SanFranBuildingPermits

San Francisco, CA Building Permit Mapping¶

About Me¶

- **Author**: Marcus Collins
- **Date**: 2024-Sep-19

Credits¶

- **Data Source**: San Francisco Building Permits OpenGov database

Documentation URLs¶

- **Python 3**: Python 3 Docs
- **NumPy**: NumPy API Reference
- **Pandas**: Pandas API Reference
- **Folium**: Folium API Reference
- **IPython**: IPython API Reference

Introduction¶

This is meant to demonstrate mapping within Python specifically using Pandas, IPython, NumPy and Folium.

Code¶

Imports¶

In [1]:

```
## standard Python included imports
import pathlib
import re

## specially installed Python imports
try:
    import numpy as np
```

```
    import pandas as pd
    import folium as FMap
    from folium import plugins
    import IPython
    from IPython.display import display
except(ImportError) as e:
    print(f"Error: {e.args}")
```

Pathnames and Filenames¶

In [2]:

```
imgpath = pathlib.Path("images")
htmlpath = pathlib.Path("html")
datapath = pathlib.Path("data")
```

In [3]:

```
permitcsvfile = datapath / "SanFranBuildingPermits_2023Jan-2023Dec.csv"
```

Basic Map¶

In [4]:

```
## set upper, lower bounds (latitudes {North,South}, longitudes {East,West}) for limiting ma
lat_min, lat_max = 37.54, 37.85
lon_min, lon_max = -122.54, -122.33
sfbasic = FMap.Map(
    location = (37.75429983275193, -122.44652683825896),
    tiles = "OpenStreetMap",
    max_bound = True,
    min_lat = lat_min,
    max_lat = lat_max,
    min_lon = lon_min,
    max_lon = lon_max,
    zoom_start = 12
)
#sfbasic.save("html/SanFranBasicMap.html")
```

In [69]:

```
sfbasic
```

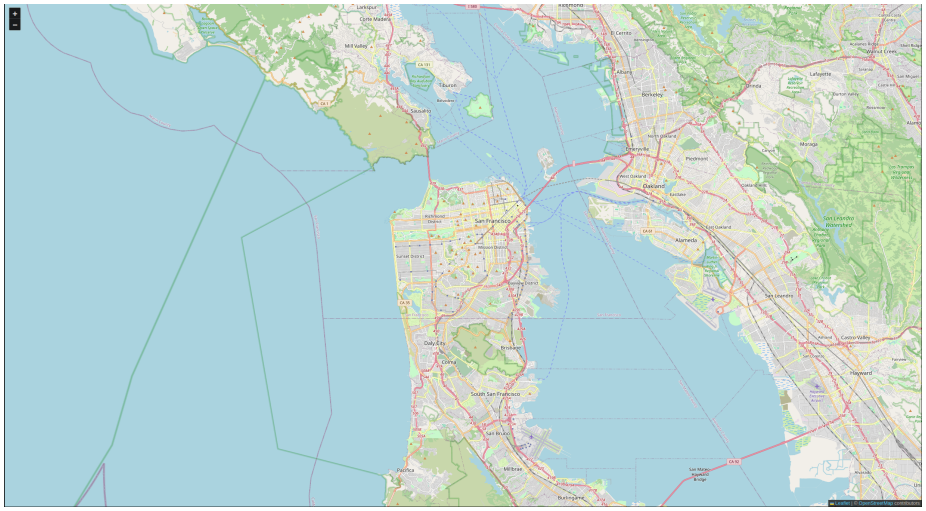
Out[69]:

Make this Notebook Trusted to load map: File -> Trust Notebook

In [5]:

```
basicmapfile = imgpath / "SanFranBasicMap.png"
IPython.display.Image(basicmapfile, format="png", embed=True)
```

Out[5]:



