In [1]:

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
import pandas as pd
pd.options.mode.chained_assignment = None  # default='warn'
from datetime import datetime,date,timedelta
import pickle
from time import sleep
```

In [2]:

```
station = 'KCAWATSO86'
staname = 'watsonvil'
now = datetime.now().date()
sdate = datetime(now.year-2,now.month-5,now.day-5).date()
daterange = (now-sdate).days
print(f"# days between start date ({sdate}) and now ({now}): {daterange}")
```

```
# days between start date (2020-05-21) and now (2022-10-26): 888
```

In [3]:

```
def dailywind(d):
    wdate = d
    url = f"https://www.wunderground.com/dashboard/pws/{station}/table/{wda
    html = urlopen(url)
    soup = BeautifulSoup(html)

    headers = [th.getText() for th in soup.findAll('thead')[3].findAll('th'
    rows = soup.findAll('tbody')[3].findAll('tr')
    hourly_data = [[td.getText() for td in rows[i].findAll('td')] for i in

    df_weather = pd.DataFrame(hourly_data,columns=headers)
    df_wind = df_weather[['Time','Temperature','Wind','Speed','Gust']]
    df_wind['Date'] = wdate
    df_wind['Azimuth']= df_wind['Wind'].map(ordir)

    df_wind = df_wind[['Date','Time','Temperature','Wind','Azimuth','Speed'
    df_wind.columns = ['Date', 'Time','Temp','Wind','Azimuth', 'Speed', 'Gu

    return df_wind
```

In [4]:

```
def ploop(nsdate):
    print(f'function called from new start date: {nsdate}')
    nddiff=(now-nsdate).days
    for i in range(nddiff):
        weatherdate = nsdate + timedelta(days = i)
        print(f'pulling {weatherdate}')
        try:
            daysdata = dailywind(weatherdate)
        except:
            new_sdate = weatherdate
            new_ddiff = (now-new_sdate).days
            print(f"\nerror thrown attempting to pull: {new_sdate}")
            print(f"current function call progress: {i} ({round(100*i/dater
            print(f"cumulative progress: {len(ndf)} ({round(100*len(ndf)/da
            print(f"days remaining: {new_ddiff} ({round(100*new_ddiff/dater
            print("giving the server a 20 second break...")
            sleep(20)
            print("restarting pull loop!\n\n")
            ploop(new_sdate)
            break
        else:
            if len(daysdata) == 0:
                print(f'skipping null date: {weatherdate}')
            else:
                ndf.append(daysdata)
```

In [46]:

```
def dftostrFF(df):
    print(df.reset_index(drop=False).to_string(index=False))
def dftostrTT(df):
    print(df.reset_index(drop=True).to_string(index=True))
def dftostrTF(df):
    print(df.reset_index(drop=True).to_string(index=False))
def dftostrFT(df):
    print(df.reset_index(drop=False).to_string(index=True))
```

In [5]:

```
ordir={"North":0.0,"NNE":22.5,"NE": 45.0,"ENE":67.5,
       "East":90.0,"ESE":112.5,"SE":135.0,"SSE":157.5,
       "South":180,"SSW":202.5,"SW":225.0,"WSW":247.5,
       "West":270.0,"WNW":292.5,"NW":315.0,"NNW":337.5}
```

```
In [6]: ndf=[]
        ploop(sdate)
```

```
pulling 2020-07-15
pulling 2020-07-16
pulling 2020-07-17
pulling 2020-07-18
pulling 2020-07-19
pulling 2020-07-20
pulling 2020-07-21
pulling 2020-07-22
pulling 2020-07-23
pulling 2020-07-24
pulling 2020-07-25
pulling 2020-07-26
pulling 2020-07-27
pulling 2020-07-28
pulling 2020-07-29
pulling 2020-07-30
pulling 2020-07-31
pulling 2020-08-01
pulling 2020-08-02
pulling 2020-08-03
```

In [13]:

```
allwind = ndf[0]
for i in range(1,len(ndf)):
    allwind = pd.concat([allwind,ndf[i]],axis=0)
allwind['Speed'] = allwind['Speed'].str[:-5].astype(float)
allwind['Gust'][allwind['Gust']=='--'] = '0.0 °mph'
allwind['Gust'] = allwind['Gust'].str[:-5].astype(float)
allwind['Temp'] = allwind['Temp'].str[:-3].astype(float)
allwind['Month'] = pd.DatetimeIndex(allwind['Date']).month_name().str[:3]

