CSE 163 **Name:** _____
Spring 2019
Exam 2 - Practice
6/03/2019 **Student Number:** _____
Time Limit: 50 Minutes

Do not open the exam before the exam begins and close the booklet when time is called. Starting early or working after time is called will lead to a -10 deduction. You may write your name and Net ID on the front of the exam before the exam starts.

This exam contains 8 pages (including this cover page) and 4 questions. Some questions have sub-parts.

You are allowed to have one sheet of paper (both sides) with you as your cheat sheet. All other materials besides writing utensils should be put away before the exam starts. This includes all electronic devices like phones, calculators, and smart watches.

This exam is not, in general, graded on style and you do not need to include comments or imports for your code. Specific questions may specify restrictions about the style of your code that you must follow to receive full credit.

You may not abbreviate any code, such as "ditto" marks or "..." marks. You may write code to the side and indicate where it should be inserted. These markings must be unambiguous and any ambiguity when grading may result in a deduction if your code is not readable. All code and answers should remain within the provided boxes if possible.

You are allowed to ask for scratch paper after the exam starts to use as additional space when writing answers, but you must indicate on the original page for the problem that part of the answer is on scratch paper. Scratch paper must be stapled to the **END** of your exam after you finish the test. Failure to do so may result in your work on scratch paper not being accepted while grading.

Initials: _____

Initial above to indicate you have read and agreed to the rules above.
Failure to initial may result in your exam not being accepted for credit.

Practice Exam Notes

Like before, this practice exam focuses more on covering the breadth of topics you might see on the exam rather than writing a practice that would take students exactly 50 minutes to complete. Some of the response pages may be shorter than they would appear on the actual exam to save space; this means on the real exam, where we will leave plenty of space to write answers, will likely look much longer even though the amount of content is about the same.

1. This question will be implementing part of the `matplotlib` library. We will be doing a very simplified version of it that won't involve any plotting at all, but instead focuses on some of the classes and methods we have used this quarter. This problem involves writing two short classes and one method to create them. All classes should have private fields that are not accessed outside the class.

   (a) Write a class called `Axis` to represent a single subplot in a figure. The class will keep track of the title of the subplot should have the following methods:

   - An initializer that sets the initial title to `No Title`.
   - A method named `set_title` that sets the title to a given title.
   - A method named `get_title` that returns the current title.

Solution:

```
class Axis:
    def __init__(self):
        self._title = 'No Title'

    def set_title(self, title):
        self._title = title

    def get_title(self):
        return self._title
```

   (b) Write a class called `Figure` that represents a `matplotlib` figure. The class should keep track of all of the axes in it and should have the following methods:

   - An initializer that takes a 2-d `numpy` array of `Axis` objects.
   - A function named `show` that prints out all the axis titles in the subplots. Axis titles that belong to the same row should be printed on the same line, separated by the character `|`, and different rows should be printed on different lines.

Solution:

```
class Figure
    def __init__(self, axes):
        self.axes = axes

    def show(self):
        for i in range(self.axes.shape[0]):
            for j in range(self.axes.shape[1]):
                print(self.axes[i, j].get_title(), end='|')
            print()
```

(c) Write a function called `subplots` that takes parameters `nrows` and `ncols` and produces a figure with the subplots that fit into the specified number of rows and columns. The function should construct an `Axis` objects for each subplot and a `Figure` object to store all of the axes. The function should return a tuple where the first element is the `Figure` and the second is a `numpy` array of all the axes.

To create an empty array to start, you will need to use the `full` function in `numpy`. For example, to make an array of `None`s with shape `(2,3)`, you would write `np.full((2,3), None)`.