Into - Top

Previewing Data Before Sampling¶ In [6]:

```
# open full CSV file to get relevant grouping information
permit_df = pd.read_csv(
    permitcsvfile,
    header = 0,            # row(0) has the column names
    on_bad_lines = "warn",
    memory_map = True      # use local RAM instead of constantly using I/O streams
)
```

In [7]:

```
display( permit_df.info() )
print("")
display( permit_df.head(n=15) )
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 24895 entries, 0 to 24894
Data columns (total 51 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   Permit Number                           24895 non-null  object
 1   Permit Type                             24895 non-null  int64
 2   Permit Type Definition                  24895 non-null  object
 3   Permit Creation Date                    24895 non-null  object
 4   Block                                   24895 non-null  object
 5   Lot                                     24895 non-null  object
```

```
 6   Street Number                             24895 non-null  int64
 7   Street Number Suffix                      396 non-null    object
 8   Street Name                               24895 non-null  object
 9   Street Suffix                             24562 non-null  object
 10  Unit                                      3327 non-null   float64
 11  Unit Suffix                               285 non-null    object
 12  Description                               24891 non-null  object
 13  Current Status                            24895 non-null  object
 14  Current Status Date                       24895 non-null  object
 15  Filed Date                                24892 non-null  object
 16  Issued Date                               22417 non-null  object
 17  Completed Date                            14873 non-null  object
 18  First Construction Document Date          18 non-null     object
 19  Structural Notification                   635 non-null    object
 20  Number of Existing Stories                22888 non-null  float64
 21  Number of Proposed Stories                22929 non-null  float64
 22  Voluntary Soft-Story Retrofit             0 non-null      float64
 23  Fire Only Permit                          3847 non-null   object
 24  Estimated Cost                            24735 non-null  object
 25  Revised Cost                              24274 non-null  object
 26  Existing Use                              24346 non-null  object
 27  Existing Units                            20005 non-null  object
 28  Proposed Use                              24275 non-null  object
 29  Proposed Units                            20227 non-null  object
 30  Plansets                                  24772 non-null  float64
 31  TIDF Compliance                           0 non-null      float64
 32  Existing Occupancy                        24307 non-null  object
 33  Proposed Occupancy                        24291 non-null  object
 34  Existing Construction Type                22832 non-null  object
 35  Existing Construction Type Description    22822 non-null  object
 36  Proposed Construction Type                22887 non-null  object
 37  Proposed Construction Type Description    22877 non-null  object
 38  Site Permit                               297 non-null    object
 39  Last Permit Activity Date                 24848 non-null  object
 40  Application Submission Method             24895 non-null  object
 41  ADU                                       24895 non-null  object
 42  Primary Address Flag                      23048 non-null  object
 43  Supervisor District                       24880 non-null  float64
 44  Neighborhoods - Analysis Boundaries       24880 non-null  object
 45  Zipcode                                   24865 non-null  float64
 46  Location                                  24880 non-null  object
 47  Point Source                              24880 non-null  object
 48  Record ID                                 24895 non-null  int64
 49  data_as_of                                24895 non-null  object
 50  data_loaded_at                            24895 non-null  object
dtypes: float64(8), int64(3), object(40)
```

```
memory usage: 9.7+ MB
```

```
None
```

	Permit Number	Permit Type	Permit Type Definition	Permit Creation Date	Block	Lot
0	202301039642	8	otc alterations permit	01/03/2023	3651	035
1	202301039602	8	otc alterations permit	01/03/2023	0225	017
2	202301039614	4	sign - erect	01/03/2023	0313	003
3	202301039611	8	otc alterations permit	01/03/2023	1239	026
4	202301039615	8	otc alterations permit	01/03/2023	6965B	017
5	202301039620	8	otc alterations permit	01/03/2023	6748	034
6	202301039601	7	wall or painted sign	01/03/2023	0315	003
7	202301039628	4	sign - erect	01/03/2023	0288	025
8	202301039600	8	otc alterations permit	01/03/2023	0145	023
9	202301039641	8	otc alterations permit	01/03/2023	7064	043
10	202301039625	8	otc alterations permit	01/03/2023	7055A	026
11	202301039637	8	otc alterations permit	01/03/2023	2717	012
12	202301039623	3	additions alterations or repairs	01/03/2023	0748	016
13	202301039638	3	additions alterations or repairs	01/03/2023	6636	012
14	202301039598	8	otc alterations permit	01/03/2023	2450	021

