Compact Course Python
Exercise 5

1 Words and POS tags

On the course homepage (check the code samples), you'll find the file `A00-pos-lines`. It's an excerpt from the British National Corpus, in which words are marked with their Part Of Speech tags (format is `wort$tag`). It's one word-tag pair per line. ''Words" might consist of more than one single string (multi word expressions like New York); in this case, the parts are separated with spaces. POS tags always consist of a single word. Punctuation marks count as words (and are tagged).

Write a program that reads the file (either directly from the homepage or from your file system) and records the frequency of the words. The program shall write this information to a file, i.e. list the absolute frequency for each word and the frequency per POS tag, roughly like that (numbers are just exemplary):

```
name   17      NN1:12  VVT:5
```

Extra task 1: Configure your program such that reads the file location from the command line, either as UR or as file name. Find a way to distinguish between URLs and file names automatically and process the input appropriately.

Extra task 2: On the homepage, there is a file called `A00-pos` written in the original BNC format. The only difference to the other file is that multiple word-pos pairs can occur in one line now. Expand your program such that it can process this format, too.

2 Gold-Bug

„The Gold-Bug" is actually a short story by E. A. Poe. It contains an episode in which somebody decrypts an encoded text using character frequencies. The underlying assumption is that the text was encrypted by replacing every character by another one (e.g. every a with a b). For the decryption, one looks at a lot of arbitrary other (non-encrypted) text and counts the frequency of every single character. Then one computes the frequency distribution for the encrypted text and replaces the most frequent character with the most frequent one of the reference text corpus, the second to the most frequent one with the second to the most frequent one in the reference, and so on.

Implement an algorithm that uses this way of decryption. To test your program, we encoded a document (brown-sample-enc.txt) that you can get from the homepage. Punctuation is unchanged. To determine the character frequencies for a reference, use the corpus brown.txt. Analyze the reference corpus first, store the character frequencies and try to read and decode the encrypted file afterwards. Write the decoded text to a new file.

3 Lists and Exceptions

On the lecture slides, we defined a class named `MyIndexError` that shall deliver detailed information in case an index out of the list's range is referenced.

Implement a class `MyList` that inherits from Python's built-in `list` and raises a `MyIndexError` instead of an `IndexError`. In order to do this, you need to overwrite the method `__getitem__(self,index)`; this method is called if a list element is accessed by using `liste[i]` . To make sure that your list objects can be used anywhere where standard lists can be used, you need to call `__getitem__` of `list` whenever a valid index is accessed. In case an error happens here, the user shall see a `MyIndexError`.
