BIOF309 Fall 2018

Kevin Oakley

“RMNDR”

Final Presentation

Merriam-Webster

object noun

ob·ject | \ˈäb-jikt, -(ˌ)jekt\

Definition of *object* (Entry 1 of 3)

1 a : something material that may be perceived by the senses
// I see an *object* in the distance.

WIKIPEDIA
The Free Encyclopedia

Article Talk

Object (computer science)

From Wikipedia, the free encyclopedia

In computer science, an **object** can be a variable, a data structure, a function, or a method, and as such, is a value in memory referenced by an identifier.

In the class-based object-oriented programming paradigm, *object* refers to a particular instance of a class, where the object can be a combination of variables, functions, and data structures.

In relational database management, an object can be a table or column, or an association between data and a database entity (such as relating a person's age to a specific person).[1]

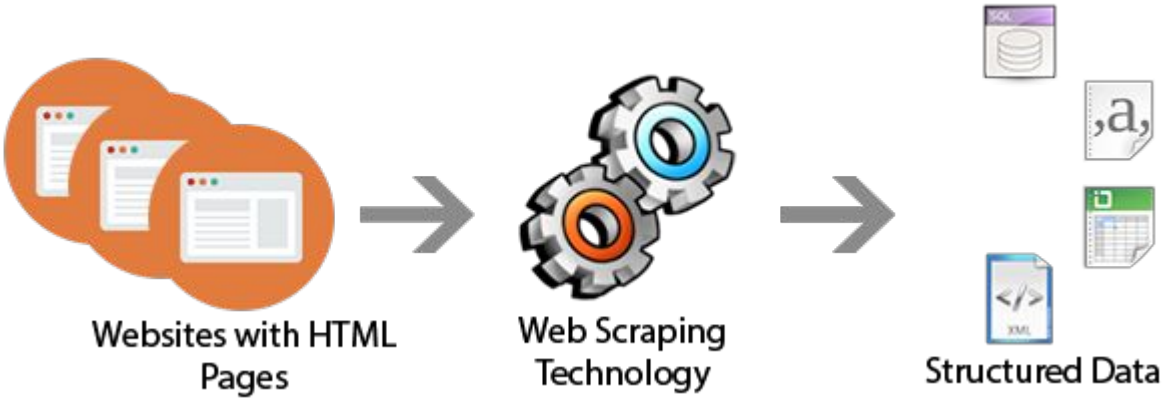
https://cdn-images-1.medium.com/max/2000/1*i1vVm3EqqDIkyucD0079wg.jpeg

https://www.python.org/static/img/python-logo@2x.png

https://www.teratotech.com/daily-thoughts/fun-daily-thoughts/programming-for-kids-some-tools/

Project Idea:

Apply some of the techniques from class and DataCamp tutorials in the “wild” to scrape the web for existing data, then submit queries against it for data visualization.



https://cdn-images-1.medium.com/max/1600/1*kfOsUxggG5wDbDcxgC0Uwg.png

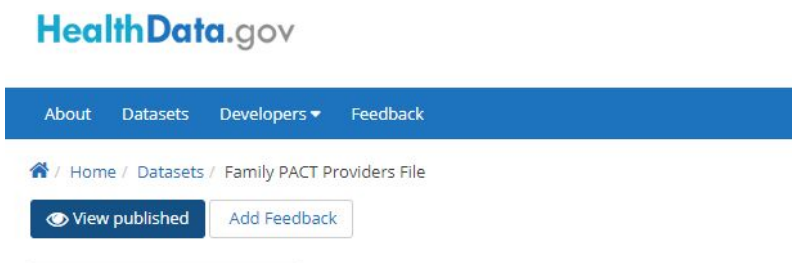
https://healthdata.gov/sites/default/files/logo.png

Family PACT Providers File

https://healthdata.gov/dataset/family-pact-providers-file

This dataset includes information on the Family Planning, Access, Care, and Treatment (Family PACT) Program providers for fiscal year (FY) 16-17. The data were retrieved from the Provider Master File (PMF) in the Management Information System/Decision Support System (MIS/DSS) data warehouse, which is maintained by the Provider Enrollment Division, and from the Office of Family Planning (OFP) which created its own production files/data sets. The Family PACT Program is administered by the California Department of Health Care Services, OFP Division and has been operating since 1997 to provide family planning and reproductive health services at no cost to California's low-income residents of reproductive age. The program offers comprehensive family planning services, including contraception, pregnancy testing, and sterilization, as well as sexually transmitted infection (STI) testing and limited cancer screening services. The variables in the dataset include provider number, name, type, specialty, geographic information, etc. For information about Family PACT provider enrollment please see http://www.familypact.org/Home/home-page.

