Assignment 1 Instructions

Functions, Loops and If Me + You = Love

Download all starter files on our course website: a1_files.zip

NOTE: For all assignments, your code should follow good style. The Python style guidelines for this course, and examples of the function design recipe, are available on our course webpage (under the Assignments section).

Part 1. Long code is wrong code

Your friend, Bob, has written a random program, playing around with the basics of Python. But for such simple operations, he feels as though his file is way too long. He remembers hearing from someone that, when you're trying to program something really simple, you'll often find that "long code is wrong code". Even if the code is working properly, making it shorter would make it much better and cleaner to handle.

Bob's file is named a1_part1.py. All Bob wants is to make the file shorter. He doesn't want the way the code works to change. Could you clean up this code for him, **using functions** and getting rid of other useless things? (For example, by putting all the mathematical calculations into one equation instead of splitting each step up, or, when checking if a boolean value is True, instead of writing the redundant code 'if (bool == True)', simply writing 'if (bool)', etc.).

Try to get the file down to less than 40 lines.

For any functions you add in, make sure to follow the **function design recipe** we discussed in class. You **must** use some functions to get full marks for this part.

Part 2. Bob's Love Compatibility Calculator

Bob's back again, and this time, he's in love. He noticed a new girl in his Math Proofs class and even though they haven't really talked yet, he's convinced they are meant for each other. Just to be sure though, Bob wanted to double-check their compatibility. He remembered a quite accurate, fool-proof compatibility calculation kids used to do back in the day – something involving a complex algorithm that counted the number of times the

letters in the word "LOVES" appears in two names, and then added digits together to calculate a percentage[1].

Much to his surprise, that completely inaccurate, childish compatibility calculator stated that 'Bob Y' and 'Bobbette Z' have a compatibility of only 44%!

This was unacceptable, Bob decided. The issue, he decided, was that considering only names to measure compatibility left out one of the most important factors that truly determines how well-matched two people are. Their zodiac signs.

So he decided to make his own, better version.
And he needs your help to finish it.

Finish the code in bobs_compatibility_calculator.py (included in a1_files.zip), as directed by the docstrings and examples in the file.

If your calculator is working properly, it should give the following output for the given input (try these out to test your code; you would have to run your program again to test out each set of people):

(1)

```
Give me your first and last name: Bob Y
Give me your birthdate in the format DD-MM-YYYY: 30-08-1998
=====
Give me your crush's first and last name: Bobbette Z
Give me your crush's birthdate in the format DD-MM-YYYY: 01-09-1998
You are 85.0% compatible in love!
```

(2)

```
Give me your first and last name: Sadia Sharmin
Give me your birthdate in the format DD-MM-YYYY: 24-08-1957
=====
Give me your crush's first and last name: Ryan Gosling
Give me your crush's birthdate in the format DD-MM-YYYY: 12-11-1980
You are 110.0% compatible in love!
```

---

[1] This is an actual thing kids used to do. If you're curious about this method (and maybe want to try it out on your own Bobbette ;)), I can tell you about it if you ask, but it's not really relevant to the assignment, so I didn't want to go into it in detail on this assignment page.

(3)

```
Give me your first and last name: Homer Simpson
Give me your birthdate in the format DD-MM-YYYY: 12-05-1955
=====
Give me your crush's first and last name: Marge Simpson
Give me your crush's birthdate in the format DD-MM-YYYY: 19-03-1955
You are 140.0% compatible in love!
```

What you will be handing in:

Submit all the following files on MarkUs by Sunday, January 28, 11:59 P.M. (midnight):

1. **`a1_part1.py`** with shorter, cleaner code
2. **`bobs_compatibility_calculator.py`**

Assessment:

Your mark will be based on both correctness as well as cleanliness of code (following the Python style guidelines, following design recipe, avoiding repetitive code, etc.).

Criteria that you should keep in mind:

- **Correctness**: Your functions should perform as specified. Correctness, as measured by our tests, will count for the largest single portion of your marks.
- **Formatting style**: Make sure that you follow the Python style guidelines that we have introduced! (see course website, Assignments section)
- **Programming style**: Your variable names should be meaningful and your code as simple and clear as possible. Where possible, you should avoid duplicate code by calling other functions within your file as helpers.
- **Commenting and docstrings:** We want to see great docstrings (this applies for part 1; part 2 docstrings are already given, you don't have to modify them) and internal comments. Again, follow the Python style guidelines.