**Introduction to Programming with Python**

**Homework 3**

***Due At 2:59 pm Tuesday, Sept. 18, Before Lecture***

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

1. (70 points) **Lists, Sets, and Dicts**

**expenses.txt** (the same file we used in Homework 2) is a small text file describing business expenses. Each line (after the header) gives the money amount, category, date, and description of an expense.

1. Create a Python program file named **hw3.1.py**. In this script, define an empty **list** named **records** (you can say: **records = []**), then read the lines from **expenses.txt** and **append** each line (*excluding* its terminating newline character) to the **records** list. Close the file when you have finished reading it (unless you have used the with form of open, which closes the file for you automatically). Then, add this code to display the lines from **records**:

**for line in records:**

**print(line)**

Confirm that the output is not double-spaced; that is, confirm that each line (**str**) in the **records** **list** does not include a terminating newline character.

1. Make sure that you have closed the **expenses.txt** file, and open **expenses.txt** again. (If you used the **with** form of **open()**, you don’t need to do **close()**: it is done for you automatically.) Create an empty list named **records2**. Read lines from **expenses.txt**. For each line, strip the newline, and **split()** the line at **':'** characters to create a **list** called **columns**. Then, append the **columns** **list** to the **records2 list**: **records2** will be a **list** of **lists**.

Finally, add this code to display the **list** of **list**s **records2**:

**for row in records2:**

**print(row)**

The output from this loop should look like:

**['Amount', 'Category', 'Date', 'Description']**

**['5.25', 'supply', '20170222', 'box of staples']**

**...**

**['8.98', 'supply', '20170325', 'Flair pens']**

1. Recall that you can **sort()** a **list**. Make a copy of **records2** named **r2\_copy**, then **sort()** **r2\_copy** and display its contents with:

**for row in r2\_copy:**

**print(row)**

Is **r2\_copy** sorted in ascending order by dollar amount? Comment.

1. We have two problems with sorting **r2\_copy** by dollar amount. First of all, the header line is part of **r2\_copy**, and sorted along with all the other lines. It turns out that in the utf-8/ASCII character set, all of the digit characters **'0'**, **'1'**, …, **'9'** come ahead of all of the uppercase and lowercase letters, so the header line is *last* in the **r2\_copy**.

(The web page [www.asciitable.com](http://www.asciitable.com) is a good reference for the ASCII character set. The first column of 32 characters are what are called *control characters*, like tab, newline, Ctrl-C, and so on. The second column of 32 characters is mostly punctuation marks and the digits **0** through **9**. The third column is mostly uppercase letters, and the fourth column is mostly lowercase letters. On our systems, sorting is done according to this character set: '0' precedes '9', which precedes 'A', which precedes 'a', and so forth.)

Second, the dollar amounts are represented as strings (**str**) in each data line, so they are sorted as strings: **'123.45'** comes ahead of **'8.76'** because **'1'** comes ahead of **'8'**.

So we need to separate the header from the data records, and we need to convert the dollar amounts from strings (**str**) to numbers (**float**).

From **records2** (which still contains the original data in the original order), make a *copy* of the first item (the **list** of column headers) into a **list** named **header**. Then, make a *copy* of a *slice* of **records2** containing all *except* the first item into a **list** of **list**s named **data**.

***Hint:*** **records2** is a **list** of **list**s, so **records2[0]** is its first item, which is a **list**. To make a copy, use **records2[0].copy()**. Use a similar idea to create a copy of a slice.

If you have done this correctly, these statements:

**print(header)**

**for d in data:**

**print(d)**

should display:

**['Amount', 'Category', 'Date', 'Description']**

**['5.25', 'supply', '20170222', 'box of staples']**

**...**

**['8.98', 'supply', '20170325', 'Flair pens']**

1. Next, loop through **data**, changing the first item of each **list** from a **str** to a **float**. If you have done this correctly, these statements:

**print(header)**

**for d in data:**

**print(d)**

should display:

**['Amount', 'Category', 'Date', 'Description']**

**[5.25, 'supply', '20170222', 'box of staples']**

**...**

**[8.98, 'supply', '20170325', 'Flair pens']**

Notice that the dollar amounts are now **float** values rather than **str** values.

1. Finally, **sort()** **data**. Then, these statements:

**print(header)**

**for d in data:**

**print(d)**

should display:

**['Amount', 'Category', 'Date', 'Description']**

**[5.25, 'supply', '20170222', 'box of staples']**

**[6.53, 'meal', '20170302', 'Dunkin Donuts, drive...'] ...**

**[383.75, 'travel', '20170223', 'flight to Boston...']**

**[1247.49, 'supply', '20170306', 'Dell 7000 ...'']**

The records are now in ascending order by dollar amount.

1. What are the expense categories? Create a **set** containing these. Start with an empty set, like this:

**categories = set() # an empty set**

Then, loop through **data** adding the category from each record to the **categories** **set**.

When done, use these statements to display the categories:

**print('There are', len(categories), 'expense categories:')**

**for c in categories:**

**print(c)**

This *might* display:

**There are 4 expense categories:**

**mean**

**travel**

**suppy**

**util**

But recall that the items in a set are not ordered, so the categories might be displayed in a different order.

***Notice*** that an advantage of using a **set** this way is that new expense categories might be added into the **expenses.txt** file and we will not have to change our code. If new expenses are entered with categories like **'entertain'** and **'charity'**, we can simply re-execute our program and we will get an updated list of the 6 expense categories that now exist.

1. Create a **dict** named **n2s** that uses two-digit month number strings as *keys* and the corresponding three-letter month abbreviation strings as *values*. For example, two of the key/value pairs in **n2s** should be **'01' : 'Jan'** and **'09' : 'Sep'**. Use a loop on the *items* of **n2s** to display a neatly formatted table:

**Key Value**

1. **Jan**
2. **Feb**
3. **Mar**

**...**

**12 Dec**

***When finished, put your hw3.1.py source code file into a .zip file, and upload your .zip file to Canvas.***