# Task 1: [3 marks]

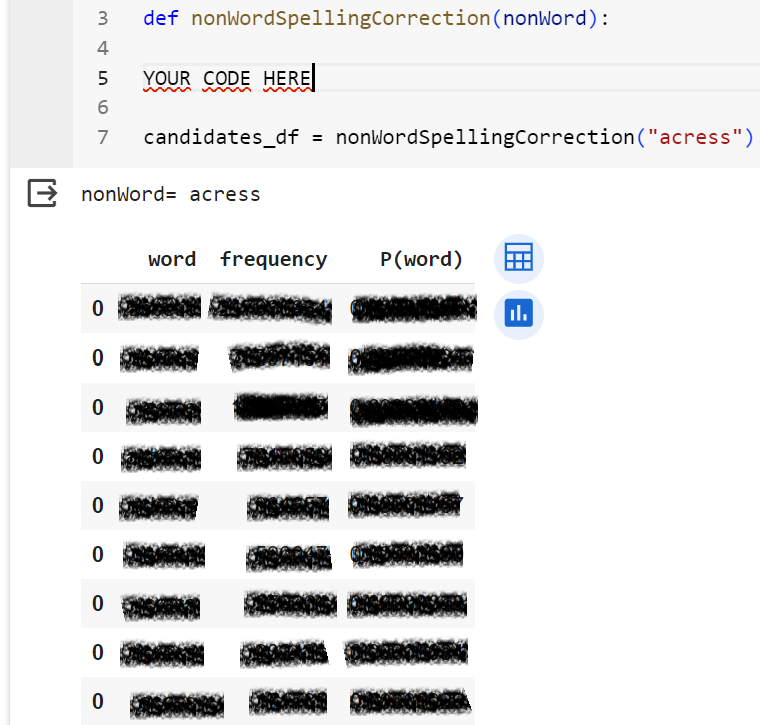
1. Complete the following code snippet to download [Peter Norvig’s unigram corpus](https://norvig.com/ngrams/count_1w.txt), and load it into a dataframe, *unigrams\_df*

**YOUR CODE HERE**

display(unigrams\_df.head(100))

1. Write a simple spelling correction function that takes a nonWord string as input (i.e. non-word misspelling such as “acress”), and prints out a list of candidate corrections.

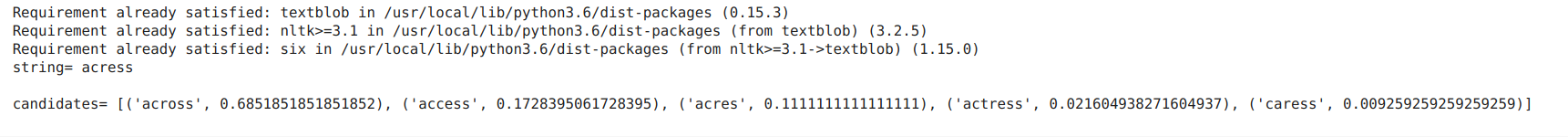
* Hint 1: only use the language model/priors, i.e. the unigram counts stored in *unigrams\_df*, and Ignore the channel/error model probabilities.
* Hint 2: use the Damerau-Levenshtein Distance function in the [jellyfish library](https://github.com/jamesturk/jellyfish) for candidate generation.



# Task 2: [0.5 mark]

Use the TextBlob Library (<https://textblob.readthedocs.io/en/dev/>) to write a function that takes a string (e.g. acress) as input and returns a list of (word, confidence) tuples with spelling suggestions.

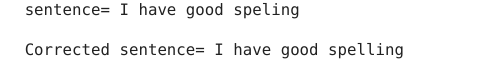
Sample output:



# Task 3: [0.5 mark]

Use the TextBlob Library (<https://textblob.readthedocs.io/en/dev/>) to write a function that takes a sentence as input and corrects its spelling.

Sample output:



## </END OF ETIVITY>

TODO (for lecturer):

* Level of difficulty (easy/fair/hard)
* <http://norvig.com/spell-correct.html>
* 3&4 too easy?
* Implementing Damerau-Levenshtein Distance instead of using a lib