Lecture 1 - Exercises

1S - Syllabus

1. What are you required to do to prepare for each class?
2. Where can you find the lecture videos, slides, exercises, and problems?
3. How many missed classes are allowed without making arrangements with Scott?
4. What is a slip day, how many are you allowed?
5. Can slip days be used on labs?
6. Should you expect all programming assignments to take you approximately the same amount of time?
7. According to the academic honesty policy, which of the following are permitted?
    - Talking about your code with your classmates.
    - Submitting someone else's program as your own.
    - Copying a few lines of someone else's code into your solution, if you understand those lines in detail.
    - Looking at a classmate's code, then immediately sitting down and typing out a very similar program, but with different variable names.

1A - Computers and Hardware

1. Computers have (at least) two components that store information: main memory (RAM) and secondary storage (e.g., hard drives). Why are both needed?
2. Which hardware component actually executes software instructions? Can it directly execute Python instructions?

1B - Algorithms and Pseudocode

3. Write pseudocode for a program that prompts a user for non-negative integers. If a user inputs a negative number, the program ends and outputs the sum of the input positive integers.

1C - Calling Functions, `print`, and `input`

4. Which of the following lines of Python contain a comment?
    1. `# Author: Scott Wehrwein`
    2. `print("Hello!") # greet the user`
    3. `input("Enter your favorite 2-digit number (##): ")`
5. What does the following code print?

```
print("CSCI", 99 + 42, "at WWU")
```

6. What does the following code print?

```
print("CSCI", "99 + 42", "at WWU")
```

7. How many arguments are given to the following call to the print function?

```
print("CSCI", 99 + 42, "at WWU")
```

Problem

Write a **math quiz program** that works as follows: it begins by printing an arithmetic problem of your choosing. Then, it prompts the user to enter an answer. Finally, once they have pressed enter, the program prints a message showing the correct answer. *Note*: we don't yet know how to check if the user's answer was correct, but we'll get there soon! A couple sample runs of such a program are shown below; note that the number at the end of the first line is what's typed by the user before they press enter:

```
What is 4 * 6? 24
4 * 6 is 24.
```

```
What is 4 * 6? 22
4 * 6 is 24.
```