EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

PHILOSOPHISCHE
FAKULTÄT
Seminar für Sprachwissenschaft

Data Science for Linguists
Summer 2025
Johannes Dellert

Assignment 00:
Getting Used To Python

handed out: 16 April, 14:00
to be submitted by: (no date, this is for practice)

The purpose of this demo exercise sheet is to give you the opportunity to get some practice in basic Python. You will be working with the file `de-word-forms.txt`, a list of the 200,000 most frequent German word tokens extracted from a subtitles corpus, and we would like to analyse it for recurring patterns, acting as if we do not yet know anything about the language.

Task 1: Reading the File Contents into a List

a) Create a new script file `datsci_ex00.py` (as an empty plain text file) and place it next to the downloaded `de-word-forms.txt` in some directory (suggestion: a new directory called `ex00` within a directory where you will do all the assignments for this course).
b) Use the method shown on the slides (`open()` in read mode within a `with` statement) in order to open a file handle, and use the handle to extract all the word forms from the file. Conveniently, there is exactly one word form per line (though you need to call the method `strip()` on each one). Store the wordlist in a Python list object called `word_list`, and print the first five entries to the terminal in order to check whether everything worked as planned. Your output should show the words *ich*, *sie*, *das*, *ist*, and *du*.

Task 2: Sorting and Counting

a) Sort the word list alphabetically, and store the result as a new list under the name `alphabetical_list`. To make sure it worked, extract and print the first ten and the final ten entries to the terminal.
b) We would like to use this sorted list in order to extract statistics about how many word forms start with which letter. To do this, you need to loop over all the words in the sorted list. Within the loop, you will need a string variable to remember the letter you are currently processing, and a count variable which you update to keep track of how many words with the same first letter you have already seen. In each iteration of the loop, you check whether the first letter of the new word is identical to the one you are tracking. If it is identical, all you do is increase the count by one. If it is not, you print the old letter with the count to the terminal, reset the count to zero, and remember the new letter instead.

Task 3: Finding Frequent Suffixes

In order to get a first impression of potential morphological patterns, we want to keep track of how often different letter combinations occur. For simplicity, we are going to focus on suffixes of length three.
a) Create an empty dictionary `suffix_to_forms`. We will use it in order to map each suffix to the set of word forms in which that suffix occurs. In English, `"ing"` would be mapped to a set containing `"wing"`.
b) In order to populate the dictionary, we again process all word forms in a loop. For each word form, extract the suffix (the last three characters), and check whether it already exists as a key. If not, store a new empty set as the value for the new key. Add the word form to the set stored under the suffix.
c) Go through the suffixes in alphabetical order. Each potential suffix which occurs in more than 200 word forms (size of the set stored!) is a good candidate for a meaningful suffix, so you print it to the terminal followed by a colon and a comma-separated alphabetically sorted list of all forms in which it was found.

Task 4: Saving Your Results to an Output File

Extend your script in such a way that it can store the list of potential suffixes you have detected to a new file `suffix_candidates.txt`, again opening the file handle within a `with` statement as we did in Task 1, only this time in write mode.