Source: chhs.data.ca.gov



Source: chhs.data.ca.gov

Data and Resources

ArcGIS Open Dataset	Go to resource
Esri Rest API	Go to resource
GeoJSON	Download
CSV	Preview Download
KML	Download
Shapefile	Download

DHCS · Family PACT · Providers

Field	Value
Publisher	State of California
Modified	2018-12-06
Release Date	2018-12-06
Identifier	4f4cffc1-ed02-4749-9dba-bfd2e5f76305
License	Open Data Commons Open Database License (ODbL)
Author	State of California
Contact Name	CHHS Open Data
Contact Email	opendata@chhs.ca.gov
Public Access Level	Public

Harvested from chhs.data.ca.gov

Harvest Source Title	chhs.data.ca.gov
Harvest Source URI	https://data.chhs.ca.gov/data.json
Last Harvest Performed	Thu, 12/06/2018 - 02:21

Contains flat and json files

- ❑ Use Numpy and Pandas
- ❑ Use application programming interfaces (API) to import from the web
- ❑ Use Matplotlib and pyplot to pull from the data to create graphs

CSV

a2742f60dd944a1fa49377bd0e8a7772_0.csv

Data Preview: Note that by default the preview only displays up to 100 records. Use the pager to flip through more records or adjust the start and end fields to display the number of records you wish to see.

Grid | Graph | Map | 1830 records | « 1 – 100 » | Search data ... | Go » | Filters | Fields

X	Y	Provi...	NPI_...	Own...	Servi...	Provi...	Enrol...	Provi...	Provi...	Provi...	Provi...	Provi...	Out_...	O
-118....	33.88...	15189...	15189...	1	1	BEHR...	1995-...	22		FNP02...			0	
-118....	34.13...	13563...	13563...	1	1	COMP...	2006-...	58	HEALT...				0	
-118....	33.97...	11848...	11848...	1	1	BELL ...	2003-...	58	HEALT...				0	
-118....	33.97...	16997...	16997...	2	1	BELL ...	2013-...	22	PHYSI...		1	Gener...	0	
-118....	33.97...	16997...	16997...	2	1	BELL ...	2013-...	22	PHYSI...		1	Gener...	0	
-118....	34.03...	17102...	17102...	1	1	BELLA...	2012-...	22	PHYSI...				0	
-118....	33.88...	19321...	19321...	1	1	BELLF...	1997-...	58	HEALT...				0	
-117....	34.10...	14877...	14877...	1	1	BENN...	2003-...	26	PHYSI...	A72290	1	Gener...	0	
-122....	37.86...	15189...	15189...	1	1	BERK...	1996-...	58	HEALT...				0	
-122....	37.86...	19029...	19029...	1	1	BERK...	1977-...	41	COMM...	ZZR1...			0	
-122....	37.86...	19029...	19029...	1	1	BERK...	1977-...	45	CLINI...				0	
-117....	32.70...	18616...	18616...	1	1	25TH ...	2007-...	58	HEALT...				0	
-118....	33.97...	19129...	19129...	1	1	BERN...	2000-...	22	PHYSI...	FNP01...	16	Obste...	0	
-119....	35.39...	15887...	15887...	1	1	34TH ...	1997-...	58	HEALT...				0	
-118....	34.05...	16899...	16899...	1	1	7TH S...	2011-...	22	PHYSI...		8	Famil...	0	
-117....	32.74...	15688...	15688...	1	1	BEST ...	2015-...	5	CERTI...				0	
-118....	34.01...	11846...	11846...	1	1	BEVE...	1977-...	15	COMM...	19090...			0	
-118....	34.20...	13960...	13960...	1	1	AAA C...	2010-...	58	HEALT...	55000...			0	
-118....	33.87...	16291...	16291...	1	1	BEVE...	1990-...	26	PHYSI...	G000...	16	Obste...	0	
-118	33.96	13160	13160	1	1	ABAIA	2001-	26	PHYSI	A000	1	Gener	0	

