Problem Set #2

MACS 30000, Dr. Evans
Due Monday, Nov. 21 at 11:30am

1. **Numpy slicing operations (3 points).**

   (a) Write a function `firsthalf()` that accepts an input string of two or more characters and returns the first half of it, excluding the middle characters if there is an odd number of characters. (Hint: the built-in function `len()` returns the length of the input string.)

   (b) Write a function `backward()` that accepts an input string and reverses the order of its characters using slicing (not a `for` loop) and returns the reversed string. (Hint: the `step` parameter used in slicing can be negative.)

   (c) Write a function called `pig_latin()` that accepts a string that is a single word, translates it into Pig Latin, then returns the translated word. Specifically, if the string starts with a vowel, add "hay" to the end. If the string starts with a consonant or two consonants or three consonants, then take the first set of consonants, move them to the end of the word, and add "ay". (Hint: use the `in` operator to check if the first letter is a vowel.)

   (d) (This problem comes from https://projecteuler.net.) In the 20×20 grid below, four numbers along a diagonal line have been marked in red.

```
08 02 22 97 38 15 00 40 00 75 04 05 07 78 52 12 50 77 91 08
49 49 99 40 17 81 18 57 60 87 17 40 98 43 69 48 04 56 62 00
81 49 31 73 55 79 14 29 93 71 40 67 53 88 30 03 49 13 36 65
52 70 95 23 04 60 11 42 69 24 68 56 01 32 56 71 37 02 36 91
22 31 16 71 51 67 63 89 41 92 36 54 22 40 40 28 66 33 13 80
24 47 32 60 99 03 45 02 44 75 33 53 78 36 84 20 35 17 12 50
32 98 81 28 64 23 67 10 26 38 40 67 59 54 70 66 18 38 64 70
67 26 20 68 02 62 12 20 95 63 94 39 63 08 40 91 66 49 94 21
24 55 58 05 66 73 99 26 97 17 78 78 96 83 14 88 34 89 63 72
21 36 23 09 75 00 76 44 20 45 35 14 00 61 33 97 34 31 33 95
78 17 53 28 22 75 31 67 15 94 03 80 04 62 16 14 09 53 56 92
16 39 05 42 96 35 31 47 55 58 88 24 00 17 54 24 36 29 85 57
86 56 00 48 35 71 89 07 05 44 44 37 44 60 21 58 51 54 17 58
19 80 81 68 05 94 47 69 28 73 92 13 86 52 17 77 04 89 55 40
04 52 08 83 97 35 99 16 07 97 57 32 16 26 26 79 33 27 98 66
88 36 68 87 57 62 20 72 03 46 33 67 46 55 12 32 63 93 53 69
04 42 16 73 38 25 39 11 24 94 72 18 08 46 29 32 40 62 76 36
20 69 36 41 72 30 23 88 34 62 99 69 82 67 59 85 74 04 36 16
20 73 35 29 78 31 90 01 74 31 49 71 48 86 81 16 23 57 05 54
01 70 54 71 83 51 54 69 16 92 33 48 61 43 52 01 89 19 67 48
```

   The product of these numbers is $26 \times 63 \times 78 \times 14 = 1788696$. Write a function that returns the greatest product of four adjacent numbers in the same direction (up, down, left, right, or diagonally) in the grid. Use array slicing. DO NOT use a `for` loop or multiple `for` loops. For convenience, this array has been saved in the file `grid.npy`. Use the following syntax to extract the array:

```
>>> grid = np.load('grid.npy')
```

2. **Functions and modules (3 points).** In this exercise, you will write two python `.py` files. The first file will be a script named `ex2script.py`. The second will be a module named `ex2module.py`.