#Mapping Datetimes to Different Periods Of The Day
allwind.loc[(allwind.Time.str.startswith(("10:","11:")) & allwind.Time.str.
            (allwind.Time.str.startswith(("12:","1:","2:","3:","4:","5:"))
            "Daypart"] = "Overnight"
allwind.loc[allwind.Time.str.startswith(("6:","7:","8:","9:")) & \
            allwind.Time.str.endswith("AM"), \
            "Daypart"] = "Morning"
allwind.loc[(allwind.Time.str.startswith(("10:","11:")) & allwind.Time.str.
            (allwind.Time.str.startswith(("12:","1:")) & allwind.Time.str.e
            "Daypart"] = "Midday"
allwind.loc[allwind.Time.str.startswith(("2:","3:","4:","5:")) & \
            allwind.Time.str.endswith("PM"), \
            "Daypart"] = "Afternoon"
allwind.loc[allwind.Time.str.startswith(("6:","7:","8:","9:")) & \
            allwind.Time.str.endswith("PM"), \
            "Daypart"] = "Evening"

#Mapping Datetimes to Seasons
allwind.loc[allwind.Month.isin(['Dec','Jan','Feb']) ,\
            "Season"] = "Winter"
allwind.loc[allwind.Month.isin(['Mar','Apr','May']) ,\
            "Season"] = "Spring"
allwind.loc[allwind.Month.isin(['Jun','Jul','Aug']) ,\
            "Season"] = "Summer"
allwind.loc[allwind.Month.isin(['Sep','Oct','Nov']) ,\
            "Season"] = "Fall"

#Rearranging Column Order
allwind = allwind[['Date','Time','Season','Month','Daypart','Temp','Wind','
```

In [104]:

```
#Finding Dates with Lowest Pct of 5-min Interval Records, And Lowest Pct of
dfcov = pd.DataFrame([[ndf[x].Date[0],\
               round(100*len(ndf[x])/288,1),\
               100-round(100*(ndf[x].Wind == '').sum()/len(ndf[x]),1)\
               ] for x in range(len(ndf))], columns=['date','pctrows','pctw

print('for days missing wind direction records, sort by % available records
baddates = dfcov.query('pctwind < 100').sort_values(['pctrows','pctwind'],a
dftostrTF(baddates.head(3))
```

```
for days missing wind direction records, sort by % available records, the
n % with wind direction:

      date  pctrows  pctwind
2022-02-01     13.9     97.5
2021-05-20     84.4     99.6
2021-02-25     85.1     99.2
```

In [121]:

```
print('day with many records missing, as well as no wind direction for some
allwind[allwind.Date==list(baddates.date)[0]]
```

