Introduction - Problem Set 6 | Problem Set 6 | Contenu du cours 6.00.1x

Getting Started

This problem set has two parts. The encryption part is graded and deals with encryption, a very important concept in computer science. The recursion part is an ungraded set of problems designed to help you practice writing recursive functions. **We will not provide graders for this recursion part but urge you to practice coding and testing these problems on your own machine.**

Download and save code_ProblemSet6.zip. This zip archive includes the following files:

- `ps6_encryption.py`:

  Skeleton code you'll fill in for the encryption part the problem set.

- `words.txt`:

  A list of English words

- `ps6_pseudo.txt`:

  Pseudocode for Problem 2. We urge you to **not** look at this file until you reach Problem 2 and read the instructions contained there.

- `story.txt`:

  An encoded story

- `ps6_recursion.py`:

  Skeleton code for the practice recursion problems.

Load `ps6_encryption.py` into a Python environment without making any modifications to it, in order to ensure that everything is set up correctly. The code that we have given you loads a list of words from a file. If everything is okay, after a small delay, you should see the following printed out:

```
Loading word list from file...
55909 words loaded.
```

The line assert applyShift(s, bestShift) == 'Hello, world!' will also print out an assertion error, and that is ok, because you haven't implemented the functions yet. If you see an `IOError` instead (e.g., `No such file or directory`), you should change the value of the `WORDLIST_FILENAME` constant (defined near the top of the file) to the complete pathname for the file `words.txt` (this will vary based on where you saved the file).

The file `ps6_encryption.py` has a few functions already implemented that you can use while writing up your solution. You can ignore the code between the following comments, though you should read and understand everything else:

```
# -----------------------------------
```

```
# Helper code
# (you don't need to understand this helper code)
. . .
# (end of helper code)
# ----------------------------------
```