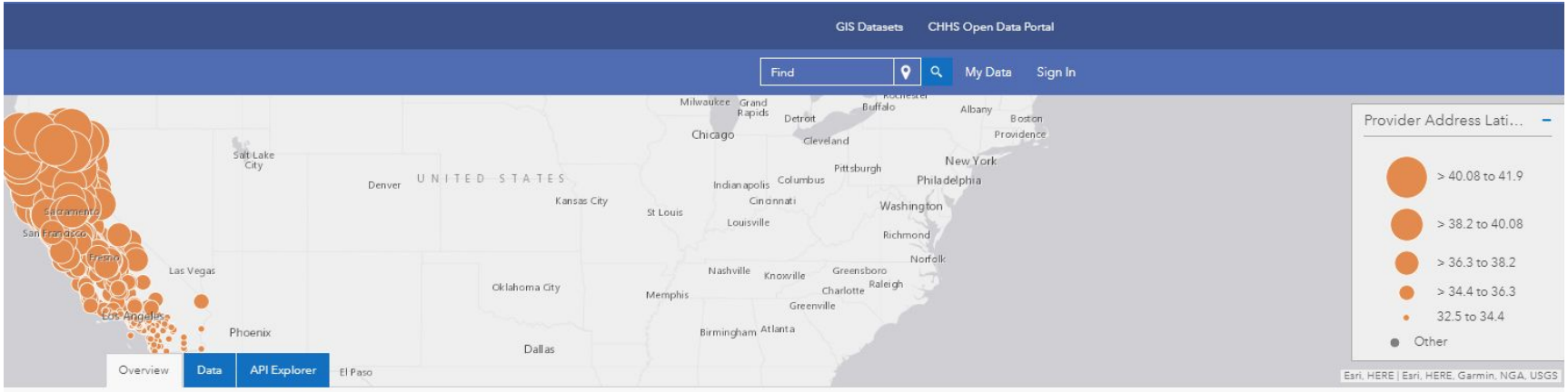
Resources

- ArcGIS Open Dataset
- Esri Rest API
- GeoJSON
- **CSV**
- KML

Additional Information

Field	Value
mimetype	text/csv
filesize	455.89 KB
resource type	file upload
timestamp	Dec 06, 2018

The csv file contains 28 columns and 1830 rows (records) of data



Family PACT Providers File

Favorite | Download | APIs

No license specified | 8/23/2018 | Spatial Dataset | 1,830 Rows

This dataset includes information on the Family Planning, Access, Care, and Treatment (Family PACT) Program providers for fiscal year (FY) 16-17. The data were retrieved from the Provider Master File (PMF) in the Management Information System/Decision Support System (MIS/DSS) data warehouse, which is maintained by the Provider Enrollment Division, and from the Office of Family Planning (OFP) which created its own production files/data sets. The Family PACT Program

More

About

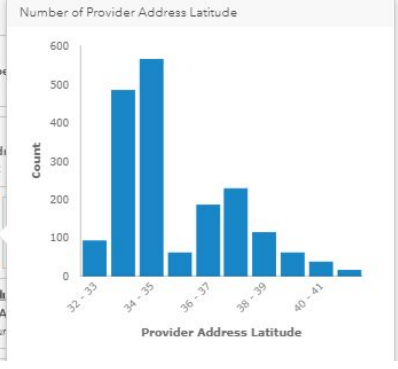
GeoHub Department of Health Care Services
Shared By: Adriana.Valdez
Data Source: services7.arcgis.com

