Assignment #3

Math 2604: Mathematical Programming in Python

Instructions: Choose 3 of the following problems to work on. Submit your responses as Python text files, with the extension `.py`. Each file should include your name and the problem number.

Several of these problems require you to get a copy of the file *five_letters.txt* from this web page, and create a list of words from it as follows:

```
words = []
input = open ( 'five_letters.txt', 'r' )
for line in input:
  words.append ( line.strip() )
```

- *Problem 3.0:* Issue Python commands which create a word list from *five_letters.txt*, and determine answers to the following questions:
  - How many words are in the list?
  - Does the list contain the word *osier*?
  - What is the 100th word in the list? (I mean the word of index 99)

  Please use Python commands to answer these questions.

- *Problem 3.1:* Issue Python commands which create a word list from *five_letters.txt*. Now print every word which contains the word *rat (in exactly that order)*.

- *Problem 3.2:* If A is an $m \times n$ matrix and v is a vector of length n, then the product $u = A * x$ is defined mathematically as the vector of length m such that

$$u_i = \sum_j A_{i,j} v_j$$

  Let $n = 5$, let A be the magic matrix that we discussed in class, and let v be the vector of length 5 whose entries are all 1. Use Python commands to compute the product $u = A * v$ and print it. Can you explain your result?

- *Problem 3.3:* Let $n = 7$, let A be the magic matrix. Use a Python `for()` loop to compute the sum of the diagonal elements of A, that is, $A[0,0] + A[1,1] + ... + A[6,6]$. Then also compute the sum of the *antidiagonal* elements $A[0.6], A[1,5], ..., A[6,0]$.

- *Problem 3.4:* Issue Python commands which create a word list from *five_letters.txt*. Now print every word in the list with the property that it is also a word when written backwards. For example, "straw" is such a word, since "warts" is also on the list. Note that in Python, the reversal of a `word` can be expressed as `word[::-1]`.

- *Problem 3.5:* If we sample n random points (x, y) in the unit square, then roughly $\frac{\pi}{4}$ of them will be inside the unit circle. If the number of such points is k, then we have $\frac{k}{n} \approx \frac{\pi}{4}$, or $\pi \approx 4\frac{k}{n}$. Use the statement `from random import random`, and use a `for()` loop to choose n pairs of random values x and y. Note the number of times that (x, y) is in the unit circle, and print the corresponding estimate of π. (Problem P2.5.10 from Hill.)

- *Problem 3.6:* DNA can be represented as a string of the letters 'A', 'C', 'G' and 'T'. A codon is a sequence of three such letters. DNA is processed or "read" by dividing it into codons formed by successive triples. However, the processing can begin at the first, second, or third letter; that is, the

processing might ignore zero, one or two initial letters. This initial skip is called the "frame". Processing continues until the end of the string is reached, or not enough letters remain to form another codon.

Let the DNA string be 'AGTCTTATATCT'. If read from frame 0, you will get 4 codons, otherwise only 3. Write a Python function which accepts as input the DNA string, and the value of `frame` (0, 1, or 2). Your function returns a list containing the codons that were found. Apply your function to the given DNA string, and consider each possible value of `frame`. (Problem P2.4.5 from Hill.)

- *Problem 3.7:* Issue Python commands which create a word list from *five_letters.txt*. We want to find all anagrams of the word `traps`. This could be a hard problem. However, notice that two words are anagrams if and only if we get identical results after alphabetizing their letters. The alphabetized version of a word is created using the `sorted()` function. Thus, `sorted ( 'traps' ) = 'aprst'` . Compare the sorted version of our target word against the sorted version of each word in the word list, and report the matches.

- *Problem 3.8:* For two vectors a and b of length 3, the dot product is

$$a \cdot b = \sum_i = t^n a_i b_i$$

and the vector cross product is

$$a \times b = \begin{bmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \end{bmatrix}$$

(In these mathematical definitions, our indices are starting at 1.) Assuming that $a = [1, 2, 3]$ and $b = [4, 5, 6]$ are stored as Python lists, compute the dot and cross products. (Problem P2.7.3 from Hill.)

- *Problem 3.9:* Given the list of numbers $a = [1, 2, 3]$, write Python statements that compute the list p with the property that $p[i]$ is the product of all the elements of a except for the i-th element. (Problem P2.4.1 from Hill.)