In [1]:
```
pip install geopy
```

```
Requirement already satisfied: geopy in c:\users\maltanno\anaconda3\envs\lear
n-env\lib\site-packages (2.0.0)Note: you may need to restart the kernel to us
e updated packages.
Requirement already satisfied: geographiclib<2,>=1.49 in c:\users\maltanno\an
aconda3\envs\learn-env\lib\site-packages (from geopy) (1.50)
```

In [2]:
```
import pandas as pd
import numpy as np
import requests
import json
import geopy.distance as GPD
import pickle
```

Data Acquisition

Found some Data at https://gis-kingcounty.opendata.arcgis.com/ (https://gis-kingcounty.opendata.arcgis.com/)
Data has lat and long for various types of places in King County including places of education, police/fire stations, airports and places for public gathering.
Though it had the data available to download, the download always failed (it was BIG). There were other ways with their own issues, I don't think poeple often want the data for the whole of King County. The way that seemed to work best is carried out below.
Unsure of how to change the starting point, I didn't have a way to make a second query after the first exceeded the transfer limit. I felt that an incomplete listing was worse than none so those I dropped.

In [20]:
```
lookup = pd.read_csv('data/sitetype_lookup.csv', index_col='value')
lookup.head(12)
```

Out[20]:

value	site
98	Other
99	No Gate No See
A1	Accessory Building
A2	Abandoned
A3	Group Mailbox
A9	Access Point
B1	Bridge
B2	Airport
B3	Helipad
B4	Campground
B9	Culvert
C1	Commercial

In [5]:
```
lookup.loc['98','site']
```

Out[5]:
```
'Other'
```

In [6]:
```
resp = requests.get("https://gisdata.kingcounty.gov/arcgis/rest/services/OpenDataPortal/property__parcel_address_area/MapServer/1722/query?where=SITETYPE%20%3D%20'R1'&outFields=SITETYPE,LAT,LON,ZIP5,PLUS4&outSR=4326&f=json")
```

In [25]:
```
x = json.loads(resp.text)
```

In [8]:
```
'exceededTransferLimit' not in x
```

Out[8]:
```
False
```

In [9]:
```
x['features'][0]['attributes']
```

Out[9]:
```
{'SITETYPE': 'R1',
 'LAT': 47.65611332,
 'LON': -122.35805293,
 'ZIP5': '98107',
 'PLUS4': '4928'}
```

In [12]:

```
data_lst = []
for value in lookup.index:
    resp = requests.get("https://gisdata.kingcounty.gov/arcgis/rest/services/OpenDataPortal/property__parcel_address_area/MapServer/1722/query?where=SITETYPE%20%3D%20'"+value+"'&outFields=SITETYPE,LAT,LON,ZIP5,PLUS4&outSR=4326&f=json")
    x = json.loads(resp.text)
    if 'exceededTransferLimit' not in x:
        for y in x['features']:
            entry = {}
            entry['sitetype'] = lookup.loc[value,'site'].replace(' ','_')
            entry['lat'] = y['attributes']['LAT']
            entry['lon'] = y['attributes']['LON']
            entry['zip'] = y['attributes']['ZIP5']
            entry['plus4'] = y['attributes']['PLUS4']
            data_lst.append(entry)
```

In [13]:
```
data = pd.DataFrame(data_lst)
data.groupby('sitetype').count()
```

Out[13]:

sitetype	lat	lon	zip	plus4
Abandoned	18	18	18	15
Access_Point	9	9	9	6
Accessory_Building	395	395	395	311
Airport	10	10	10	9
Camp/Bungalow	16	16	16	10
Campground	10	10	10	7
Cemetery	54	54	54	42
Commercial_Farm	75	75	75	67
Cultural	125	125	125	118
Educational	780	780	780	750
Fire	98	98	98	89
Gate_w/o_Building	5	5	5	3
Gated_w/_Building	28	28	28	22
Government	421	421	421	386
Group_Mailbox	1	1	1	1
Hanger	11	11	11	11
Lodging	379	379	379	360
No_Gate_No_See	5	5	5	5
Other	588	588	588	484
Other_Residential	315	315	315	306
PSAP	4	4	4	4
Police	14	14	14	13
Public_Gathering	762	762	762	593
Seasonal_Home	80	80	80	0
Town_Boundary_Point	1	1	1	1
Utility	736	736	736	505