View Metadata
Create Webmap
Create a Story Map

Attributes

Chart • Map Visualization

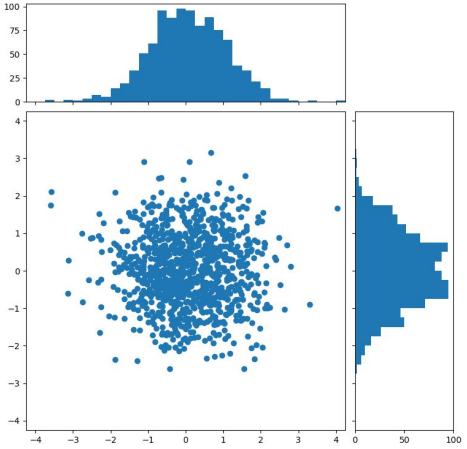
Enrollment Status Effective date — Date or Time
MSSA_ID — Text
NPI Provider Numbe — Number
Owner Number — Number
Provider Address Attention Line — Text
Provider Addr — Text
Provider Address County Code Desc — Text
Provider Address Latitude — Number
Provider Address Longitude — Number
Provider Address State — Text
Provider A — Nu

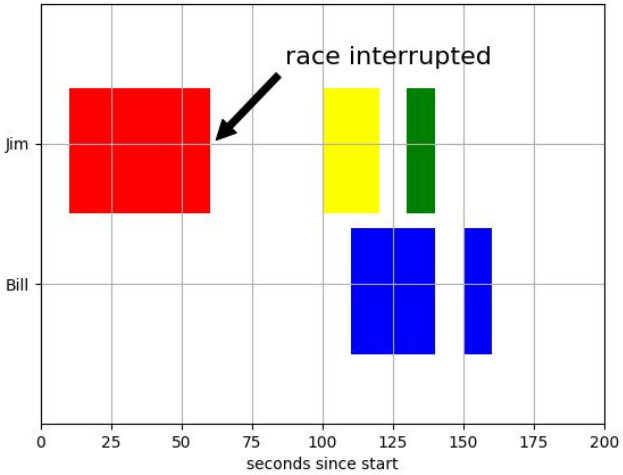


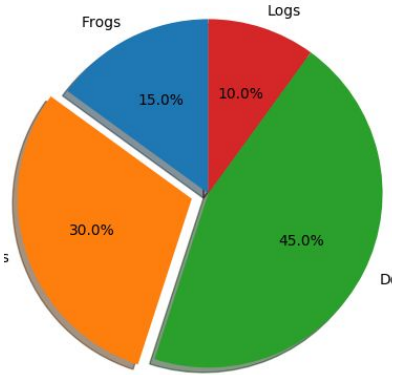
The link for the [ArcGIS open dataset] resource page is interactive and automatically populates some charts for the attributes of their dataset.

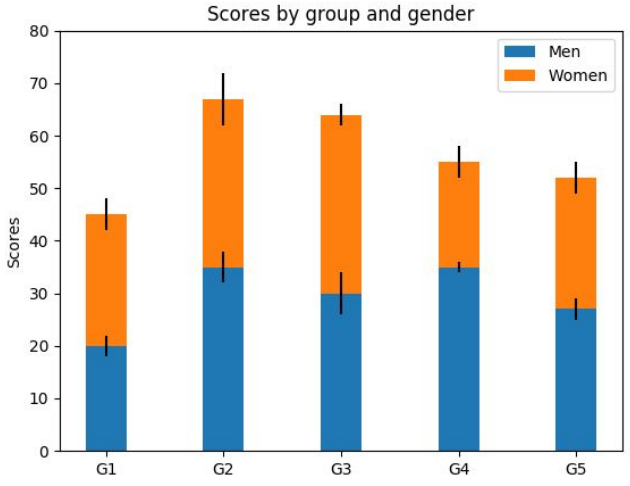
My goal was to then query the data to apply Matplotlib code (referring to tutorials on their site) to visualize the data in different ways than the automated 2 variable charts that the open dataset provides.

- ❏ Stacked Bar Graphs
- ❏ Scatter Histograms
- ❏ Pie Charts
- ❏ etc.