Solution:

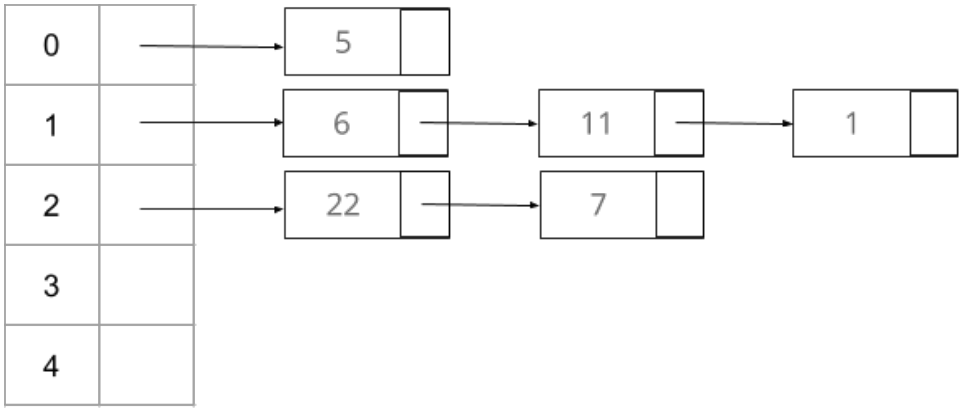
```
def subplots(nrows, ncols):
    axes = np.full((nrows, ncols), None)
    for i in range(nrows):
        for j in range(ncols):
            axes[i, j] = Axis()
    fig = Figure(axes)
    return fig, axes
```

2. This question has a few short answer questions on miscellaneous topics from the quarter.

(a) This question concerns hashing and what the result hash table will be if we inserted the following values in the order given. Suppose we had a hash table of numbers of length 5 and were using a hash function that is just the value of the number mod 5. When resolving collisions, we make a chain of values where the newest values appears at the front of the chain. Make sure to label the indices of the hash table.

Values: `1, 5, 7, 11, 22, 6`

Solution:



(b) In two or three sentences, explain why Uber's "Rides of Glory" blog post would be considered unethical while Google using location data to improve Google Maps seems more permissible.

Solution: Valid answers include talking about respecting the user or describing user's consent to how their data is used. A really good answer would compare how Uber's use case violates either of these while Google's does not. Here is an example of a complete answer:

> The difference between these cases lies in the consent the user gives to using their data. In Google's case of improving the app, people wouldn't think their privacy was being violated since they are already agreeing to send the data to Google to use the service and it seems reasonable that Google can use that data to improve said service. Uber's case is different since it doesn't seem reasonable the user would necessarily consent to their data contributing to that blog post that might seem disrespectful of users to some.

(c) In two or three sentences, explain how `geopandas`'s `dissolve` function is similar to `pandas`'s `groupby` and also how they differ.

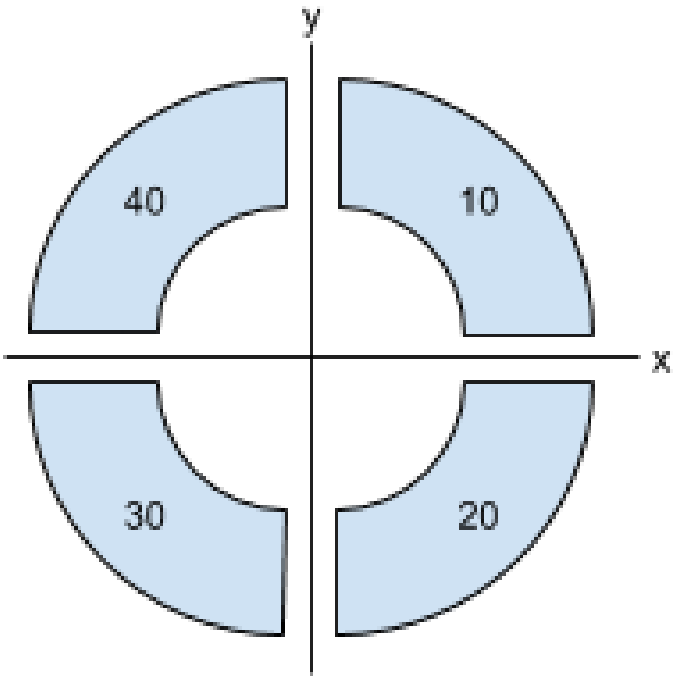
Solution: Here is an example of a complete answer:

> `dissolve` behaves exactly the same as a `groupby` except for the column of the data relating to the geometry of the `GeoDataFrame`. For the non-geometry columns, it is equivalent to `groupby`, but for the geometry column it combines all of the geometries that fall into that group to be one big geometry that is the union of all of them.

