MTH 3300
Spring 2019
E. Fink
April 2, 2019

Homework 8

Directions: Download the template files I have provided on Blackboard. Then open Spyder, load these template files, and write the following programs. Submit your source code to me via Blackboard, in `.py` format; do NOT send any other files. READ THE INSTRUCTIONS on how to submit your work in the Course Documents section of Blackboard.

Be sure to read the SPECIFICATIONS carefully for all problems! And write comments!

---

1) twins.py

Prime numbers (with the exception of 2) are always odd; so, aside from 2 and 3, the smallest possible difference between consecutive prime numbers is 2. For instance,

5 and 7 are primes, and 7 - 5 = 2;

11 and 13 are primes, and 13 - 11 = 2;

17 and 19 are primes, and 19 - 17 = 2;

etc. Two primes that differ by 2 are called *twin primes*. It has been known since antiquity that there are infinitely many prime numbers; however, it is still unproven that there are infinitely many twin prime pairs. (That is, it is unproven that the list that starts with (3,5), (5,7), (11,13), (17,19), (29,31), (41,43), ... *never ends.*) Interestingly, huge progress was made four years ago by an until-then-unfamous mathematician named Yiteng Zheng.

Your first mission is to create a function called `is_prime`. This function should take an integer as input, and return `true` if it has exactly two factors, and `false` otherwise. In my `twins.py` file, I have kindly put some test code, so that you can see if your function is working properly – none of the lines should say `ERROR!`.

Once your function is working, delete my code. Your next mission is to create a program that asks the user to provide two positive integers P and Q. Your program should print all the twin prime pairs that lie between P and Q, inclusive. So, for example, if the user enters 11 and 43, the program should output `(11,13), (17,19), (29,31), (41,43)`. If the user enters 11 and 41, the program should just output `(11,13), (17,19), (29,31)`, since 43 isn't between 11 and 41.

The simplest way to do this is, roughly, go through the integers from P to Q; for each number, determine whether that number *and* that number plus 2 are both prime. Of course, it is up to you to get all the details right.

Specifications: your program must

- write a function called `is_prime`, which takes in an integer, and returns the value `True` if the integer is prime, and `False` if the integer is not prime. (It only needs to work for positive integers; note that 1 is technically not prime!)
- ask the user to enter in two positive integers, P and Q.
- print out all pairs of twin primes where both the primes are between P and Q inclusive.

Challenge (optional)*: read this article* `https://www.nature.com/news/peculiar-pattern-found-in-random-prime-numbers-1.19550`*. The second section,* Not so random*, describes a surprising recent observation about the last digits of consecutive primes. Write a program that will confirm these observations – only look at the primes less than 10,000,000.*

---

2) spoonerism.py

A *spoonerism* occurs when you swap the first letters of two adjacent words in a sentence, and the result still ends up consisting of proper English words. An example: if you intended to say "I ate some jelly beans today", but actually say "I ate some belly jeans today" – the "b" and "j" get swapped, but the resulting words "belly" and "jeans" are still both proper English words. (Actually, there is usually a bit more liberty in what constitutes a spoonerism, but we'll use this definition for the main problem.

You are to write a program that accepts a file containing text as input. *You may assume that the text file contains only lower case letters, and no punctuation.* The program should read through the text, and try to find all pairs of adjacent words that can be "spoonerized" – that is, two words in a row that can have their first letters swapped to create two new valid English words. For every pair of words that can be spoonerized, the program should print out the new words.

For example, if you input a text file containing the words

```
my house mates ate all the jelly beans and now i am hungry
```

then the program should output

```
mouse hates
belly jeans
```

(Notice that the program should not print out `ate all`, since when you swap the first letters, you get the same words you started with!)

To determine what constitues a valid English word, use the file `words.txt`, which contains a fairly exhaustive list of English words.

Write at least one function. Identify a "subproblem" that has clear-cut input and a clear-cut output, and implement a Python function that solves this subproblem.

Specifications: your program must

- ask the user to enter the name of a file containing text; assume the file contains only lowercase letters.
- read through the text file, and locate all pairs of adjacent words where we could swap the first letters to create two new valid English words; pairs should only be printed if the first letters are different.
- print out those two words, with their first letters swapped.
- write at least one function to assist you in your task.

***Challenge**: add two improvements to your program. First, have it work so that it can still identify spoonerizable pairs even if the words happen to contain capital letters, or if they contain parentheses, quotes, or other punctuation. Second, have it be able to identify pairs of words like, for example, `crushing blow` and `blushing crow`: in these examples, you can swap all the letters in each word that come before the first vowel (`cr` is `crushing` and `bl` in `blow`).*