Objectives:

1. Create a Jupyter Notebook to work with the data
2. Connect to the data
3. Understand the composition of the data
4. Code to manipulate the data to extract information

Goal:

Solve for a way to determine the quantity and description of providers in any given zip code found in the dataset.

jupyter PACT data - Final-Copy1 Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run Code

In [1]:
```
### Import packages to pull csv file from repo for analysis
import requests
import pandas as pd
import numpy as np
from sqlalchemy import create_engine
import matplotlib.pyplot as plt
from urllib.request import urlretrieve
```

In [2]:
```
### Assign URL to variable: url
url = 'https://raw.githubusercontent.com/BIOF309/group-project-rmndr/master/Family_PACT_Providers_File.csv'
```

In [3]:
```
### Apply pandas package to read the .csv file: url
df = pd.read_csv(url)
```

In [4]:
```
### Display the DataFrame in Notebook
df
```

Out[4]:

	X	Y	Provider_Number	NPI_Provider_Number	Owner_Number	Service_Location_Number	Provider_Businness_Legal_Name	Enrollme
0	-118.160002	33.889872	1518996313	1518996313	1	1	BEHROOZ YAGOOBIAN, MD	1(
1	-118.239588	34.137132	1356362743	1356362743	1	1	COMPREHENSIVE COMM HLTH	2(
2	-118.147588	33.972803	1184833501	1184833501	1	1	BELL GARDEN FAMILY	2(
3	-118.179412	33.977217	1699767517	1699767517	2	1	BELL PLAZA MEDICAL CLINIC	2(
4	-118.179412	33.977217	1699767517	1699767517	2	1	BELL PLAZA MEDICAL CLINIC	2(
5	-118.208139	34.030985	1710261284	1710261284	1	1	BELLA MEDICAL GROUP INC	2(
6	-118.120765	33.882105	1932115375	1932115375	1	1	BELLFLOWER HEALTH CENTER	1(
7	-117.435811	34.103851	1487742979	1487742979	1	1	BENNY J. GUZMAN, M.D., CORPO	2(
8	-122.269858	37.863641	1518963693	1518963693	1	1	BERKELEY PRIMARY CARE	1(

In [5]:
```
### Display the shape of the DataFrame
print(np.shape(df))
```

```
(1830, 28)
```

