Assignment 6

Arrays

Date Due: Friday November 1, 2019, 6:00pm **Total Marks: 20**

General Instructions

- **This assignment is individual work.** You may discuss questions and problems with anyone, but the work you hand in for this assignment must be your own work.
- **Assignments are being checked for plagiarism.** We are using state-of-the-art software to compare every pair of student submissions.
- **Read every question carefully.** All of it. These questions cannot be described in a few sentences, and except for a little bit of fun context, all parts of the question have important information crucial to your success.
- Each question indicates what to hand in. You must give your document the name we prescribe for each question, usually in the form aNqM, meaning Assignment N, Question M. Put name and student number appear at the top of every document you hand in. These conventions assist the markers in their work. Failure to follow these conventions will result in needless effort by the markers, and a deduction of grades for you. Do not submit folders, zip documents, even if you think it will help.
- **Programs must be written in Python 3.** Marks will be deducted with severity if you submit code for Python 2.
- **Assignments must be submitted to Moodle.** There is a link on the course webpage that shows you how to do this.
- **Moodle will not let you submit work after the assignment deadline.** It is advisable to hand in each answer that you are happy with as you go. You can always revise and resubmit as many times as you like before the deadline; only your most recent submission will be graded.
- Questions are annotated use descriptors like "easy" "moderate" and "tricky". All students should be able to obtain perfect grades on "easy" problems. Most students should obtain perfect grades on "moderate" problems. The problems marked "tricky" may require significantly more time, and only the top students should expect to get perfect grades on these. We use these annotations to help students be aware of the differences, and also to help students allocate their time wisely. Partial credit will be given as appropriate, so hand in all attempts.

Question 1 (20 points):

Purpose: To practice arithmetic and indexing with arrays.

Degree of Difficulty: Easy to Moderate.

In this question, you will again use information from the City of Saskatoon Open Data Catalogue (http://opendata-saskatoon.cloudapp.net/). The file Neighbourhood Dwelling Unit Count has been retrieved and saved as the file `q6.txt`. It contains information on the numbers of various types of dwellings in the various neighbourhoods in Saskatoon(as of 2016). Each dwelling is classified as either single family, two unit or multi unit. The overall goal of the assignment is to determine which are the dense neighbourhoods with fewer that 40% single family dwellings. The input file is a tabular file with commas for delimiters (if you want a better view of what the data in the file looks like, open it in Excel, OpenOffice, or similar spreadsheet program; this will nicely visualize the data in columns for you).

To complete the assignment, do the following parts. Make sure each part is correct before moving on to the next one. **The final three parts must be done with array slicing and array operations.**

(a) Begin by opening and reading the file `a6.txt` into Python. Next, the first line in the file is discarded as it contains only column labels. The first line can be discarded by invoking the `readline()` method for file objects. Next, use the file to produce a Python list , referred to with variable `dataList`, whose items are lists containing the contents of each line. Each line is a list of strings. Print a slice of `dataList` containing the first two items.

The contents of this slice should be should be

```
[['027f2eb8-dc58-4057-b120-afcfca01653a', 'Parkridge',
'4', '1136', '124', '329', '1589', '166', '409', '10', '4'],
['07767a30-94e5-4f98-8904-4a0334cc7896', 'Aspen Ridge',
'', '', '', '', '', '', '', '', '']]
```

In each line we are interested in the second item, which is the neighbourhood, the fourth item, which is the number of single family dwellings, the fifth item, which is the number of two unit dwellings, the sixth item which is the number of multiunit dwellings and the seventh item, which is the total number of dwellings in the neighbourhood

(b) As seen above, the second item in `dataList` is a list that contains a bunch of empty strings. All those empty strings indicate that, as of 2016, there was no known dwellings in the neighbourhood "Aspen Ridge". In order to do our analysis, we need to change the empty strings to zeros for the items we use. In this part clean `dataList` by converting empty strings to zeros for the items whose index is 3,4,5, or 6 (The fourth through seventh items), in every row. After doing this the second item in `dataList` should look like

`['07767a30-94e5-4f98-8904-4a0334cc7896', 'Aspen Ridge', '', 0, 0, 0, 0, '', '', '', '']`

Print the second item in `dataList` to the console.