15 rows × 51 columns

In [8]:

```
permit_types_df = permit_df[["Permit Type","Permit Type Definition"]].value_counts()
display( permit_types_df )
```

```
Permit Type  Permit Type Definition
8            otc alterations permit                 22880
3            additions alterations or repairs        1465
4            sign - erect                             307
2            new construction wood frame               86
7            wall or painted sign                      73
6            demolitions                               47
1            new construction                          31
5            grade or quarry or fill or excavate        6
Name: count, dtype: int64
```

In [9]:

```
# Create a list of the permit types
permit_types_list = list(range(1,9)); display(permit_types_list)
```

```
[1, 2, 3, 4, 5, 6, 7, 8]
```

Sample the full DataFrame¶ In [10]:

```
# Create an array of [random] row indices
df_size = len(permit_df);  display(df_size) # original was 24895

sample_size = np.int64(1000)

### Modern way to setup a random number generator (rngen) with a seed value
###   See NumPy documentation for the various generator functions available
rngen = np.random.default_rng(1357911)

### rngen.choice(
###    Largest Value or ArrayRange,
###    size=Number of Choices,
###    shuffle=Rearrange between picks,
###    replace=Choices are aviable again if picked
### )
sample_indices = pd.Index(rngen.choice(df_size, size=sample_size, shuffle=False, replace=Fal
```

```
24895
```

```
Index([6298, 226, 16633, 13903, 18923, 13243, 12075], dtype='int64')
```

In [11]:

```
# Keep some columns and reset the index

### List of column names
keep_cols = ["Permit Number", "Permit Type", "Permit Type Definition", "Filed Date", "Descri

### Make a "true" copy of the rows to be sampled rather than a view of the data (referring m
sampled_df = permit_df.loc[sample_indices, keep_cols].copy(deep=True)

### keep the old row (index) values in a new column and reset the new row numbers within the
sampled_df.reset_index(drop=False, inplace=True)

### Convert the "Filed Date" column to DateTime objects instead of strings (provides more fl
sampled_df["Filed Date"] = sampled_df["Filed Date"].apply(lambda x: pd.to_datetime(x))

### Drop any row(s) that have an empty "Location" value
sampled_df.dropna(subset=["Location"], inplace=True)
```

In [12]:

```
display( sampled_df.info() )
print("")
display( sampled_df.head(n=5) )
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 999 entries, 0 to 999
```

```
Data columns (total 7 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   index                   999 non-null    int64
 1   Permit Number           999 non-null    object
 2   Permit Type             999 non-null    int64
 3   Permit Type Definition  999 non-null    object
 4   Filed Date              999 non-null    datetime64[ns]
 5   Description             999 non-null    object
 6   Location                999 non-null    object
dtypes: datetime64[ns](1), int64(2), object(4)
memory usage: 62.4+ KB
```

None

	index	Permit Number	Permit Type	Permit Type Definition	Filed Date	Description
0	6298	202304135626	8	otc alterations permit	2023-04-13 09:02:11	voluntary syste
1	226	202301069771	8	otc alterations permit	2023-01-06 11:09:38	5th floor:fire al
2	16633	202308285457	8	otc alterations permit	2023-08-28 15:12:41	for complaint n
3	13903	202307242770	8	otc alterations permit	2023-07-24 08:45:18	f/a modificatio
4	18923	202309287561	8	otc alterations permit	2023-09-28 07:48:36	fire alarm modi