In [6]:
```
### Get information on DataFrame
df.info()
```

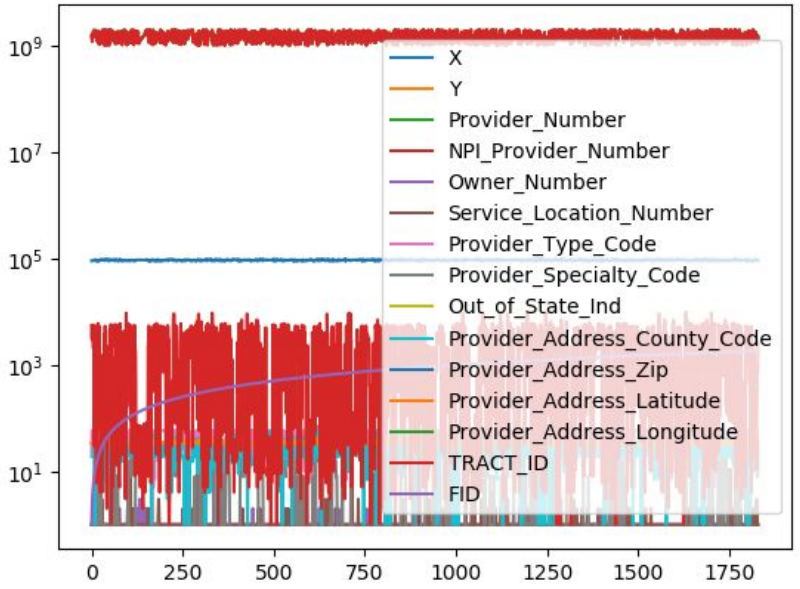
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1830 entries, 0 to 1829
Data columns (total 28 columns):
X                                   1830 non-null float64
Y                                   1830 non-null float64
Provider_Number                     1830 non-null int64
NPI_Provider_Number                 1830 non-null int64
Owner_Number                        1830 non-null int64
Service_Location_Number             1830 non-null int64
Provider_Businness_Legal_Name       1830 non-null object
Enrollment_Status_Effective_dat     1830 non-null object
Provider_Type_Code                  1830 non-null int64
Provider_Type_Code_Desc             1636 non-null object
Provider_License_Number             685 non-null object
Provider_Specialty_Code             633 non-null float64
Provider_Specialty_Code_Desc        502 non-null object
Out_of_State_Ind                    1830 non-null int64
Out_of_State_Desc                   1830 non-null object
Provider_Address_County_Code        1829 non-null float64
Provider_Address_County_Code_De     1829 non-null object
Provider_Address_Attention_Line     1017 non-null object
Provider_Address_Line_1             1830 non-null object
Provider_Address_Line_2             632 non-null object
Provider_Address_City               1830 non-null object
Provider_Address_State              1830 non-null object
Provider_Address_Zip                1830 non-null int64
Provider_Address_Latitude           1830 non-null float64
Provider_Address_Longitude          1830 non-null float64
MSSA_ID                             1830 non-null object
TRACT_ID                            1830 non-null float64
FID                                 1830 non-null int64
dtypes: float64(7), int64(8), object(13)
memory usage: 400.4+ KB
```

In [7]:
```
### Test ability to manipulate DataFrame by display the last 6 columns of the first 5 rows
test = df.iloc[:,-6:]
test.head(5)
```

Out[7]:

	Provider_Address_Zip	Provider_Address_Latitude	Provider_Address_Longitude	MSSA_ID	TRACT_ID	FID
0	90723	33.889872	-118.160002	78.2m	5539.01	1
1	91205	34.137132	-118.239588	78.2ff	3021.02	2
2	90201	33.972803	-118.147588	78.2c	5339.02	3
3	90201	33.977217	-118.179412	78.2ddd	5338.04	4
4	90201	33.977217	-118.179412	78.2ddd	5338.04	5

In [8]:
```
### Visualize the DataFrame without conditions
df.plot()
plt.yscale('log')
```



In [9]:
```
### Plot all columns as subplots
df.plot(subplots=True)
plt.show()
```

In [10]:
```
### Return the number of entries of zipcodes in the series (column)
df['Provider_Address_Zip'].describe()
```

Out[10]:
```
count     1830.000000
mean     92610.478142
std       1878.618118
min      90001.000000
25%      90813.000000
50%      92411.000000
75%      93960.000000
max      96161.000000
Name: Provider_Address_Zip, dtype: float64
```

In [11]:
```