3. For this problem, assume we have the following geo-spatial dataset stored in a `GeoDataFrame` named `df`:

group	value	geometry
A	10	Polygon(...)
B	40	Polygon(...)
A	20	Polygon(...)
B	30	Polygon(...)

When the `GeoDataFrame` is plotted using the column `value`, we see the plot below. Instead of color, we just write the value inside the shape.



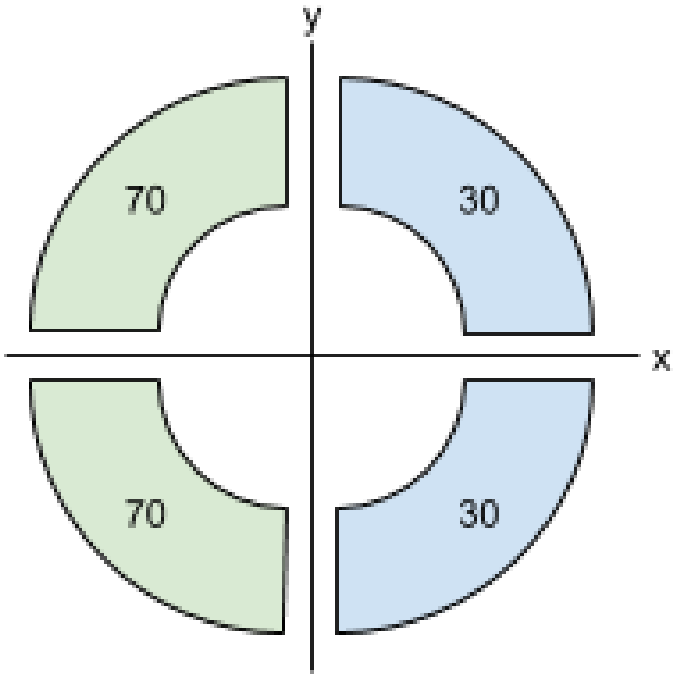
(a) In the space below, write the code that makes a plot of the data above that has been aggregated by the column `group`. Data that belongs to the same group should be plotted with the sum of the values in that group .

Solution:

```
grouped = df.dissolve(by='group', aggfunc='sum')
grouped.plot(column='value')
```

(b) On the next page, draw the plot that would be created from the code you wrote for the previous problem.

Solution:



4. The following problems relate to image processing.

   (a) Given the following image

1	2	1	3
4	3	3	3
5	5	4	4
1	1	2	4
3	2	1	5

   In the space below, write out the result of doing a convolution with the following kernel on this image

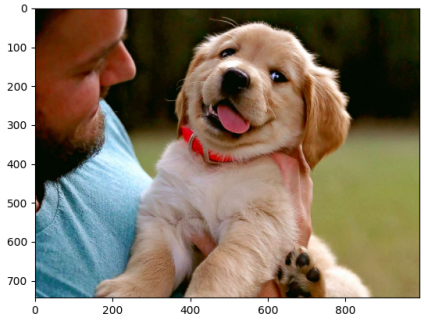
0	1	2
0	0	2
0	0	1

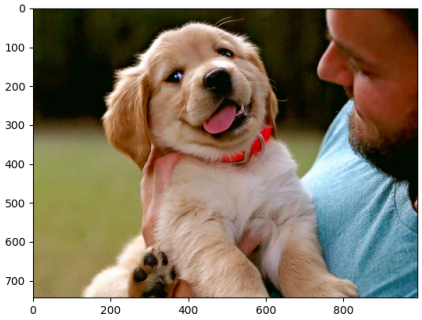
Solution:

14	17
19	21
18	25

(b) Write a function called `flip_horizontal` that takes a color image (a `(n, m, 3)` `numpy` array) and returns a new color image that is the result of flipping the image on its horizontal axis. For example, if we had a `numpy` array named `img`

`img` `flip_horizontal(img)`





Note, it is important that you explicitly create a new array by copying the image since indexing into the original array will return an array that references the data inside the original image. Your solution should have no loops for full credit.

Solution:

```
def flip_horizontal(img):
    result = img.copy()
    return result[:, ::-1, :]
```

(c) Write a function called `convolution` that takes an gray-scale image (`numpy` array with shape `(n, m)`) and a square kernel (`numpy` array with shape `(k, k)`) and performs a convolution on the image using the given kernel. Your solution should have no loops besides the two needed to move the sliding window.

Solution:

```
def convolution(img, kernel):
    height, width = img.shape
    k_width = kernel.shape[0]
    res = np.zeros((height - k_width + 1, width - k_width + 1))

    for i in range(res.shape[0]):
        for j in range(res.shape[1]):
            curr = image[i:i+k_width, j:j+k_width]
            val = np.sum(curr * kernel)
            res[i, j] = val
    return res
```

- **Built-in Python functions**
  - `print(*strings)`
  - `range(end)` / `range(start, end, step?)`
  - `abs(v)`
  - `min(v1, v2)` / `max(v1, v2)`
  - `sum(v1, v2)`
  - `open(fname)`
  - `zip(l1, l2)`
  - Types: `int(v)`,`float(v)`,`str(v)`,`bool(v)`
- **String methods**
  - `upper()`, `lower()`
  - `find(s)`
  - `strip()`
  - `split()`
- **List methods**
  - Construct: `list()` or `[]`
  - `append(val)`
  - `extend(lst)`
  - `insert(idx, val)`
  - `remove(val)`
  - `pop(idx)`
  - `index(val)`
  - `reverse()`
  - `sort(key?)`
- **Set methods**
  - Construct: `set()`
  - `add(val)`
  - `remove(val)`
- **Dictionary methods**
  - Construct: `dict()` or `{}`
  - `keys()`
  - `values()`
  - `items()`
- **Special object methods**
  - `__init__`
  - `__repr__`
  - `__eq__`
  - `__hash__`
- **Pandas methods**
  - `mean()`
  - `min()` / `max()`
  - `idxmin()` / `idxmax()`
  - `count()`
  - `unique()`
  - `groupby(col)`
  - `apply(fun)`
  - `isnull()` / `notnull()`
  - `dropna()` / `fillna(v)`
  - `sort_values(col)` / `sort_index()`
  - `nlargest(n, col)`
  - `merge(df, left_on, right_on, how)`
- **Pandas fields**
  - `index`
  - `loc[row, col]`
- **Geopandas obnject methods**
  - `plot(column?, legend?, ax?, color?, vmin?, vmax?)`
  - `dissolve(by, aggfunc)`
  - Any of the pandas functions above
- **Geopandas module methods**
  - `geopandas.sjoin(left, right, op, how)`
- **matplotlib model classes**
  - `plt.subplots(nrows, ncols)`
  - `plt.show()`
  - `plt.savefig(f_name)`
- **numpy module methods**
  - `np.array(vals?)`
  - `np.arange(end)` / `np.arange(start, end, step?)`
  - `np.ones(shape)` / `np.zeros(shape)`
  - `np.dot(a1, a2)`
  - `np.sum(a)` / `np.min(a)` / `np.max(a)` / `np.mean(a)`
- **numpy array object methods**
  - `reshape(shape)`
  - `sum()` / `min()` / `max()` / `mean()`
  - `copy()`
- **numpy module methods**
  - `shape`