**Introduction to Programming with Python**

**Homework 4**

***Due At 2:59 pm Thursday, Sept. 27, Before Lecture***

***You will lose 1 point every 10 minutes after that time***

1. (100 points) **More Strings, Lists, Sets, Dicts, and Modules**

In this part of the homework, we will extend the processing of the **expenses.txt** file that we worked on in Homework 3.

1. In IDLE or the Python development environment of your choice, edit the provided **hw4.1.py**. This script contains the part of the code from Homework 3 that:
2. Displays the expenses sorted by ascending dollar amount;
3. Displays the expense categories; and,
4. Displays the mappings from 2-digit month number strings to 3-character month names.

Examine the code in the **hw4.1.py** script to confirm that you understand what it’s doing, then run **hw4.1.py** to confirm that it works.

1. Display the expenses sorted by ascending dollar amount, as in **a.1** above, but neatly formatted rather than as just a sequence of lists. That is, display neatly aligned columns, with expense amounts in 8-character wide fields with 2 digits after the decimal point (so that the decimal points line up vertically), and no **[]** characters or **'** characters. Your output should look like this:

Amount Category Date Description

5.25 supply 20170222 box of staples

6.53 meal 20170302 Dunkin Donuts, drive to...

...

383.75 travel 20170223 flight to Boston, to ...

1247.49 supply 20170306 Dell 7000 laptop/workstation

1. Display the same report again, but this time with the Date column first, Category column next, then Amount and Description, like this:

Date Category Amount Description

20170222 supply 5.25 box of staples

20170302 meal 6.53 Dunkin Donuts, drive to...

...

20170223 travel 383.75 flight to Boston, to ...

20170306 supply 1247.49 Dell 7000 laptop/workstation

1. Those dates are pretty ugly: programmer-friendly perhaps, but not user friendly. Display the same report again, but this time using 3-character month names rather than 2-character month numbers. You will need to slice the date string into pieces, then use your **n2s** dictionary to look up the 3-character month name for the 2-digit month number, and display output that looks like this:

Date Category Amount Description

22-Feb-2017 supply 5.25 box of staples

02-Mar-2017 meal 6.53 Dunkin Donuts, drive to...

...

23-Feb-2017 travel 383.75 flight to Boston, to ...

06-Mar-2017 supply 1247.49 Dell 7000 laptop/...

1. Display the report again, but this time in ascending order by date. For expenses occurring on the same date, the records should be sub-sorted by category. (*Hint:* create a copy of the **data** list of lists with Date moved to the first column and Category moved to the second column, then sort.) The output should look like this:

Date Category Amount Description

22-Feb-2017 meal 79.81 lunch with ABC Corp. ...

22-Feb-2017 supply 5.25 box of staples

22-Feb-2017 travel 43.00 cab back to office

...

23-Mar-2017 util 284.23 Peoples Gas

25-Mar-2017 supply 9.98 Flair pens

1. Display the ascending date order report again, with totals displayed at the end. Display a total for all expenses, and also a total for each category. *Notice* that the money amounts for the totals should line up vertically with the individual expense amounts. Make sure that you use the categories found in the input data file to compute the per-category totals, rather than “hard-coding” the four categories that exist in the current **expenses.txt** input file. Without “giving away” the correct totals, your output should look like this:

Date Category Amount Description

22-Feb-2017 meal 79.81 lunch with ABC Corp. ...

22-Feb-2017 supply 5.25 box of staples

22-Feb-2017 travel 43.00 cab back to office

...

23-Mar-2017 util 284.23 Peoples Gas

25-Mar-2017 supply 9.98 Flair pens

Total: XXXX.XX

meal Total: XXXX.XX

supply Total: XXXX.XX

travel Total: XXXX.XX

util Total: XXXX.XX

1. Save a copy of your **hw4.1.py** code file as **exp\_report.py**. Edit **exp\_report.py** to create a function named **exp\_report()** that takes a file name (a **str**) as an argument, and that processes its input file so as to produce a report as described in part **1.f**, above. The **exp\_report()** function has to do a lot, so it will be very long! The **exp\_report()** function should *only* display the **1.f**-style report, and should *not* display all the other output that **hw4.1.py** displays for parts **1.a** through **1.e** above.

Below the definition of your **exp\_report()** function, test it by calling it with your **expenses.txt** file as its argument:

**exp\_report('***path\_to\_your\_***expenses.txt')**

The output should be the same as for part **1.f**, above.

1. Modify your **exp\_report.py** code file so that the function call

**exp\_report('***path\_to\_your\_***expenses.txt')**

only occurs if **exp\_report.py** is the *main module*, that is, **if \_\_name\_\_ == '\_\_main\_\_'**. Run **exp\_report.py** to test that your modified module works correctly and displays the output report for **expenses.txt** as in **1.f**.

1. Create a new code file, **exp\_test.py**, to test your **exp\_report.py** module using a different input file. The provided **expenses2.txt** file is in the same format as **expenses.txt**, but has different records and different expense categories. Edit your **exp\_test.py** file to import the **exp\_report** module with the abbreviation **er**, and then call the **er.exp\_report()** function with your **expenses2.txt** file as its argument. You should see output like this:

Date Category Amount Description

22-Jul-2018 Entr 79.81 DEF client dinner

22-Jul-2018 Supp 11.25 tape dispenser

22-Jul-2018 Trav 63.00 cab back to office

...

23-Aug-2018 Util 221.83 Peoples Gas

25-Aug-2018 Supp 12.25 Sharpie pens

Total: XXXX.XX

Educ Total: XXXX.XX

Entr Total: XXXX.XX

Meal Total: XXXX.XX

Supp Total: XXXX.XX

Trav Total: XXXX.XX

Util Total: XXXX.XX

***When finished, put your hw4.1.py, exp\_report.py, and exp\_test.py source code files into a .zip file, and upload your .zip file to Canvas.***