###  Find all of the unique zipcodes from the DataFrame; assign to zc_array
zc_array = df['Provider_Address_Zip'].unique()
zc_array.shape
```

Out[11]:
```
(704,)
```

In [12]:
```
### Sort the df by zipcode
df.sort_values(by='Provider_Address_Zip')
```

Out[12]:

	X	Y	Provider_Number	NPI_Provider_Number	Owner_Number	Service_Location_Number	Provider_Businness_Legal_Name	Enrollme
900	-118.254103	33.974765	1417005695	1417005695	1	1	JOHNSON, LORNA M CNM	1(
332	-118.251468	33.974773	1649426990	1649426990	1	1	FLORENCE HOOPER FAMILY MEDIC	2(
334	-118.246322	33.974759	1013932979	1013932979	1	1	FLORENCE MEDICAL CTR INC	1(
1219	-118.241186	33.982620	1750469318	1750469318	1	1	SGHIATTI, VINCENT R APC	1(
1507	-118.262163	33.960132	1235286436	1235286436	1	1	NEIGHBORHOOD MEDICAL CLINIC	2(
116	-118.254103	33.974765	1033267216	1033267216	1	1	ADVANCED FAMILY CARE	1(
19	-118.256403	33.961796	1316061674	1316061674	1	1	ABAIAN, ALI MD	2(
1725	-118.249384	33.974773	1750442836	1750442836	1	2	ZACOALCO MEDICAL GROUP	1(
83	-118.256441	33.985662	1235283094	1235283094	1	1	CENTRAL CITY COMM HLTH	2(

In [13]:
```
### Parse the data to reindex against the Provider_Type_Code, Provider_Type_Code_Desc and Provider_Address_Zip
parsed = pd.read_csv(url, index_col=['Provider_Type_Code', 'Provider_Type_Code_Desc','Provider_Address_Zip'])
parsed
```

Out[13]:

Provider_Type_Code	Provider_Type_Code_Desc	Provider_Address_Zip	X	Y	Provider_Number	NPI_Provider_Number	Owner_Number	Ser
22	NaN	90723	-118.160002	33.889872	1518996313	1518996313	1	
58	HEALTH ACCESS PROGRAM	91205	-118.239588	34.137132	1356362743	1356362743	1	
		90201	-118.147588	33.972803	1184833501	1184833501	1	
22	PHYSICIANS GROUP	90201	-118.179412	33.977217	1699767517	1699767517	2	
		90201	-118.179412	33.977217	1699767517	1699767517	2	
		90023	-118.208139	34.030985	1710261284	1710261284	1	
58	HEALTH ACCESS PROGRAM	90706	-118.120765	33.882105	1932115375	1932115375	1	
26	PHYSICIANS	92335	-117.435811	34.103851	1487742979	1487742979	1	
58	HEALTH ACCESS PROGRAM	94704	-122.269858	37.863641	1518963693	1518963693	1	

In [14]:
```
### sort the parsed data
zc = parsed.sort_values(by='Provider_Address_Zip')
zc
```

Out[14]:

Provider_Type_Code	Provider_Type_Code_Desc	Provider_Address_Zip	X	Y	Provider_Number	NPI_Provider_Number	Owner_Number	Ser
5	CERTIFIED NURSE MIDWIFE	90001	-118.254103	33.974765	1417005695	1417005695	1	
22	PHYSICIANS GROUP	90001	-118.251468	33.974773	1649426990	1649426990	1	
	NaN	90001	-118.246322	33.974759	1013932979	1013932979	1	
26	PHYSICIANS	90001	-118.241186	33.982620	1750469318	1750469318	1	
		90001	-118.262163	33.960132	1235286436	1235286436	1	
22	NaN	90001	-118.254103	33.974765	1033267216	1033267216	1	
26	PHYSICIANS	90001	-118.256403	33.961796	1316061674	1316061674	1	
22	PHYSICIANS GROUP	90001	-118.249384	33.974773	1750442836	1750442836	1	
58	HEALTH ACCESS	90001						

In [15]:
```
### Ask user their zipcode (Attempted to raise a ValueError, but was not successful)
CAzip = int(input('What is your zipcode?: '))
type(CAzip)
```