In [13]:

```
# Delete the original DataFrame to save memory
del permit_df, permit_types_df
```

In [14]:

```
# Slim down the memory usage of sampled_df and show info

### Convert "Permit Type" to an unsigned 8-bit integer
sampled_df['Permit Type'] = sampled_df['Permit Type'].astype(np.uint8)

### Convert "Permit Type Definition" to a Pandas Categorical
sampled_df['Permit Type Definition'] = pd.Categorical(sampled_df['Permit Type Definition'].v
```

In [15]:

```
# Write function to be called by apply function on "Location" column to extract Latitudes ar
def extractLatLon(locstr):
    point_pattern = '^POINT\\s*\\(([-]*[\\d]+\\.[\\d]+)\\s+(-*[\\d]+\\.[\\d]+)\\)$'
    re_controller = re.compile(point_pattern)

    #display( re_controller ) ## used for testing only
    re_match = re_controller.findall(locstr)
```

```
    #display( locstr ) ## used for testing only
    #display( f"{re_match[0][1]}, {re_match[0][0]}" )  ## used for testing only

    ## Returns a formatted string to make splitting into separate columns simpler using the
    return f"{re_match[0][1]},{re_match[0][0]}"
```

In [16]:

```
# Split 'Location' into separate columns and show information
sampled_df[['Latitude','Longitude']] = sampled_df['Location'].apply(extractLatLon).str.split
sampled_df['Latitude'] = sampled_df['Latitude'].astype(np.float32)
sampled_df['Longitude'] = sampled_df['Longitude'].astype(np.float32)

sampled_df.drop(columns=['Location'], inplace=True)
```

In [17]:

```
display( sampled_df.info() )
print("")
display( sampled_df.head(n=5) )
print("")
display( sampled_df.tail(n=5) )
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 999 entries, 0 to 999
Data columns (total 8 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   index                   999 non-null    int64
 1   Permit Number           999 non-null    object
 2   Permit Type             999 non-null    uint8
 3   Permit Type Definition  999 non-null    category
 4   Filed Date              999 non-null    datetime64[ns]
 5   Description             999 non-null    object
 6   Latitude                999 non-null    float32
 7   Longitude               999 non-null    float32
dtypes: category(1), datetime64[ns](1), float32(2), int64(1), object(2), uint8(1)
memory usage: 49.1+ KB

None
```

	index	Permit Number	Permit Type	Permit Type Definition	Filed Date	Description
0	6298	202304135626	8	otc alterations permit	2023-04-13 09:02:11	voluntary syste
1	226	202301069771	8	otc alterations permit	2023-01-06 11:09:38	5th floor:fire al
2	16633	202308285457	8	otc alterations permit	2023-08-28 15:12:41	for complaint n
3	13903	202307242770	8	otc alterations permit	2023-07-24 08:45:18	f/a modificatio

	index	Permit Number	Permit Type	Permit Type Definition	Filed Date	Description
4	18923	202309287561	8	otc alterations permit	2023-09-28 07:48:36	fire alarm modi

	index	Permit Number	Permit Type	Permit Type Definition	Filed Date	Descr
995	9938	202306019000	8	otc alterations permit	2023-06-01 09:15:39	volun
996	16784	202308295480	3	additions alterations or repairs	2023-08-29 09:11:56	replac
997	24892	202312293292	8	otc alterations permit	2023-12-29 14:26:28	re-roo
998	24569	202312223005	8	otc alterations permit	2023-12-22 09:41:45	to ob
999	20025	202310138624	8	otc alterations permit	2023-10-13 10:24:23	reroo

Map Samples¶

Choice of folium.Icon(colors=Values) ==> [‘red’, ‘blue’, ‘green’, ‘purple’, ‘orange’, ‘darkred’,
’lightred’, ‘beige’, ‘darkblue’, ‘darkgreen’, ‘cadetblue’, ‘darkpurple’, ‘white’, ‘pink’, ‘lightblue’, ‘lightgreen’, ‘gray’, ‘black’, ‘lightgray’]

If I were to choose 8: 'blue', 'purple', 'gray', 'darkpurple', 'darkgreen', 'lightblue', 'black', 'darkblue'

Note: I prefer to use RGB hex codes, because the Folium color string list is very limited.

