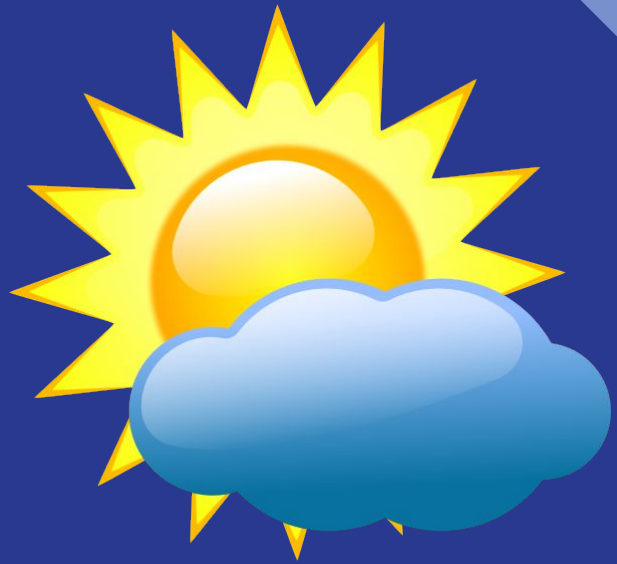
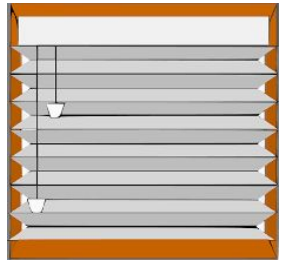
MAD2502 Capstone: Weather Data Collection & Weather App

Gavin Sidhu, Matthew Peck, Tan Hollman



Approach

- Python program that can be used like a normal weather app
  - However, countless websites and apps already exist for this purpose
- Can also be used to automate everyday processes
  - Automatically adjusting thermostat or AC
  - Window shades that close when it’s sunny, open when it’s cold
  - Automated sprinklers that don’t run when if rain is near
  - Porch light that turns on when it’s cloudy



Libraries Used

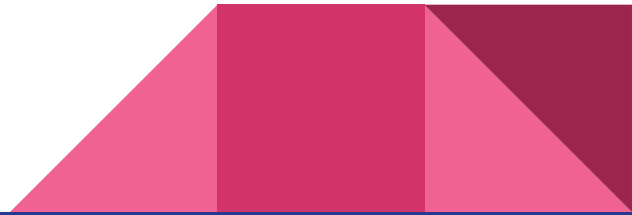
Requests: Allows us to make requests from websites to get their data

JSON: Allows us to convert website data into a JSON format, which can be read like a Python dictionary

Datetime: Allows us to assign and manipulate dates and times

GeoPy: Allows us to take a city, state and convert it to latitude and longitude

BeautifulSoup: Allows us to pull data out of HTML code



Geolocation

- Since OpenWeatherMap only takes latitude and longitude, we needed a way to take the user's city and state and geolocate it

- For this, we used a Python library called GeoPy, which uses open-source geocoding APIs including Google Maps to find the latitude and longitude of a given location

```
city = str(input("Please enter 'City, State': "))

geolocator = Nominatim(user_agent="weather_app")
location = geolocator.geocode(city)
latitude = location.latitude
longitude = location.longitude
```

```
Please enter 'City, State': Gainesville, fl
The latitude of this location is 29.6519684
The longitude of this location is -82.3249846
```

OpenWeatherMap API

- Open-source API that sends JSON data about past, current, and future weather
- Requires an API key obtained through signing up on their website
- “Free Plan” Limitations:
  - History: up to 5 days in the past
  - Forecasts: up to 7 days in the future

```
forecast_url = f"https://api.openweathermap.org/data/2.5/onecall?lat={latitude}&lon={longitude}&exclude=minutely,hourly,current&appid={api_key}&units=imperial"
```

Collecting JSON data

```
response = requests.get(forecast_url)

response_text = response.text

forecast_json = json.loads(response_text)
```

- Requests data from OpenWeatherMap API via URL
- Converts response into JSON format
- Data can be interpreted as lists and dictionaries
    - E.g. weather in “days” number of days can be found with `forecast_json["daily"][days]["weather"][0]["description"]`

