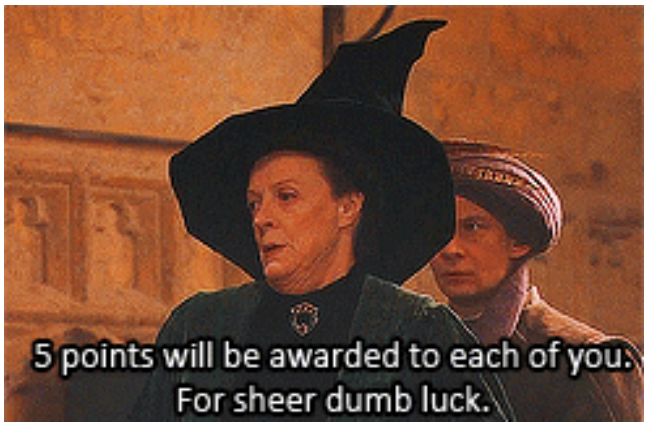
CMPUT 174 Lab 5: Harry Potter Spells Typing Trainer

Theme of this lab: Harry Potter Spells!



Are you a fast typist? Or maybe you'd like to improve your typing speed? This lab will help you!

No, not because you'll be typing lots of Python code (well, not *only* because of it). In this lab, you will create a program that will help you type faster! A Harry Potter Spells Typing Trainer!

We hope no one casts ***Obliviate*** on you, so you remember what user-defined functions are.

Good luck!

Learning Outcomes

- Employ user-defined functions to decompose computational problems
- Identify appropriate parameters and return values for user-defined functions
- Program complex logic with interconnected user-defined functions
- Employ control structures (selection, iteration) to solve computational problems
- Use manual tests to ensure the correctness of a program
- Apply evolutionary prototyping to design programs step by step
- Increase your typing speed

Software Quality Requirements

For this lab, you must apply **all** software quality requirements, **except** Section 7 (User-defined Classes).

Tasks

The following tasks are versions of the same problem. We're learning to write code in an incremental way. Start with the simplest version, and then add more functionalities to make your code more complex. Please do the tasks in order, starting with the first one.

1. Version 1: Aberto
2. Version 2: Bombardo
3. Version 3: Confundo
4. Version 4: Diffindo

Reflection Questions

Once you're done coding, use these questions to think about your code. It's an essential part of learning because we can never write good code if we don't think about the problem and consider different ways of solving it.

When you demo your lab, a TA may ask some of these questions.

1. Some user-defined functions can be quite short, with only one or two lines. What are advantages and disadvantages of having such short functions?

2. What are the advantages of using the `with` clause to open files?

3. Code templates provided in this lab include type annotations. For example:
`def get_random_spell(spells: list[str]) -> str:`
What would happen if we pass a list of `int` instead of a list of `str` to this function? Would Python throw an error?

4. Software quality requirements state that a user-defined function should have 12 or fewer statements. We know that it's mainly a rule of thumb, and in some cases, your function might exceed this recommendation. However, why do you think we recommend keeping your functions small enough?

5. In ver. 4, you use the `time` built-in module to find the typing speed. Imagine you need to find the overall time the user spent using the program. How would you do that?

6. In ver. 3, you implemented the functionality that allows user to play the game multiple times. Which loop did you use (`while` or `for`)? How would you refactor the code to use another type of the loop?

7. Does the structure of user-defined functions seem natural to you? Or would you design your program differently? How would you do it?

Resources

- The following text file contains a list of spells: spells.txt
- Please use the following text file for instructions: instructions.txt
- Curious to know more about Harry Potter spells? Check out the Harry Potter Wiki page.

Marking

The are no part marks, no in-between marks

4/4	Your code clearly meets all requirements of **Version 4** and all software quality requirements. You clearly understand your code and your answers are correct.
3/4	One of the following: a) Your code meets all requirements of **Version 3** and all related software quality requirements. You clearly understand your code and your answers are correct. b) Your code meets most **Version 4** requirements and most software quality requirements; it runs and does what is expected. However, some minor requirements are missing, or some details in your answers are missing or incorrect.
2/4	One of the following: a) Your code meets all requirements of **Version 2** and all related software quality requirements. You clearly understand your code and your answers are correct. b) Your code meets most **Version 3** requirements and most software quality requirements; it runs and does what is expected. However, some minor requirements are missing, or some details in your answers are missing or incorrect.
1/4	One of the following: a) Your code meets all requirements of **Version 1** and all related software quality requirements. You clearly understand your code and your answers are correct. b) Your code meets most **Version 2** requirements and most software quality requirements; it runs and does what is expected. However, some minor requirements are missing, or some details in your answers are missing or

	incorrect. c) You put effort into your lab assignment, but your code doesn't run at all or runs with major problems. Missing major requirements, or your answers are mainly incorrect.
0/4	One of the following: a) Incomplete, or very insufficient code, or no submission. b) Code submitted but no show, or no answers, or irrelevant answers.