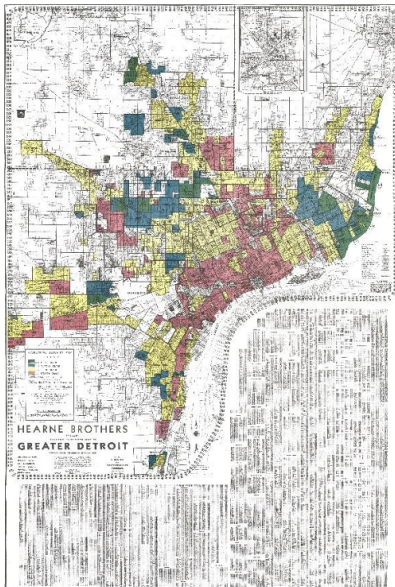


HW Redlining

Using Text Data, JSON & APIs in the wild.

Introduction

In the prior homework, we replicated a simulation of residential segregation that 'explained' segregation through individual preference. However, this model did not account for the active policy interventions that have led to segregation in the United States. One such policy intervention is the use of 'residential security maps' to determine which neighborhoods would be granted government-backed low-interest rate mortgages. These maps were used explicitly to exclude neighborhoods with high African American representation from receiving government investment. Those neighborhoods were marked in red, and the practice has become known as redlining. All non-python readings linked in this HW are optional (i.e. not required).



(<https://www.smartcitiesdive.com/ex/sustainablecitiescollective/short-history-redlining/1162160/>)

In today's homework, we will examine the original redlining map of Detroit as we practice the following technical skills:

- Working with text files
- Parsing new JSON files
- Navigating new APIs
- General Python skills including list comprehensions, dictionaries, etc.

Notice:

1. Do not change the method name.
2. For the cache file, follow the naming requirements in the docString.
3. You should not use any nlp package, such as nltk. Importing an unneeded package might cause the autograder to crash and your grade could be influenced.
4. Test your implementation in the main() function, not the other place.
5. Test your implementation and make sure your project will yield a result before submitting it to gradeScope.

Step 1

Using the Python methods we used in class to load the **redlines_data.json** file which was originally from <https://dsl.richmond.edu/panorama/redlining/static/downloads/geojson/MIDetroit1939.geojson> and understand the structure of it. The JSON object will be used to construct **DetroitDistrict** instance.

In the first line of your .py file include the following, which set the random seed.

```
import random
random.seed(17)
```

Step 2

Background

The JSON is a digitized version of the original dataset used in 1936. The file divides Detroit into **238** districts. Each district has a set of latitude and longitude coordinates, a letter grade (A, B, C, or D), and a text description of the demographics of the neighborhoods. The district grades are associated with the following colors and designations: green for the "Best," blue for "Still Desirable," yellow for "Definitely Declining," and red for "Hazardous." These grades were determined primarily by the racial composition of the districts (<https://www.esri.com/arcgis-blog/products/arcgislivingatlas/announcements/redlining-data-now-in-arcgis-living-atlas/>)

Content warning – the text

descriptions are from 1936 and may contain offensive language and ideas. Reading each of those text descriptions **is NOT mandatory** for this assignment.

Task

Define the DetroitDistrict. Refer to **class DetroitDistrict** in **red_lines_starter.py**.

Step 3

Define **RedLines** Class and its methods in the following steps. Refer to **class RedLines** in **red_lines_starter.py**.

Create 238 objects of the class DetroitDistrict from the redlining data in a list.

Note that these must be created in the order they are present in the redlining data.

Step 4

Define **plotDistricts**.

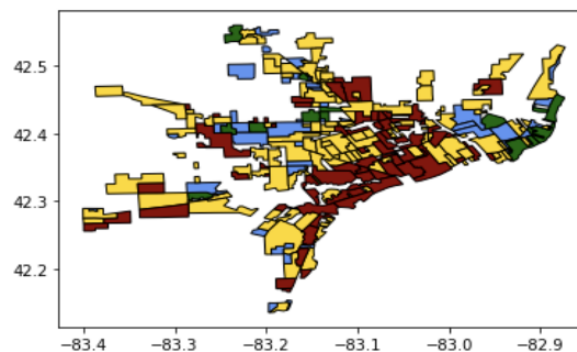
One of the learning objectives of this course is to be able to use documentation to figure out how to use packages you have not been taught. This will be a critical skill in your internships and your year 2 courses. In line with this objective, you will be asked to complete the following code to produce a redlining map of Detroit.

The starter code provided only needs 2 lines to be filled in – see the comments in red. You must use the matplotlib documentation to figure out how to create a polygon for each district and give it the color appropriate to that district. Keep the borders of the district as black. Matplotlib documentation can be found here: <https://matplotlib.org/3.1.1/index.html> Please note that a polygon is a specific method within matplotlib. After creating the graph, you need to name it `redlines_graph.png` and later submit it. How to save the graph, https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.savefig.html

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib

def plotDistricts(self):
    """
    Plots the districts using matplotlib, displaying each
    district's location and color.
    Name it redlines_graph.png and save it to the current
    directory.
    """
    fig, ax = plt.subplots()
    for ____: # what kind of for loop makes sense?
        ax.add_patch(.....) # add arguments here
        ax.autoscale()
    plt.rcParams["figure.figsize"] = (15,15)
    plt.show()
```

Expected Graph



Step 5

Define `generateRandPoint`

The following code worked until very recently to pick a latitude and longitude coordinate from each of the districts. Due to updates in one of the libraries, this will no longer work for most people. Your job is to decipher the error messages and figure out how to update this code for the `generateRandPoint` method.

Notice you should include the import at the first line of your code, not within the method.

Also, pay close attention to the variable name, and you need to change some variables' names to class attributes' names. Or it will not pass the autograder.

