Week 4 Review

Submit Assignment

Due Sunday by 11:59pm **Points** 18 **Submitting** a text entry box or a file upload

Overview

When attempting these problems, please keep in my mind the nature of **Academic Honesty** in this course. This week you should submit your response to the Week 4 Review. This will support your review of the topics covered this week and prepare you for writing your program.

Programming Exercises

1. **[mad-libber.py]** A mad lib is a game where you take a sentence and fill in some of the words to make a (typically) silly story. For example, you might make up a sentence "She ate the [noun] and [past_tense_verb]." Then someone else picks a noun and a past tense verb at random, like "alarm clock" and "smiled", and they are put into the sentence to make "She ate the alarm clock and smiled." Usually the mad libs are a much longer story, but this is the general idea.

You are to create your own mad lib. Here is an example:

```
Madlib Maker
name: Paul
place: airport
verb: wait

Old Mc-Paul had a airport, E-I-E-I-O
And on that airport he liked to wait, E-I-E-I-O
With a wait wait here, and a wait wait there, everywhere a wait wait
```

I picked part of a song, and you could too, but your mad lib should:

- have at least three lines or sentences
- have at least three and no more than eight replacements (mine are name, place, and verb) and two of them must be present in more than one sentence (e.g., in the example note how `airport` is lines 1 and 2, and `wait` is in lines 2 and 3)
- *not use the concatenation operator*: +. Why? Because although it is straightforward to do so, your solution would very likely be tedious and would not work for more than just your sentences. String methods exist to make work with Strings more efficient than that. You can solve this simply with sequences and some String methods (maybe **string formatting**?).

HINT: If you find yourself devising an elaborate algorithm involving programming content we have not yet covered (e.g., conditionals), you are overthinking it.

2. **[sales-totaler.py]** Write a program that takes some sales numbers from a file written as dollars (e.g., $1120.47), sums them across rows, and outputs the rows again with the sum at the end. For example for an input file named `17-oct-sales.txt` with these contents:

```
$1120.47 $944.42
$72.29 $588.23
$371.21 $2183.84
```

your program would output to another file the contents:

```
$ 1120.47  $  944.42  $ 2064.89
$   72.29  $  588.23  $  660.52
$  371.21  $ 2183.84  $ 2555.05
```

Notice how in the output file all the numbers are nicely formatted with right alignment. Hint: you will probably need to use **splitting**, **string slicing**, converting data types, and **string formatting**.

You program should prompt a user to input a file containing the sales numbers. Then prompt the user to enter a file name for outputting the sales numbers and their totals, formatted as indicated in the example above. Here is an example interaction:

```
Enter sales file name: 17-oct-sales.txt
Enter name for total sales file: 17-oct-totals.txt

Done writing totals to 17-oct-total.txt
```

You will find the `17-oct-sales.txt` file **here**.

Submission

Please post all necessary .py files to Canvas and include your answers to the questions under the "Canvas Submission" banner in the textbox provided.

Canvas Submission

When you submit this assignment here in Canvas, I would like you to answer the following question(s):

- How many hours do you estimate you used completing this assignment?
- What was easiest for you when completing this assignment?
- What was the most difficult challenge you experienced when completing this assignment?

Week 4 Review Rubric

Criteria	Ratings		Pts
1. Mad Libber: At Least 3 Inputs Used	**2.0 pts** **Full Marks**	**0.0 pts** **No Marks**	2.0 pts
1. Mad Libber: At Least 3 Lines	**2.0 pts** **Full Marks**	**0.0 pts** **No Marks**	2.0 pts
1. Mad Libber: Uses Replacements in more than one line	**2.0 pts** **Full Marks**	**0.0 pts** **No Marks**	2.0 pts
1. Mad Libber: Does not use concatenation	**2.0 pts** **Full Marks**	**0.0 pts** **No Marks**	2.0 pts
2. Sales Totaler: File Input	**2.0 pts** **Full Marks**	**0.0 pts** **No Marks**	2.0 pts
2. Sales Totaler: Parses Sales to Numeric Values, Computes Totals	**3.0 pts** **Full Marks**	**0.0 pts** **No Marks**	3.0 pts
2. Sales Totaler: File Output has correct values	**2.0 pts** **Full Marks**	**0.0 pts** **No Marks**	2.0 pts
2. Sales Totaler: File Output Right Aligned	**2.0 pts** **Full Marks**	**0.0 pts** **No Marks**	2.0 pts
Answers Questions in Canvas Submission	**1.0 pts** **Full Marks**	**0.0 pts** **No Marks**	1.0 pts
			Total Points: 18.0