages are displayed, they should line up in this width.

Sample output from the program is below and forms part of the specification. You should make your program match this **exactly** (except for your name). Do not add or remove anything from the program even if you think it would make it better. You must aim to have yours work and look just like this including spacing and formatting.

Planning:

Write up the algorithms in pseudocode… first. For each function, you need separate pseudocode as well as a function header that shows any parameters. Follow the style of the answers provided for tutorial 5. Please do this in a docstring at the top of your code file. Make sure to write it properly, following the guide to good pseudocode and examples presented in the subject.
You may show this part of the assignment to your tutor during practical time (after finishing your prac work) to get comments or suggestions. You can only get help from your tutors in practical time after your prac work is finished, and only if you show your planning.
No help will be given on code without seeing your planning first.

Program Code:

You need to hand in one complete, functional Python (.py) code file (version 3 not 2) containing appropriate comments (docstring with name, date, description at the top, planning docstring, and inline comments as appropriate). Please name this **LastnameFirstnameA1Pt2.py**

Submission:

Your single .py file should be submitted by uploading it to the drop box in LearnJCU.

Due:

The assignment is to be submitted by the date and time specified on LearnJCU.
Submissions received after this date will incur late penalties as described in the subject guide.

Coding:

Part of the challenge (and assessment) for this assignment is for you to decide what programming constructs to use to solve the problem. Here are some guidelines:

- Use functions appropriately to implement top-down design. Carefully think about the inputs and outputs for functions and ensure that they are useful and reusable, as discussed in lectures.
- One function for each menu option (except quit and instructions) seems reasonable, plus use others as appropriate. E.g. a function like `getValidInt` might be suitable and able to be used in multiple situations. Don't put user input or print statements inside functions unless that's what the function is for. Think about reusability.
- Do not use global variables.
- Include meaningful docstrings for all of your functions as according to the Python style guide.
- Use inline comments (start with # and put them on the line above the code they are referring to) for anything that could benefit from some extra explanation (e.g. the random discount section).
- You should use exceptions (`try: except:`) for error-checking.
- Don't forget to close any files you open, at the appropriate spot(s) in your program.
- Use a default parameter (packages.txt) for the filename. The user is not asked for it, but having it as a default parameter to the functions rather than hard-coding it inside the functions means you could extend the program to use different filenames without having to rewrite those functions at all (this is an example of extensibility – thinking ahead).
- Make sure you learn from the marks and feedback from your first part. Don't repeat any of the same mistakes (if you made any).

Sample Output:

(Bold text below shows user input.)

```
Welcome to the Great CP1200 Catering Calculator 2.0!
Written by Lindsay Ward, April 2012

Menu:
(I)nstructions
(C)alculate Catering
(D)isplay Packages
(L)oad Packages
(S)ave Packages
(A)dd Package
(Q)uit
>>> u
Invalid menu choice.

Menu:
```

```
(I)nstructions
(C)alculate Catering
(D)isplay Packages
(L)oad Packages
(S)ave Packages
(A)dd Package
(Q)uit
>>> I
This program allows you to calculate catering costs based on choice of package and
number of adults and children.
You can load packages from a file, add new ones and save the file for next time.

Menu:
(I)nstructions
(C)alculate Catering
(D)isplay Packages
(L)oad Packages
(S)ave Packages
(A)dd Package
(Q)uit
>>> c
You need to add or load a package first.

Menu:
(I)nstructions
(C)alculate Catering
(D)isplay Packages
(L)oad Packages
(S)ave Packages
(A)dd Package
(Q)uit
>>> d
You need to add or load a package first.

Menu:
(I)nstructions
(C)alculate Catering
(D)isplay Packages
(L)oad Packages
(S)ave Packages
(A)dd Package
(Q)uit
>>> a
Enter package name:
Package name can not be blank and must be less than 17 characters
Enter package name: What do you mean it must be less than 17?
Package name can not be blank and must be less than 17 characters
Enter package name: Fish & Chips
Enter package price per adult: $
Price must be valid and >= $0.01.
Enter package price per adult: $0
Price must be valid and >= $0.01.
Enter package price per adult: $4.95f
Price must be valid and >= $0.01.
Enter package price per adult: $4.95
Enter package price per child: $4.95

Menu:
(I)nstructions
(C)alculate Catering
(D)isplay Packages
(L)oad Packages
```

```
(S)ave Packages
(A)dd Package
(Q)uit
>>> c
Please enter the number of adults:
Number must be valid and >= 0
Please enter the number of adults: zero
Number must be valid and >= 0
Please enter the number of adults: 0
Please enter the number of children: -1
Number must be valid and >= 0
Please enter the number of children: 3
1.     Fish & Chips - $4.95 / $4.95
Which package would you like?:
Number must be valid and >= 1
Which package would you like?: 0
Number must be valid and >= 1
Which package would you like?: 2
Number must be <= 1
Which package would you like?: 1

That will be $14.85 for the Fish & Chips package for 0 adults and 3 children.
Enjoy!

Menu:
(I)nstructions
(C)alculate Catering
(D)isplay Packages
(L)oad Packages
(S)ave Packages
(A)dd Package
(Q)uit
>>> L

Menu:
(I)nstructions
(C)alculate Catering
(D)isplay Packages
(L)oad Packages
(S)ave Packages
(A)dd Package
(Q)uit
>>> D
1.     Super Dooper - $95.00 / $45.00
2.    Middle Ground - $23.95 / $14.50
3.  Not Really Food - $3.78 / $1.95

Menu:
(I)nstructions
(C)alculate Catering
(D)isplay Packages
(L)oad Packages
(S)ave Packages
(A)dd Package
(Q)uit
>>> a
Enter package name: Leftovers
Enter package price per adult: $5
Enter package price per child: $3

Menu:
(I)nstructions
```

```
(C)alculate Catering
(D)isplay Packages
(L)oad Packages
(S)ave Packages
(A)dd Package
(Q)uit
>>> c
Please enter the number of adults: 4
Please enter the number of children: 1
1.     Super Dooper - $95.00 / $45.00
2.    Middle Ground - $23.95 / $14.50
3.  Not Really Food - $3.78 / $1.95
4.        Leftovers - $5.00 / $3.00
Which package would you like?: 4

That will be $23.00 for the Leftovers package for 4 adults and 1 child. Enjoy!

Menu:
(I)nstructions
(C)alculate Catering
(D)isplay Packages
(L)oad Packages
(S)ave Packages
(A)dd Package
(Q)uit
>>> C
Please enter the number of adults: 1
Please enter the number of children: 2
1.     Super Dooper - $95.00 / $45.00
2.    Middle Ground - $23.95 / $14.50
3.  Not Really Food - $3.78 / $1.95
4.        Leftovers - $5.00 / $3.00
Which package would you like?: 1

That will be $135.00 for the Super Dooper (adults at kids' prices!) package for 1
adult and 2 children. Enjoy!

Menu:
(I)nstructions
(C)alculate Catering
(D)isplay Packages
(L)oad Packages
(S)ave Packages
(A)dd Package
(Q)uit
>>> s

Menu:
(I)nstructions
(C)alculate Catering
(D)isplay Packages
(L)oad Packages
(S)ave Packages
(A)dd Package
(Q)uit
>>> q
Thank you for using the Great CP1200 Catering Calculator 2.0.
```

Note the correct use of adult/adults and child/children in this version.

This run shows the random discount applied.

Marking scheme follows on the next page…

Marking Scheme

Take note of the following marking criteria so that you know what needs to be done for marks.
Don't just try and get your program to work, but instead do proper planning followed by careful coding. Make sure you give attention to what is rewarded here.

Requirement	Marks	Out Of
Algorithm – Pseudocode – for each function, including main (clear, complete, well-formatted, consistent, accurate)		**4**
Program Execution Calculating catering costs including random discount		**2**
Displaying packages		**1**
Adding packages		**1**
Loading from file		**1**
Saving to file		**1**
Similarity to sample output (including all formatting) (Remember this is an important part of the specification)		**2**
Quality of Code Use of appropriate, meaningful identifiers (variables, functions, constants)		**2**
Code readability: formatting, indentation, line spacing, etc. (Watch for consistency – e.g. do you always have the same number of lines between functions)		**2**
Useful, descriptive comments: good amount & quality, docstrings (at top and for all functions) and inline comments		**2**
Code Constructs Use of functions: logical choices of functions, parameters, returns		**2**
Methods for handling invalid data		**2**
Appropriate use of loops		**1**
Method for storing packages (choice of data types)		**1**
Default parameter for filename		**1**
Deductions Inappropriate or sloppy code, global variables		**-2**
Incorrect submission (must be 1 Python file)		**-1**
Total		**25**