Applications

- 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

For each program do not over complicate it and please include line comments describing what the line is doing

Tasks

The following tasks are versions of the same problem. 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: Things are only impossible until they're not

Write a program that asks the user to translate the word "computer" into Klingon and checks the answer.

What to do

1. Read data from klingon-english.txt
2. Ask user to translate `computer` into Klingon *(it's line #3 in the file)*
3. Print `Correct` if the user's answer is correct, or `Sorry, you're wrong! The correct answer is De'wI'` if the user's answer is wrong

You **must** read data from the provided text file – that is absolutely essential in this lab assignment. You can not simply hardcode the word `De'wI'`, it will be considered incorrect.

To-do

1. Carefully study the text file (`klingon-english.txt`). Remember that not all whitespace characters are the same! They might look similar but be different in nature.
2. Choose an appropriate data structure to store Klingon and English words. There is more than one way to do it. Do not overcomplicate this. The data structures we've learned in class will be enough for this program. Using two parallel lists (one list for Klingon words and one list for English words) is one of the alternatives.

Testing

To make sure your program works correctly, you should test it.

Test Case 1

Run your program with `python klingon-quiz1.py`. Type `De'wI'` and press Enter. Your program should print:

```
Correct!
```

Test Case 2

Run your program with `python klingon-quiz1.py`. Type `dewi` and press Enter. Your program should print:

Sep 24, 10:22 am: Output message corrected (to make it consistent with the demo video):

```
Sorry, you're wrong!
The correct answer is De'wI'.
```

2. Version 2: Without freedom of choice there is no creativity

Here are Klingon consonants: **b, ch, D, gh, H, j, l, m, n, p, q, Q, r, S, t, v, w, y, '**

Klingon experts will notice that two consonants ("ng" and "tlh") are missing. To keep things simple, we omitted them.

For each consonant, the text file that you're using in this lab, contains exactly one word that starts with that consonant.

In this version, your program should ask the user to choose a Klingon consonant

What to do

1. Read data from klingon-english.txt
2. Ask the user to choose a Klingon consonant they want to practice with. Ask again if the user's answer is not a valid Klingon consonant, until the user enters a valid consonant.
3. Find a Klingon word that starts with the chosen consonant *(the text file contains only **one** word that starts with any given consonant, so you don't need to use the* `random` *library)*
4. Ask the user to translate the chosen word into Klingon
5. Print `Correct` if the user's answer is correct
6. Print `Sorry, you're wrong! The correct answer is ...` if the user's answer is wrong

You **must** read data from the provided text file – that is absolutely essential in this lab assignment. You can not hardcode the words.

To-do

1. Some Klingon consonants are represented by more than one English letter. One consonant (`'`) is not an English letter.
2. Your program should not proceed to the later steps until the user enters a correct consonant. That's the job for some `while` loops!

Testing

To make sure your program works correctly, you should test it. At this point, there are many test cases that you should consider. Refer to the screen recording above for some ideas. Come up with your own test cases for this program.

3. Version 3: Logic is the beginning of wisdom, not the end

In this version, your program should show a hint after the first attempt and, with the hint, give two more attempts to give the right answer.

What to do

Already implemented:

1. Read data from klingon-english.txt
2. Ask the user to choose a Klingon consonant they want to practice with. Ask again if the user's answer is not a valid Klingon consonant, until the user enters a valid consonant.
3. Find a Klingon word that starts with the chosen consonant *(the text file contains only* ***one*** *word that starts with any given consonant, so you don't need to use the* `random` *library)*
4. Ask the user to translate the chosen word into Klingon
5. Print `Correct` if the user's answer is correct
6. Print `Sorry, you're wrong!` if the user's answer is wrong

In this version:

7. If the answer is incorrect, show a hint: the first and last characters of the correct Klingon word. When showing a hint, replace all other characters with a star (`*`)
8. If the answer is still incorrect, show the same hint once again.

Refactor in this version:

9. Print `The correct answer is ...` if **all three** user's answers are wrong

You **must** read data from the provided text file, that is absolutely essential

If any problems occur consider the following:

1. You can use slicing to generate a hint. Think:
    a. How to get the first character of a string?
    b. How to get the last character of a string?
    c. How to generate a string containing a certain number of star (`*`) characters?
    d. How to concatenate these elements?

Testing

To make sure your program works correctly, you should test it. At this point, there are many test cases that you should consider. Refer to the screen recording above for some ideas. Come up with your own test cases for this program.

4. Version 4: Live long and prosper!

Klingon Quiz Problem Version 4: Live long and prosper!

This is the final version of your Klingon Quiz program.

Now, The goal is to show an additional (random!) hint after the second incorrect answer.

What to do

Already implemented:

1. Read data from klingon-english.txt
2. Ask the user to choose a Klingon consonant they want to practice with. Ask again if the user's answer is not a valid Klingon consonant, until the user enters a valid consonant.
3. Find a Klingon word that starts with the chosen consonant *(the text file contains only **one** word that starts with any given consonant, so you don't need to use the* `random` *library)*
4. Ask the user to translate the chosen word into Klingon
5. Print `Correct` if the user's answer is correct
6. Print `Sorry, you're wrong!` if the user's answer is wrong

You will refactor in this version:

7. If the answer is incorrect, show **the first** hint: the first and last characters of the correct Klingon word. When showing a hint, replace all other characters with a star (`*`)
8. If the answer is still incorrect, show **the second hint**: the first and the last characters plus an extra random character of the correct Klingon word.

Already implemented:

9. Print `The correct answer is ...` if **all three** user's answers are wrong

Demo

https://asciinema.org/a/doUtAbuY0eLrRL80SsU5P7sBd

Testing

To make sure your program works correctly, you should test it. At this point, there are many test cases that you should consider. Refer to the screen recording above for some ideas. Come up with your own test cases for this program.

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.