Homework 1

The examination of the module *31-M29 Data Science* consists of a portfolio of four homeworks and a final exam. This is the first homework. Please hand in your solution as a single .ipynb file by via "LernraumPlus" by December 22, 2020. Note that by handing in a solution you begin the examination of the Data Science module.

Snowman (22.5 Points)

Snowman is new paper-and-pencil game that is meant to be played in the winter. One player thinks of a word and the opponent tries to guess it by suggesting letters within a certain number of guesses. Your task is to develop a Python implementation of Snowman that allows a single player to play against a computer-controlled opponent. The implemented game should work as follows: The player thinks of a word and enters the number of letters of the word into the program (but not the word itself). Now the computer-controlled opponent suggests a letter and the player reveals if the suggested letter is part of the word (and at which positions). The game ends after the computer-controlled opponent suggests 8 letters that are not part of the word (in this case the player wins) or if all letters of the word have been suggested by the opponent (the player loses).
Your implementation should offer the following features:

- Implement a function "`start_game(dummy_player=None)`" that starts a game. The function should return `True` if the player looses and `False` if the player wins. A human player should be asked for input via the `input()` function. At the beginning of the game the player is asked for the number letters of the word. In the following iterations the player is asked for the (zero-indexed) positions of the suggested letters in the word. The player input can either be an empty string (i.e., "") (if the letter is not part of the word), a string representing a single integer (e.g., "2") (if the letter occurs in the word once), or a string representing a space-separated list (e.g., "2 3") (if the letter occurs in the word more than once). Do not ask the player for any other input!

- The function "`start_game(dummy_player = None)`" accepts a DummyPlayer object as an argument. We already provide you with a notebook that implements the class DummyPlayer: `https://colab.research.google.com/drive/1dSwahknnbHWtELoLtIoVlhtSANlEin-C?usp=sharing`. The DummyPlayer class allows to simulate a user and we will use it to evaluate your code. Do not change the implementation of DummyPlayer and make sure that your code works properly with our implementation.

- Do not visualize the snowman. However, show the number of incorrect guesses and also a representation of the word including the correctly guessed letters of the opponent (e.g. _ _ _ a _).

- The implementation should only be able to handle the 1,000 most common English words (see `https://gist.github.com/deekayen/4148741`).
- **Update 9.12.20**: Your implementation should only consider words with at least 3 characters. Apostrophes in a word should be removed (e.g. treat the word "don't" as "dont").

Example

How many letters does the word have?
>>> 6 (User or dummy user input)
_ _ _ _ _ _ [Remaining wrong guesses: 8]
I guess: a. What are/is the position(s) of the letter a in the word?
>>> (User or dummy user input)
_ _ _ _ _ _ [Remaining wrong guesses: 7]
I guess: o. What are/is the position(s) of the letter o in the word?
>>> 4 (User or dummy user input)
_ _ _ _ o _ [Remaining wrong guesses: 7]
I guess: e. What are/is the position(s) of the letter e in the word?
>>> (User or dummy user input)
_ _ _ _ o _ [Remaining wrong guesses: 6]
I guess: p. What are/is the position(s) of the letter p in the word?
>>> 0 (User or dummy user input)
P _ _ _ o _ [Remaining wrong guesses: 6]
...

Collaboration & code re-use policy

Working in groups is not permitted. Do not share your code with the other students. You are only allowed to discuss high level concepts with the other students. Plagiarism will result in 0 points. Whenever you copy pieces of code from the internet (e.g., stackoverflow) or other sources make the source clearly visible in your code using comments. The failure to do so will also result in the deduction of points. It is perfectly OK to use code snippets (consisting of only a few lines of code) that implement general purpose functionalities from the internet or other sources. However, it is not permitted to re-use large pieces of code or to re-use pieces of existing implementations of the game (or of similar games).

Grading

Grading is based on the following criteria:

1. **Specified requirements are fulfilled (10 Points)**
   (a) Does the overall game logic work as intended (e.g., the opponents suggests letters and asks for player feedback)?

(b) Does the implementation handle the specified user input syntax.

(c) Is it possible to win a game?

(d) Is it possible to lose a game?

(e) Are all other specified requirements fulfilled?

2. **Playing experience (2 Points)**:

   (a) Are the number of incorrect guesses and the representation of the word shown after each round?

   (b) Is it clear to a new player what input he needs to provide (including the expected syntax of the input)?

3. **Computer-controlled opponent (5 Points)**:

   (a) How many of the 1,000 most common English words is the opponent able to guess?

4. **Code quality (5.5 Points):**

   (a) How well is the code structured (e.g., by using functions)?

   (b) How difficult is it to understand the code (including whether or not there are comments)?

   (c) Are there other code quality issues?

Late Submission Policy

Late submissions are penalized 5% per started hour after the deadline (i.e., a solution that is handed in 122 minutes after the deadline is penalized 15%). Note that these should still be handed in through LernraumPlus.