Website Response

After JSON conversion


```
{
  "dt": 1648141200,
  "sunrise": 1648121296,
  "sunset": 1648165362,
  "moonrise": 1648100640,
  "moonset": 1648137600,
  "moon_phase": 0.73,
  "temp": {
    "day": 59.81,
    "min": 55.33,
    "max": 68.32,
    "night": 55.33,
    "eve": 58.87,
    "morn": 65.19
  },
  "feels_like": {
    "day": 59.81,
    "night": 54.84,
    "eve": 58.73,
    "morn": 65.55
  },
  "pressure": 1016,
  "humidity": 92,
  "dew_point": 57.63,
  "wind_speed": 9.51,
  "wind_deg": 299,
  "wind_gust": 21.72,
  "weather": [
    {
      "id": 501,
      "main": "Rain",
      "description": "moderate rain",
      "icon": "10d"
    }
  ],
  "clouds": 100,
  "pop": 1,
  "rain": 7.72,
  "uvi": 1.01
},
```

Time Conversions

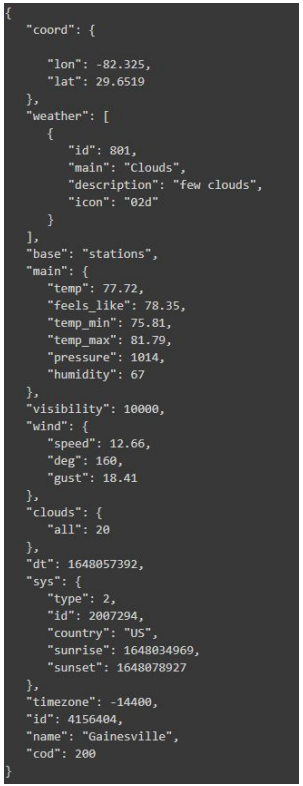
```
"sunrise": 1648034969,
"sunset": 1648078927
```

- OpenWeather interprets time and dates in unix time, which is the number of seconds since Jan. 1, 1970
- For outputs involving time such as sunrise and sunset times, we used the datetime library to convert from unix time to standard date and time

```
def unix_to_datetime(unix_time): # converts from unix time to standard date and time

    standard_date = (dt.fromtimestamp(unix_time)).strftime('%m-%d-%Y %H:%M:%S')

    return standard_date
```

```
elif data == "sunrise":
    return unix_to_datetime(forecast_json["daily"][days]["sunrise"]) # sunrise time
elif data == "sunset":
    return unix_to_datetime(forecast_json["daily"][days]["sunset"]) # sunset time
```

Example of All Data for Current Weather



```
{
   "coord": {

      "lon": -82.325,
      "lat": 29.6519
   },
   "weather": [
      {
         "id": 801,
         "main": "Clouds",
         "description": "few clouds",
         "icon": "02d"
      }
   ],
   "base": "stations",
   "main": {
      "temp": 77.72,
      "feels_like": 78.35,
      "temp_min": 75.81,
      "temp_max": 81.79,
      "pressure": 1014,
      "humidity": 67
   },
   "visibility": 10000,
   "wind": {
      "speed": 12.66,
      "deg": 160,
      "gust": 18.41
   },
   "clouds": {
      "all": 20
   },
   "dt": 1648057392,
   "sys": {
      "type": 2,
      "id": 2007294,
      "country": "US",
      "sunrise": 1648034969,
      "sunset": 1648078927
   },
   "timezone": -14400,
   "id": 4156404,
   "name": "Gainesville",
   "cod": 200
}
```

```
if data == "temp":
    return current_json["main"]["temp"]
elif data == "temp_low":
    return current_json["main"]["temp_min"]
elif data == "temp_high":
    return current_json["main"]["temp_max"]
elif data == "feels_like":
    return current_json["main"]["feels_like"]
elif data == "weather":
    return current_json["weather"][0]["description"]
elif data == "humidity":
    return current_json["main"]["humidity"]
elif data == "wind_speed":
    return current_json["wind"]["speed"]
elif data == "wind_direction":
    if current_json["wind"]["deg"] < 90:
        return "Northeast"
    elif 90 <= current_json["wind"]["deg"] < 180:
        return "Southeast"
    elif 180 <= current_json["wind"]["deg"] < 270:
        return "Southwest"
    elif 270 <= current_json["wind"]["deg"] < 360:
        return "Northwest"
```