Some sitetypes have too few entries, like Group_Mailbox, or are to unclear like other; I'll drop these:

In [14]:
```
to_drop = ['Group_Mailbox','No_Gate_No_See', 'Other', 'Other_Residential',
           'Town_Boundary_Point', 'Accessory_Building', 'Camp/Bungalow',
           'PSAP','Hanger']
keep = list(set(data.sitetype) - set(to_drop))
```

In [15]:
```
data = data[data.sitetype.isin(keep)]
```

In [16]:
```
data.to_csv('gis.csv', index=False)
```

In [ ]:
```

```

In [17]:
```
clean = pd.read_csv('data/clean.csv')
clean.head()
```

Out[17]:

	date	price	bedrooms	sqft_living	sqft_lot	waterfront	view	condition	grade	sqft_abov
0	164	221900.0	3	1180	5650	0	0	3	7	118
1	221	538000.0	3	2570	7242	0	0	3	7	217
2	299	180000.0	2	770	10000	0	0	3	6	77
3	221	604000.0	4	1960	5000	0	0	5	7	105
4	292	510000.0	3	1680	8080	0	0	3	8	168

In [23]:
```
data
```

Out[23]:

	sitetype	lat	lon	zip	plus4
988	Abandoned	47.657762	-121.909312	98014	6321
989	Abandoned	47.523178	-121.927341	98027	None
990	Abandoned	47.529621	-122.164212	98059	3214
991	Abandoned	47.458155	-122.337274	98166	3019
992	Abandoned	47.192230	-121.993525	98022	9470
...	...	...	...	...	...
4935	Utility	47.363479	-122.246452	98032	7270
4936	Utility	47.365357	-122.239844	98032	7379
4937	Utility	47.492411	-122.268456	98178	3432
4938	Utility	47.478652	-122.261931	98168	4721
4939	Utility	47.392477	-122.274941	98032	None

3604 rows × 5 columns

In [24]:
```
clean
```

Out[24]:

	date	price	bedrooms	sqft_living	sqft_lot	waterfront	view	condition	grade	sqft_a
0	164	221900.0	3	1180	5650	0	0	3	7	
1	221	538000.0	3	2570	7242	0	0	3	7	
2	299	180000.0	2	770	10000	0	0	3	6	
3	221	604000.0	4	1960	5000	0	0	5	7	
4	292	510000.0	3	1680	8080	0	0	3	8	
...	...	...	...	...	...	...	...	...	...	
21229	19	360000.0	3	1530	1131	0	0	3	8	
21230	297	400000.0	4	2310	5813	0	0	3	8	
21231	52	402101.0	2	1020	1350	0	0	3	7	
21232	259	400000.0	3	1600	2388	0	0	3	8	
21233	166	325000.0	2	1020	1076	0	0	3	7	

21234 rows × 20 columns

Now I have this data I need to use it in some way, make it compatible with the data provided. In the code below, for each row in clean I take its lat and long and for each sitetype in keep, use geopy to measure the distance between it and each site of that type, find the minimum and record this.
I also printed a number for each row, so I could see it was still running, and where it was up to.

In [ ]:
```
#Note code below takes a looooooooooooooooooooooooooooong time to run


row_dists = []
for e, row in enumerate(clean.iterrows()):
    print(e)
    lat1 = row[1]['lat']
    lon1 = row[1]['long']
    coords1 = (lat1, lon1)
    dist_cols = {}
    for st in keep:
        df = data[data['sitetype'] == st]
        dists = []
        for site in df.iterrows():
            lat2 = site[1].lat
            lon2 = site[1].lon
            coords2 = (lat2, lon2)
            dist = GPD.geodesic(coords1, coords2).km
            dists.append(dist)
        dist_cols[st] = min(dists)
    row_dists.append(dist_cols)
```

In [ ]:
```
rd = pd.DataFrame(row_dists)
rd.rename(lambda col: col.replace('/', ''), axis=1, inplace=True)
#rd.to_csv('data/distances.csv', index=False)
```

Note:
Access Points: Regional Access Points (RAPs) are an entry point to CEA. These entry points are resource centers where households experiencing homelessness can get help finding housing and other resources. Individuals and families experiencing homelessness may call ahead to schedule an appointment.
PSAP: A public-safety answering point (PSAP), sometimes called "public-safety access point" is a call center where emergency calls (like police, fire brigade, ambulance) initiated by any mobile or landline subscriber are terminated.