   (a) In the script `ex2script.py` declare parameters of a lognormally distributed random variable *scores*. Let the mean of the logarithm of that variable be $\mu = \log(70)$ and let the standard deviation of the logarithm of that variable be $\sigma = 0.2$. Randomly generate 10,000 draws from this lognormal distribution. [You can do this using the `numpy.random.normal()` function and then exponentiating the output, or you can use the `numpy.random.lognormal()` function.] Use the `seed` command to make everybody's random draws the same. Import the module `ex2module.py`, and use the function `threescrubs()` from that module to remove outliers in a certain way.

```
import numpy as np
import ex2module as ex2m

# Declare your parameters
...
seed = 300
np.random.seed(seed)
# Then add your code generating 10,000 draws here.
...
# Then run the scores through the function that performs
# three waves of eliminating outliers that are 3 standard
# deviations away from the mean.
scrubbed_scores = ex2m.threescrubs(scores)
```

   (b) In the module `ex2module.py`, write two functions. Call one function `kill_outliers()`. This function takes as an input a one-dimensional NumPy array, deletes all elements of the array that are greater than 3 standard deviations above or below the mean of the series the mean, and returns the one-dimensional NumPy array with the remaining elements (all of which should be less than or equal to three standard deviations away from the mean of the originally input series. Call the other function `threescrubs()`. This function takes as an input a one-dimensional numpy array. It then runs three consecutive filters of `kill_outliers()`. Each time you run the array through `kill_outliers()`, the returned array should get shorter.

   (c) Report the number of elements in the array returned by the function `threescrubs()`, the mean of that vector, and the standard deviation of that vector.

   (d) Make sure your functions in the module `ex2module.py` are commented as in the instructions in the `PythonFuncs.ipynb` notebook.

3. **Reshaping data, groupby (4 points).**

(a) Define the following matrices as NumPy arrays.

$$A = \begin{bmatrix} 0 & 2 & 4 \\ 1 & 3 & 5 \end{bmatrix} \quad B = \begin{bmatrix} 3 & 0 & 0 \\ 3 & 3 & 0 \\ 3 & 3 & 3 \end{bmatrix} \quad C = \begin{bmatrix} -2 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & -2 \end{bmatrix}$$

Write a function `ipmstack(A, B, C)` that uses NumPy's stacking functions to create and return the following block matrix D:

$$D = \begin{bmatrix} \mathbf{0} & A^T & \boldsymbol{I} \\ A & \mathbf{0} & \mathbf{0} \\ B & \mathbf{0} & C \end{bmatrix}$$

where $\boldsymbol{I}$ is the identity matrix of the appropriate size, and each $\mathbf{0}$ is a matrix of all zeros also of appropriate sizes. A block matrix of this form is used in the interior point method for linear optimization.

(b) Read in the comma-separated file `popagesex2015.csv` as a pandas DataFrame. This file contains total U.S. population levels by gender and age for the years 2010 through 2015. Note that this file has some rows at the top that must be ignored. It also has three rows in the interior of the file that should be ignored. These are the three rows with `age=999`, and are total population for each `age` for a given `gender` value. The values in the `gender` column are $\{0, 1, 2\}$, where `gender=0` is the sum of male and female for a given `age`, `gender=1` is male, and `gender=2` is female.

i. Read in the `popagesex2015.csv` file as a pandas DataFrame. In the one line of the read-in command, ignore the appropriate lines at the top of the data as well as the three lines in the interior of the data, make both `gender` and `age` index columns of the DataFrame, relabel the columns as `['gender', 'age', 'pop10', 'pop11', 'pop12', 'pop13', 'pop14', 'pop15']`, and relabel the values of the `gender` variable to `0=both`, `1=male`, `2=female`.

ii. Use pandas `groupby` command in one line to create a 2×6 DataFrame that gives the percent of the total population in a given year (the 6 columns) that is a given gender (the two rows). Note that you are ignoring `gender=0`.

iii. Use pandas `groupby` command in one line to create a 101-element Series object that gives the average percent of the total population (both men and women) that is of a given age. The average is over the six years.