(c) Create a one dimensional numpy array `neighbourhood` that contains the names of the neighbourhoods. Also create a 2D numpy array `dwelling` where each row contains information about a single neighbourhood, and the first column is the number of single family dwellings, the second column is the number of two unit dwellings, the third column which is the number of multiunit dwellings and the fourth column is the total number of dwellings in the neighbourhood. These numbers should be integers. The first row of dwelling should look like `[ 1136 124 329 1589]` **For full marks use list comprehesions rather than loops to do this.** Print the first row of dwelling.

(d) Print out the shape attribute of both `neighbourhood` and `dwelling`

```
(77,)
(77, 4)
```

(e) Some of the rows, or some of the columns ,can be selected from a 2D array by logically indexing with a 1D array. For example the following code

```
import numpy as np
a = np.array([[1,2,3],[4,5,6],[7,8,9]])
print(a)
z = np.array([True,False,True])
print(z)
c = a[z,:]
print(c)
d = a[:,z]
print(d)
```

produces the following output

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
[ True False  True]
[[1 2 3]
 [7 8 9]]
[[1 3]
 [4 6]
 [7 9]]
```

Remove the rows in in both `neighbourhood` and `dwelling` for neighbourhoods in which there are no dwellings by doing the following. First create a 1d Boolean array `nonzero` where an item is TRUE if the corresponding neighbourhood has some dwellings. **Do this by indexing** every row and the fourth column of `dwelling` and comparing the value to zero. Next use `nonzero` to **logically index** both `neighbourhood` and `dwelling`, and then remove the empty neighbourhoods. Again print out the shape attribute of both `neighbourhood` and `dwelling`.

```
(74,)
(74, 4)
```

(f) Create an array `notsingle` which contains, for each remaining neighbourhood, the number of dwellings that are either two unit or multiunit. **Do this without using a loop, by indexing dwelling.** The first two items in `notsingle` should be 453 and 150. Print a slice of `notsingle` which contains the first two items.

Use array arithmetic to create an array `dense_ratio` which for each neighbourhood remaining contains the ratio of the dwellings which contains more than one unit to the total number of dwellings. The first two numbers in `dense_ratio` should be approximately 0.28508496 and 0.11169025 Print a slice of `dense_ratio` which contains the first two items.

(g) Create an array `dense_boolean` which is a boolean array which is True for a neighbourhood if the `dense_ratio` value for that neighbourhood is greater than 0.6. **Do this using array operations and without using a loop**. **Use array logical indexing** to print out the names of the dense neighbourhoods, ie those whose `dense_ratio` are greater than 0.6.

```
['Airport Business Area' 'Nutana SC' 'Lakewood SC' 'Blairmore SC'
 'U of S Lands South MA' 'West Industrial' 'Wildwood' 'Pleasant Hill'
 'Central Business District' 'Nutana' 'Kensington' 'Confederation SC'
 'Lawson Heights SC' 'University Heights SC' 'City Park'
 'Central Industrial']
```

What to Hand In

(a) A document entitled `a6q1.py` containing your finished program, as described above.

Evaluation

- 1 mark for correctly reading the file into `dataList`
- 1 mark for cleaning `dataList` by changing the empty strings to zeros in the correct columns.
- 1 mark for creating array `neighbourhood`
- 2 marks for creating array `dwelling`
- 1 mark for printing out the correct shapes of `neighbourhood` and `dwelling`
- 2 marks for creating boolean array `nonzero`
- 1 marks for removing neighbourhoods with no dwellings from array `neighbourhood`
- 2 marks for removing neighbourhood data from array `dwellings` for neighbourhoods with no dwellings.
- 1 mark for printing the shapes of `neighbourhood` and `dwelling`
- 2 marks for creating array `notsingle`
- 2 marks for creating the array `dense_ratio`
- 2 marks for creating the array `dense_boolean`
- 2 marks for printing out the correct dense neighbourhoods