```
day with many records missing, as well as no wind direction for some reco
rds:
```

Out[121]:

| | Date | Time | Season | Month | Daypart | Temp | Wind | Azimuth | Speed | Gust |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2022-02-01 | 8:44 PM | Winter | Feb | Evening | 51.1 | ENE | 67.5 | 0.7 | 1.0 |
| **1** | 2022-02-01 | 8:49 PM | Winter | Feb | Evening | 50.7 | NE | 45.0 | 0.6 | 1.2 |
| **2** | 2022-02-01 | 8:54 PM | Winter | Feb | Evening | 50.5 | NE | 45.0 | 0.3 | 0.8 |
| **3** | 2022-02-01 | 8:59 PM | Winter | Feb | Evening | 49.2 | NNE | 22.5 | 0.0 | 0.1 |
| **4** | 2022-02-01 | 9:04 PM | Winter | Feb | Evening | 47.6 | NNE | 22.5 | 0.3 | 0.6 |
| **5** | 2022-02-01 | 9:09 PM | Winter | Feb | Evening | 46.7 | ENE | 67.5 | 0.0 | 0.0 |
| **6** | 2022-02-01 | 9:14 PM | Winter | Feb | Evening | 47.1 | ENE | 67.5 | 0.1 | 0.2 |
| **7** | 2022-02-01 | 9:19 PM | Winter | Feb | Evening | 46.1 | North | 0.0 | 0.3 | 0.6 |
| **8** | 2022-02-01 | 9:24 PM | Winter | Feb | Evening | 45.0 | North | 0.0 | 0.4 | 0.9 |
| **9** | 2022-02-01 | 9:29 PM | Winter | Feb | Evening | 44.5 | ENE | 67.5 | 0.0 | 0.0 |
| **10** | 2022-02-01 | 9:34 PM | Winter | Feb | Evening | 44.4 | NE | 45.0 | 0.4 | 0.8 |
| **11** | 2022-02-01 | 9:39 PM | Winter | Feb | Evening | 43.7 | NE | 45.0 | 0.1 | 0.3 |
| **12** | 2022-02-01 | 9:44 PM | Winter | Feb | Evening | 44.1 | NE | 45.0 | 0.2 | 0.2 |
| **13** | 2022-02-01 | 9:49 PM | Winter | Feb | Evening | 44.0 | East | 90.0 | 0.0 | 0.0 |
| **14** | 2022-02-01 | 9:54 PM | Winter | Feb | Evening | 44.3 | NW | 315.0 | 0.0 | 0.1 |
| **15** | 2022-02-01 | 9:59 PM | Winter | Feb | Evening | 43.2 | ENE | 67.5 | 0.4 | 0.8 |
| **16** | 2022-02-01 | 10:04 PM | Winter | Feb | Overnight | 41.8 | | NaN | 0.2 | 0.5 |
| **17** | 2022-02-01 | 10:09 PM | Winter | Feb | Overnight | 41.6 | East | 90.0 | 0.0 | 0.0 |
| **18** | 2022-02-01 | 10:14 PM | Winter | Feb | Overnight | 41.6 | NE | 45.0 | 0.0 | 0.0 |
| **19** | 2022-02-01 | 10:19 PM | Winter | Feb | Overnight | 41.4 | NE | 45.0 | 0.7 | 1.0 |
| **20** | 2022-02-01 | 10:24 PM | Winter | Feb | Overnight | 40.8 | SSE | 157.5 | 0.0 | 0.0 |
| **21** | 2022-02-01 | 10:29 PM | Winter | Feb | Overnight | 40.5 | NE | 45.0 | 0.3 | 0.5 |
| **22** | 2022-02-01 | 10:34 PM | Winter | Feb | Overnight | 40.1 | NE | 45.0 | 0.6 | 0.9 |
| **23** | 2022-02-01 | 10:39 PM | Winter | Feb | Overnight | 39.8 | NNE | 22.5 | 0.0 | 0.1 |
| **24** | 2022-02-01 | 10:44 PM | Winter | Feb | Overnight | 39.8 | ENE | 67.5 | 1.3 | 1.6 |
| **25** | 2022-02-01 | 10:49 PM | Winter | Feb | Overnight | 39.0 | ENE | 67.5 | 0.0 | 0.1 |
| **26** | 2022-02-01 | 10:54 PM | Winter | Feb | Overnight | 39.6 | ENE | 67.5 | 0.7 | 0.9 |
| **27** | 2022-02-01 | 10:59 PM | Winter | Feb | Overnight | 39.4 | ENE | 67.5 | 0.0 | 0.0 |
| **28** | 2022-02-01 | 11:04 PM | Winter | Feb | Overnight | 39.7 | NE | 45.0 | 0.8 | 1.3 |
| **29** | 2022-02-01 | 11:09 PM | Winter | Feb | Overnight | 39.5 | ESE | 112.5 | 0.3 | 0.5 |

| | Date | Time | Season | Month | Daypart | Temp | Wind | Azimuth | Speed | Gust |
|---|---|---|---|---|---|---|---|---|---|---|
| **30** | 2022-02-01 | 11:14 PM | Winter | Feb | Overnight | 38.4 | ENE | 67.5 | 0.5 | 0.9 |
| **31** | 2022-02-01 | 11:19 PM | Winter | Feb | Overnight | 39.6 | ENE | 67.5 | 3.3 | 4.6 |
| **32** | 2022-02-01 | 11:24 PM | Winter | Feb | Overnight | 41.0 | NE | 45.0 | 1.4 | 2.0 |
| **33** | 2022-02-01 | 11:29 PM | Winter | Feb | Overnight | 39.3 | ENE | 67.5 | 1.6 | 2.1 |
| **34** | 2022-02-01 | 11:34 PM | Winter | Feb | Overnight | 39.2 | ENE | 67.5 | 0.4 | 0.7 |
| **35** | 2022-02-01 | 11:39 PM | Winter | Feb | Overnight | 39.4 | NE | 45.0 | 0.3 | 0.6 |
| **36** | 2022-02-01 | 11:44 PM | Winter | Feb | Overnight | 38.9 | NNE | 22.5 | 0.2 | 0.6 |
| **37** | 2022-02-01 | 11:49 PM | Winter | Feb | Overnight | 39.0 | East | 90.0 | 0.3 | 0.7 |
| **38** | 2022-02-01 | 11:54 PM | Winter | Feb | Overnight | 39.2 | NNW | 337.5 | 0.8 | 1.3 |
| **39** | 2022-02-01 | 11:59 PM | Winter | Feb | Overnight | 38.7 | NE | 45.0 | 1.4 | 2.0 |

In [79]:

```
allwind.to_pickle(f'{staname}_raw_{sdate}_{now-timedelta(days=1)}.pkl')
allwind.to_csv(f'{staname}_raw_{sdate}_{now-timedelta(days=1)}.csv')
```

In [122]:

```
dfcov.to_pickle(f'{staname}_coverage_{sdate}_{now-timedelta(days=1)}.pkl')
dfcov.to_csv(f'{staname}_coverage_{sdate}_{now-timedelta(days=1)}.csv')
```