1. **Array Creation and Operations**

   (a) Create the following array $\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 & 1 \\ 1 & 1 & 3 & 1 & 1 \\ 1 & 1 & 1 & 4 & 1 \end{bmatrix}$

   (b) Compute the row sums of the above matrix

   (c) Compute the column sums of the above matrix

   (d) Download and read into memory the matrix found below. Check that it is equal to the array you created above.

   http://stanford.edu/~danfrank/cme193/data/lec4_array.txt

2. **Array Slicing and Indexing**

   Using the array above return the second and third rows and the columns containing an even number as a 2×2 array using...

   (a) integer indexes

   (b) slices

   (c) boolean arrays

   (d) boolean arrays computed from the array

3. **Copys and Views** NumPy arrays are objects and follow the same assignment and copy rules as ordinary python objects. However, when we slice an array python returns a *view* on that same data, meaning that a new object is created but shares the same underlying data. Integer indexing and boolean indexing do not create views.

   (a)
   ```
   >>> A = np.ones((2, 5))
   >>> B = A
   >>> A[0,0] = 42.
   >>> A[0, 0] == B[0, 0].
   ```

   ```
   >>> A = np.ones((2, 5))
   >>> B = A[:, 1:3]
   >>> B[0,0] = 42.
   >>> A[0, 1] == 1.
   ```

   (b)
   ```
   >>> A = np.ones((2, 5))
   >>> B = A[:, np.array([False, True, True, False, False])]
   >>> B[0, 0] = 42
   >>> A[0, 1] == 1
   ```

   (c)
   ```
   >>> A = np.ones((2, 5))
   >>> B = A[:, np.array([1, 3])]
   >>> B[0, 0] = 42
   >>> A[0, 1] == 1
   ```

   (d)
   ```
   >>> A = np.ones((2, 5))
   >>> B = A[:, 1:3]
   >>> A += 1
   >>> np.all(B == 2)
   ```

4. **Broadcasting**

   Using the above array assigned as *arr*, describe the following operations

   (a) `arr * 5.`

(b) `arr * np.arange(arr.shape[1])`

(c) `arr * np.arange(arr.shape[0])`

(d) `arr.T * np.arange(arr.shape[0])`

(e) compute the dot product of the array with $\begin{bmatrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$ in two ways