OpenWeather Functions & Their Capabilities

```
def current(latitude, longitude, data):
```

```
def forecast(latitude, longitude, data, days):
```

```
def history(latitude, longitude, data, date):
```

- Current
  - Gives data for weather at this exact moment
- Forecast
  - Gives data for weather in the future, including today, up to a week from now
- History
  - Gives weather history up to 5 days prior

	Current	Forecast	History
Temp	Yes	Yes	Yes
Temp_Low	Yes	Yes	
Temp_High	Yes	Yes	
Feels_Like	Yes	Yes	
Weather	Yes	Yes	Yes
Humidity	Yes	Yes	Yes
Wind_speed	Yes	Yes	Yes
Wind_DIrection	Yes	Yes	Yes
Weekly High		Yes	
Weekly Low		Yes	
Weekly Average		Yes	
Temp Average		Yes	
Sunrise		Yes	Yes
Sunset		Yes	Yes
Rain		Yes	
Snow		Yes	

Web Scraping HTML Code

Due to “Free Plan” limitations from OpenWeather, we web scraped from The Old Farmer’s Almanac after inputting a “City, State” combination.

We were able to obtain the temperature data for any day in the past including the average, minimum, and maximum temperatures.

```
on_this_day_url = f"https://www.almanac.com/weather/history/{state}/{city}/{year}-{month}-{day}"

response = requests.get(on_this_day_url)

response_content = response.content

html = BeautifulSoup(response_content, 'html.parser')
```

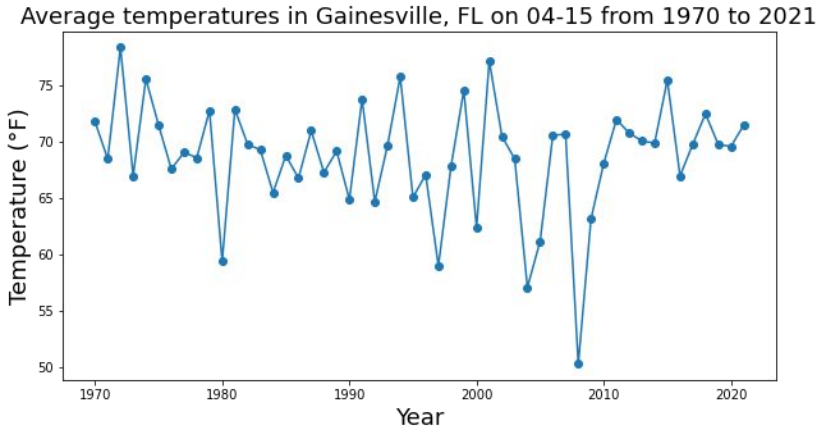
```
temp_average = html.find(class_ = "weatherhistory_results_datavalue temp").find(class_ = "value").get_text()

temp_min = html.find(class_ = "weatherhistory_results_datavalue temp_mn").find(class_ = "value").get_text()

temp_max = html.find(class_ = "weatherhistory_results_datavalue temp_mx").find(class_ = "value").get_text()

temp_units = (html.find(class_ = "weatherhistory_results_datavalue temp_mn").find(class_ = "units").get_text())[1]
```

Almanac Functions & Historic Data

- On This Day
  - Gives either the average, lowest, or highest temperature for a certain day back to 1970
  - Also returns a graph of that statistic for every year
- Compare Today
  - This function compares today's average temperature to the historic average
  - Example: compare_today()

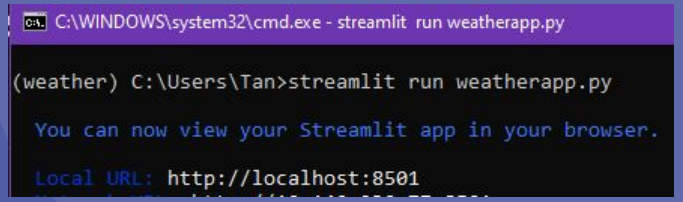
Example: onthisday("Gainesville, FL", "average", "04-15")

```
68.69
```

