# **CMPUT 174 Lab 3: Klingon Quiz Problem**

Theme of this lab: Star Trek. And it’s fascinating.



## **Learning Outcomes**

* Apply logically-controlled (while) loops to implement repetition
* Construct increasingly complex control structures (including different kinds of iteration and conditional statements) to manage the control flow of the program
* Choose appropriate data structures to manipulate data efficiently
* Use file input-output to read data from disk
* Use manual tests to ensure the correctness of a program
* Apply evolutionary prototyping to design programs step by step
* Discover Klingon 🖖

*To create this program, you may need to use a few programming language features that have not been used yet in class. Discovering new language features and how to use them is an integral part of problem-solving in computing science and an essential skill that you should learn. Think about what you need to do, search the web for Python 3 programming examples, and/or use the Python documentation to help you find the programming constructs that you need. If you get stuck, ask a TA for help/hints about the programming constructs you need to use.*

## **Software Quality Requirements**

For this lab, you must apply the following requirements:

* 2. Comments (except 2.4)
* 3. Names
* 4. Repetition
* 5. Limiting Literals

## **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.

### [Version 1: Things are only impossible until they’re not](https://docs.google.com/document/d/1kX-yt1ZJl7KaQSHdEm2CHlmh7q7ncSUPITPUWpVI6-E/edit?usp=sharing)

### [Version 2: Without freedom of choice there is no creativity](https://docs.google.com/document/d/1fYsf6V6N-jj6UN9bRgseSNwPMLihAtkfck4pLECdqZY/edit?usp=sharing)

### [Version 3: Logic is the beginning of wisdom, not the end](https://docs.google.com/document/d/1caUJ1LncbIMDT24F-7Z8hGUPAGJWvqIQH5lvng6qq4A/edit?usp=sharing)

### [Version 4: Live long and prosper!](https://docs.google.com/document/d/19e8GQLIYEC6A9sim9PpklMUofV8w_Ow7KGgbqI9ZD6k/edit?usp=sharing)

## **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. Why do you think we need to close a file once we’ve finished working with it? How can the with clause help?

2. When you read data from the text file, you need to *parse* each line, splitting it into separate elements. How do you do it in your code? Do you think it’s the most efficient way?

3. Each line of the text file that you’re using has a consistent format: <klingon-word> <english-word>. Imagine that your text file does not have a consistent format and looks like this example:

| batlh honor be irritable bergh chargh:conquer |
| --- |

How would you approach parsing such a file?

4. For simplicity, we removed two Klingon consonants: ng and tlh. It’s incredibly hard to pronounce the latter one, but it was not the reason. Consider your code and think why did we remove these specific consonants?

5. Imagine Python didn’t have string slicing. How would you generate a hint in version 3?

6. You’re likely using multiple while loops in this lab. Now, try to rewrite some of your while loops as for loops. Do you think the code is now more or less readable and maintainable?

7. How did you apply the concept of mutability to implement hints in version 4?

## **Resources**

* If you’d like to learn more Klingon, here is the full Klingon-to-English Dictionary that we used to create this lab: <https://github.com/warkruid/anki-klingon/blob/master/klingon-english.csv>.
* If you’re curious about pronunciation of the Klingon words, see the following link: <https://youtu.be/YjROGAY19pU>.

# **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:   1. Your code meets all requirements of **Version 3** and all related software quality requirements. You clearly understand your code and your answers are correct. 2. 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:   1. Your code meets all requirements of **Version 2** and all related software quality requirements. You clearly understand your code and your answers are correct. 2. 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:   1. Your code meets all requirements of **Version 1** and all related software quality requirements. You clearly understand your code and your answers are correct. 2. 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. 3. 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:   1. Incomplete, or very insufficient code, or no submission. 2. Code submitted but no show, or no answers, or irrelevant answers. |