```
What is your zipcode?: 90001
```

Out[15]:
```
int
```

I found an example of MultiIndex slicing for cross-sections (xs) on StackOverflow to guide me through this code

In [16]:
```
### Find all rows that contain input CAzip
xs = pd.IndexSlice
row = zc.loc[xs[:,:,CAzip],:]
row
```

Out[16]:

Provider_Type_Code	Provider_Type_Code_Desc	Provider_Address_Zip	X	Y	Provider_Number	NPI_Provider_Number	Owner_Number	Service
5	CERTIFIED NURSE MIDWIFE	90001	-118.254103	33.974765	1417005695	1417005695	1	
22	PHYSICIANS GROUP	90001	-118.251468	33.974773	1649426990	1649426990	1	
	NaN	90001	-118.246322	33.974759	1013932979	1013932979	1	
26	PHYSICIANS	90001	-118.241186	33.982620	1750469318	1750469318	1	
		90001	-118.262163	33.960132	1235286436	1235286436	1	
22	NaN	90001	-118.254103	33.974765	1033267216	1033267216	1	
26	PHYSICIANS	90001	-118.256403	33.961796	1316061674	1316061674	1	
22	PHYSICIANS GROUP	90001	-118.249384	33.974773	1750442836	1750442836	1	
58	HEALTH ACCESS PROGRAM	90001	-118.256441	33.985662	1235283094	1235283094	1	
26	PHYSICIANS	90001	-118.234728	33.974782	1134206436	1134206436	1	

10 rows × 25 columns

In [17]:
```
### Sort resulting dataframe by Provider_Type_Code
row.sort_values(by='Provider_Type_Code')
```

Out[17]:

Provider_Type_Code	Provider_Type_Code_Desc	Provider_Address_Zip	X	Y	Provider_Number	NPI_Provider_Number	Owner_Number	Service
5	CERTIFIED NURSE MIDWIFE	90001	-118.254103	33.974765	1417005695	1417005695	1	
22	PHYSICIANS GROUP	90001	-118.251468	33.974773	1649426990	1649426990	1	
	NaN	90001	-118.246322	33.974759	1013932979	1013932979	1	
		90001	-118.254103	33.974765	1033267216	1033267216	1	
	PHYSICIANS GROUP	90001	-118.249384	33.974773	1750442836	1750442836	1	
26	PHYSICIANS	90001	-118.241186	33.982620	1750469318	1750469318	1	
		90001	-118.262163	33.960132	1235286436	1235286436	1	
		90001	-118.256403	33.961796	1316061674	1316061674	1	
		90001	-118.234728	33.974782	1134206436	1134206436	1	
58	HEALTH ACCESS PROGRAM	90001	-118.256441	33.985662	1235283094	1235283094	1	

10 rows × 25 columns

In [18]:
```
### Select rows and columns for zipcode to determine the count of unique zipcodes in DataFrame
only_zip = df.loc[:,'Provider_Address_Zip']
uzips = only_zip.unique()
soz = only_zip.sort_values()
pd.value_counts(soz)
```

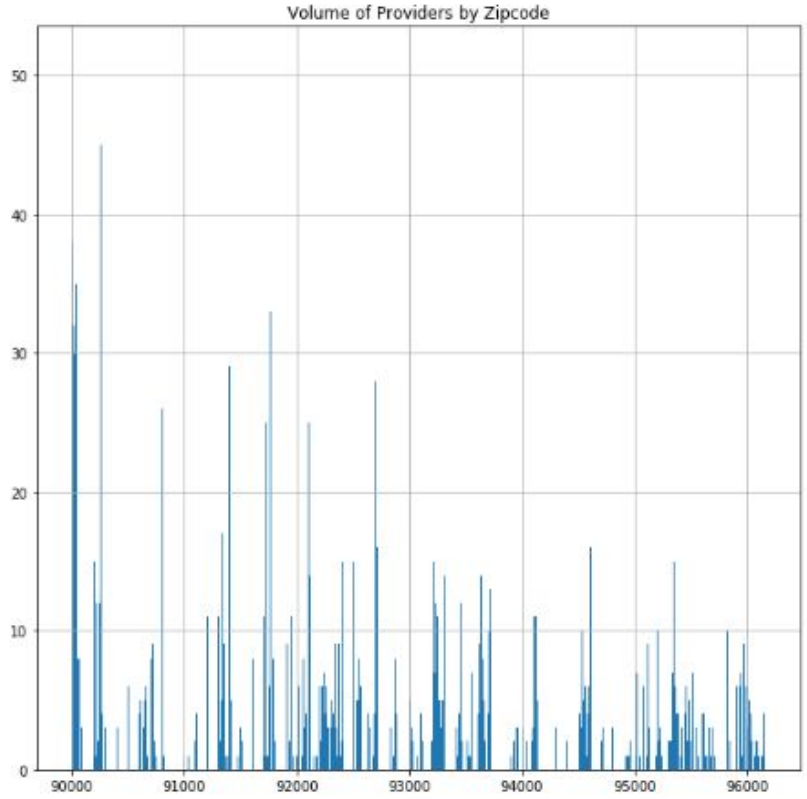
Out[18]:
```
90255    28
90057    18
92701    16
90023    16
90280    15
90201    15
90037    14
91767    13
90806    13
90015    12
90022    12
91402    12
90033    11
91731    11
90262    11
90001    10
90813    10
93454    10
91950    10
90029     9
91303     9
93230     9
91331     9
91744     9
90250     8
91340     8
90006     8
90011     8
93618     8
90723     8
         ..
95519     1
95536     1
95388     1
95694     1
93235     1
95382     1
93241     1
93249     1
91202     1
```

In [11]:
```
###  Find all of the unique zipcodes from the DataFrame; assign to zc_array
zc_array = df['Provider_Address_Zip'].unique()
zc_array.shape
```

Out[11]:
```
(704,)
```

In [19]:
```
### Show a crude Histogram - Distribution of providers by zipcode
plt.figure(figsize=(10,10)) # figure size argument
df['Provider_Address_Zip'].hist(bins=704)
plt.title('Volume of Providers by Zipcode')
```

Out[19]: Text(0.5,1,'Volume of Providers by Zipcode')



Going Forward

1. Continue to explore the world of coding in Python
2. Choose a different dataset with more object types (integers and floats) that will be more amenable to statistical and visualization tools
3. Keep practicing

https://image.slidesharecdn.com/servicediscoverywitheureka-160825025146/95/service-discovery-with-eureka-and-spring-cloud-16-638.jpg?cb=1475359463