In [62]:

```
# Create the map on which to add features, pins, etc.

sfpermitmap = FMap.Map(
    location = (37.75429983275193, -122.44652683825896),
    tiles = "OpenStreetMap",
    zoom_start = 14
)
```

In [63]:

```
# Create dictionary to control colors mapped to permit type

#### Color list is in the order to be associated with each permit type 1,2,...,8
color_list = ['#269A92', '#606060', '#660505', '#CC6600', '#B0B05E', '#3B620F', '#FF99CC', '
```

In [64]:

```
# Create Folium CircleMarkers with grouping, colors, tooltips and popups

### Create 8 map groups
permit_dct = dict()
```

```
for group, color in enumerate(color_list, start=1):
    if group not in permit_dct:
        permit_dct[group] = {
            "color":color,
            "grpobj": FMap.FeatureGroup(f"Permit Type {group}").add_to(sfpermitmap)
        }

display( permit_dct )
```

```
{1: {'color': '#269A92',
  'grpobj': <folium.map.FeatureGroup at 0x76121c86b410>},
 2: {'color': '#606060',
  'grpobj': <folium.map.FeatureGroup at 0x76121c86b4d0>},
 3: {'color': '#660505',
  'grpobj': <folium.map.FeatureGroup at 0x76121de462a0>},
 4: {'color': '#CC6600',
  'grpobj': <folium.map.FeatureGroup at 0x7612240ff020>},
 5: {'color': '#B0B05E',
  'grpobj': <folium.map.FeatureGroup at 0x76121ff64230>},
 6: {'color': '#3B620F',
  'grpobj': <folium.map.FeatureGroup at 0x7612240fef30>},
 7: {'color': '#FF99CC',
  'grpobj': <folium.map.FeatureGroup at 0x7612267a9d60>},
 8: {'color': '#4C74B7',
  'grpobj': <folium.map.FeatureGroup at 0x76121de45be0>}}
```

In [65]:

```
# Iterate over the DataFrame rows and create a Folium Icon within the appropriate group

#### Map Circles in Folium have their radius unit in meters instead of pixels like CircleMar
####  Additionally, Circles are fixed position & size while CircleMarkers change size based
circ_radius = float(30)
circ_opacity = float(0.45)
border_weight = float(1)


for row_tuple in sampled_df.itertuples():
    ##print( f"PermitID={row_tuple._2}, PermitType={row_tuple._3}, PermitTypeLabel={row_tupl
    if row_tuple._3 in permit_dct:
        FMap.Circle(
            location = (row_tuple.Latitude,row_tuple.Longitude),
            fill = True,
            radius = circ_radius,
            color = permit_dct[row_tuple._3]["color"],
            weight = border_weight,
```

```
            fill_color = permit_dct[row_tuple._3]["color"],
            fill_opacity = circ_opacity,
            tooltip = f"Permit ID {row_tuple._2}",
            popup = FMap.Popup(
                        html = "<div><b>Permit Number</b>: {}<br/><b>Type</b>: {}<br/><b>Typ
                        max_width = int(300)
                    )
        ).add_to(permit_dct[row_tuple._3]["grpobj"])

FMap.plugins.MousePosition(position="bottomleft").add_to(sfpermitmap)
FMap.LayerControl().add_to(sfpermitmap)
```