```

import random as random
from matplotlib.path import Path
import numpy as np

xgrid = np.arange(-83.5, -82.8, .004)
ygrid = np.arange(42.1, 42.6, .004)
xmesh, ymesh = np.meshgrid(xgrid, ygrid)
points = np.vstack((xmesh.flatten(), ymesh.flatten())).T
for j in Districts:
    p = Path(j.Coordinates[0])
    grid = p.contains_points(points)
    point = points[random.choice(list(np.where(grid)[0]))]
    print(j, " : ", point)
    j.RandomLong = point[0]
    j.RandomLat = point[1]

```

generateRandPoint method: Refer to **class RedLines** in **red_lines_starter.py**.

Comment on the code so you understand it. Search documentation to understand the parts you are not familiar with. This is an essential skill!

Step 6

Define `fetchCensus`.

Now each district has 1 latitude/longitude point that represents it. We want to use that point to find the present-day median household income of that district. To do this we need to identify the census tract code for the random point from each district. This can be obtained using the following

API: <https://geo.fcc.gov/api/census/>

Add the tract code to the appropriate attribute in our class.

NOTE

In Step 7 you will use census data from 2018. What census year should you use for the tract number? You may need to google this if you are unaware.

Important

Please use the area API

The order of the API call parameter has to follow the following.


'lat': xxx, 'lon': xxx, 'censusYear': xxx, 'format': 'json' Or

'lat': xxx, 'lon': xxx, 'censusYear': xxx

Hint

Think about why the `censusTract` attribute needs to be 9 digits for each district instance.

The census tract combines the 2 digit state, 3 digit county, and 6 digit tract code (with no decimal). The county code is a 5 digit number that combines state and county codes. The FFIEC census tool and FFIEC geocoder can assist in providing the correct state, county and census tract combinations.

 Consumer Financial Protection Bureau (.gov)
<https://hmdahelp.consumerfinance.gov/article/The-pl...>

[How do I find the correct census tract number? - HMDA Help](#)



Refer to `def fetchCensus(self)` in `red_lines_starter.py`.

Step 7

Define `fetchIncome`.

Background Information

Redlining played a critical role in preserving and exacerbating racial wealth inequality in the United States. Appreciation of property value is one of the key mechanisms of intergenerational wealth transfer in the United States. By blocking access to homeownership by limiting access to government-insured mortgages (the alternatives were twice as expensive or more), African American citizens were not allowed to benefit from the infrastructure investments of the United States government. This mechanism is detailed in the following article:

<https://www.npr.org/2017/05/03/526655831/a-forgotten-history-of-how-the-u-s-government-segregated-america>

Additionally, it is important to know that while official government redlining was banned in the 1960s, there is some evidence of algorithmic redlining going on in the present.

- <https://revealnews.org/article/for-people-of-color-banks-are-shutting-the-door-to-homeownership/>
- <https://www.washingtonpost.com/news/wonk/wp/2018/03/14/the-senate-rolls-back-rules-meant-to-root-out-discrimination-by-mortgage-lenders/>

Now we will check for evidence of the legacy of redlining in Detroit.

1. Sign up for an API key for the US Census at: https://api.census.gov/data/key_signup.html (But you could still finish the homework without it)
2. Examine the documentation at <https://api.census.gov/data/2018/acs/acs5/variables.html> (where you could find the name of the desired file) and <https://api.census.gov/data/2018/acs/acs5/examples.html> (where you could reference how to make the API call) and use your census tract info to find the MEDIAN HOUSEHOLD INCOME IN THE PAST 12 MONTHS (IN 2018 INFLATION-ADJUSTED DOLLARS) for each tract. You will need to use your API and JSON skills from this course.
 - **Note you can and should get all of them at once by just specifying the state and then parsing the larger JSON file to get the specific data you need for each district.**
3. Add the median household income to the appropriate attribute.

Refer to `def fetchIncome(self)` in `red_lines_starter.py`.

Step 8.1

Define `cacheData`

Create a JSON cache that cache all the information of each district instance.

Refer to `def cacheData(self, fileName)` in `red_lines_starter.py`.

Step 8.2

Define `loadCache`. Refer to `def loadCache(self, fileName)` in `red_lines_starter.py`.

```
###Hint self.districts = [ DetroitDistrict(**data) for data in district_data]
```

Related Reading

<https://stackoverflow.com/questions/36901/what-does-double-star-asterisk-and-star-asterisk-do-for-parameters>

Step 8.3

Revise your `init` method of `RedLines`, which now takes an argument called `cacheFile`. The default value is `None`. If it is not `None`, call the load cache method. If it is `None`, set `self.districts = Empty List`.

The second time you run your code it should get the data from your cache and not the API (it should run much faster). You will lose points if your submitted code does not include the `redlines_cache.json`.

Step 9

Define `calcIncomeStats`

Refer to `def calcIncomeStats(self)` in `red_lines_starter.py`.

Step 10

Define `findCommonWords`

Refer to `def findCommonWords(self)` in `red_lines_starter.py`.

Bonus : 10 points

Define the following methods

Refer to `def calcRank(self)`, `def comment(self)` and `def calcPopu(self)` in `red_lines_starter.py`.

Submit

Please submit the following to gradescope

1. `red_lines.py` with the classes defined
2. `redlines_cache.json`
3. `redlines_graph.png`