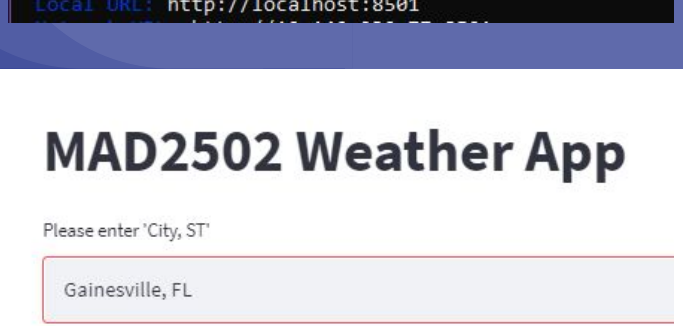



```
Today's temperature of 75.59°F is hotter than the historic average of 68.69°F
```

Streamlit



- Streamlit is a Python library that allows one to create apps and GUIs for data-related projects
- We wanted to keep it simple and just use Python for this project
- The other potential libraries were more complex and had more limitations




```
st.header("Expand the tabs below for more details!")
temperature = st.expander("Temperature")
temperature.subheader("Current temperature: " + str(
temperature.subheader("High: " + str(current(latitude
temperature.subheader("Low: " + str(current(latitude,
temperature.subheader("Humidity: " + str(current(lati
temperature.subheader("Feels like: " + str(current(la
weather = st.expander("Weather/Wind")
```

Streamlit (continued)

MAD2502 Weather App

Now displaying the weekly forecast for Gainesville, FL!

04/15	04/16	04/17	04/18	04/19	04/20	04/21
83°	90°	87°	83°	80°	81°	83°
65°	64°	66°	64°	57°	56°	59°

Expand the tabs below for more details!

- Temperature +
- Weather/Wind +
- Sunset/sunrise times +
- Weekly forecast +

Expand the tabs below for more details!

Temperature −

Current temperature: 68.52°F

High: 70.09°F

Low: 66.67°F

Humidity: 88%

Feels like: 69.21°F

Weather/Wind −

The current weather is: overcast clouds

Wind speed: 2.89

Wind direction: Northwest

Rain: 0.61 mm

Snow: 0 mm

Sunset/sunrise times −

Sunrise time: 04-15-2022 07:02:33

Sunset time: 04-15-2022 19:55:38

Tomorrow's sunrise time: 04-16-2022 07:01:28

Weekly forecast −

Weekly average: 83.58°F

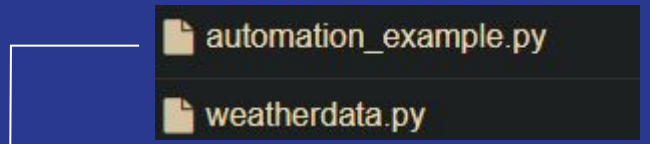
Weekly high: 90.03°F

Weekly low: 56.39°F

Automating Processes with Weather Data

- Since weather can change our daily routines, it would be nice to have something that can automate different processes for you depending on the weather

- Using a Raspberry Pi would be an easy way to automate physical tasks in your home, like closing blinds or adjusting the thermostat




```
city = "Gainesville, FL"

import weatherdata
from geopy.geocoders import Nominatim
import time

geolocator = Nominatim(user_agent="weather_app")
location = geolocator.geocode(city)
latitude = location.latitude
longitude = location.longitude

starttime = time.time()

while True:
    print(weatherdata.current(latitude, longitude, "temp"))
    print(weatherdata.unix_to_datetime(time.time()))
    if weatherdata.current(latitude, longitude, "temp") >= 80:
        print("Close blinds") # insert function to close blinds here
    else:
        print("Open blinds") # insert function to open blinds here
    print("-----------")
    time.sleep(300 - ((time.time() - starttime) % 300))
```

```
80.44
04-14-2022 18:51:48
Close blinds
-----------
80.33
04-14-2022 18:56:48
Close blinds
-----------
80.33
04-14-2022 19:01:48
Close blinds
-----------
80.33
04-14-2022 19:06:48
Close blinds
-----------
79.83
04-14-2022 19:11:48
Open blinds
-----------
```