Out[65]:

```
<folium.map.LayerControl at 0x76121ce7cfb0>
```

Display Map with Markers¶ In [66]:

```
sfpermitmap
```

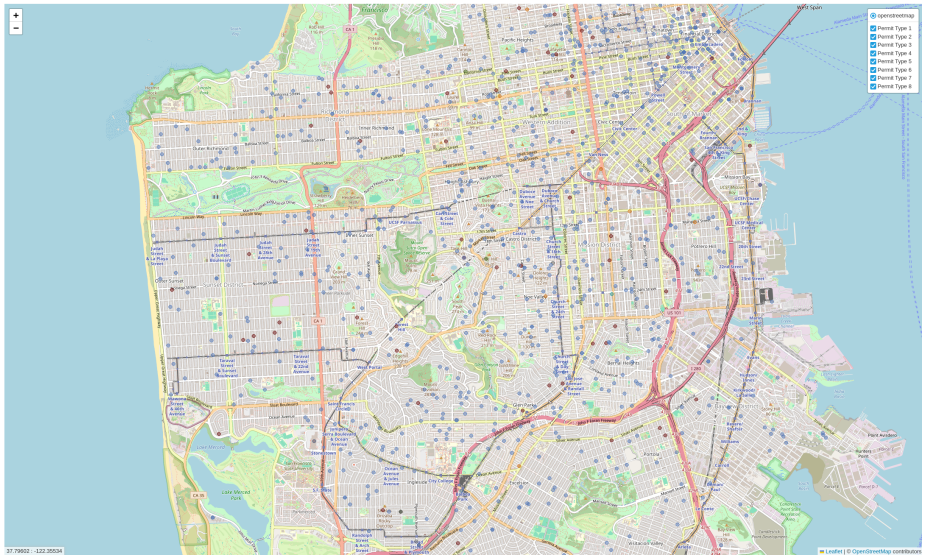
Out[66]:

Make this Notebook Trusted to load map: File -> Trust Notebook

Displaying screenshots for PDF since images only appear via nbconvert preview website otherwise¶ In [67]:

```
basic_pin_image = imgpath / "SF_Jupyter_FullMap-Legend.png";
IPython.display.Image(basic_pin_image, format="png", embed=True)
```

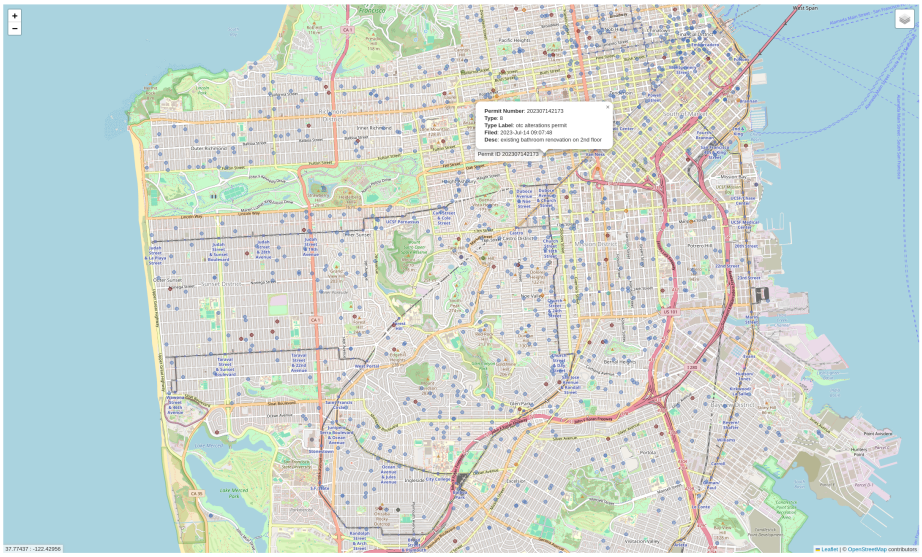
Out[67]:



In [68]:

```
tooltip_pin_image = imgpath / "SF_Jupyter_FullMap-Tooltip.png";
IPython.display.Image(tooltip_pin_image, format="png", embed=True)
```

Out[68]:



Go to **Top**