My wife and I have been house hunting for several months, and the mortgage rate market in the United States has been up and down. I found myself looking at rates from local lenders every day, navigating to websites and writing the rates down or mentally taking note of where they are at. It was difficult to see trends and understand if rates were rising or falling over time.

I decided to try writing a script that would retrieve the rates from the lender website and save it for me to track. My plan was to build a short Python script that would scrape the website and save the mortgage rates in an existing MySQL database I own. Once I had the data stored there, I would create a small dashboard in Metabase to view the day's rate and simple trends over time.

Writing the Script

I was familiar with the Beautiful Soup package for Python, but hadn't used it aside from a small project when I first learned Python. It was easy follow the documentation and fetch HTML from the web, but parsing the values wasn't going well for me. I was trying to extract a few values from a table, and having difficulty getting a table cell's value from multiple tables on the webpage.

The approach the worked for me was to capture all the `td` values using the `.findall()` method and parse the values I was after using substrings. This worked for me in the moment, but if the design of the webpage changes or the format of the table is adjusted my script may not parse the correct values from the table. I might revisit this part of the script in the future as I learn more about Beautiful Soup and parsing HTML.

```
# Fetch mortgage rates
URL = "https://www.wingsfinancial.com/rates/mortgages-purchase"
page = requests.get(URL)

soup = BeautifulSoup(page.content, "html.parser")
table = soup.find_all('td')

fr30 = float(str(table[0])[4:9])
fr30pt = float(str(table[2])[4:6])
ARM71 = float(str(table[25])[4:9])
ARM71pt = float(str(table[7])[4:6])
```

Now that I had the values, I needed to write them to the MySQL database I use for hobby projects. Since, I had done this in a previous project using the `mysql.connector` package, I was able to quickly refresh my understanding and write the rest of the script to write the values to a table. If you want to learn more, MySQL has good documentation and there's lots of YouTube tutorials out there.

```
# Create variables
source = "Wings"
ct = datetime.datetime.now()

# Connect to database
db = mysql.connector.connect(
    host = config.host,
    user = config.user,
    password = config.pw,
    database = config.db
)
cursor = db.cursor()

# Write to database
sql = "INSERT INTO `daily_rates` (source, timestamp, 30_year_fixed_rate,
30_year_fixed_points, 71_arm_rate,71_arm_point) VALUES (%s,%s,%s,%s,%s,%s)"
val = (source,ct,fr30,fr30pt,ARM71,ARM71pt)
cursor.execute(sql,val)

db.commit()

# Close database connection
cursor.close()
db.close()
```

I committed the final version of the script to GitHub and felt pretty accomplished. All that was left was to schedule the script to run daily!

Scheduling with cron

In the past, I have used a `time.sleep` function within the Python script to "schedule" a run, essentially only delaying a loop from running until a certain amount of time had elapsed. I realized this wasn't a good approach since my Python script never stopped running, using some system resources. I didn't need to run a script 24/7 when it only needed to execute once per day. Since the MySQL database I use is hosted on a Linux server, I decided to use `cron` to schedule my script to run on the server daily at noon central time. This was my first time using `cron` and I found lots of guides on setting up the `crontab` file. After cloning my git project to the server's `/root` directory, I set up the `crontab` file. I wanted to run the python script Monday - Friday at noon so my entry into the `crontab` file was:

```
0 12 * * 1-5 /usr/bin/python3 /root/mortgage-rate-scraper/script.py
```

I was so excited to see the results of my first run, but they were written to MySQL at 6am instead of noon. Timezones! I hadn't considered that the linux server I scheduled the job on was on UTC time. A simple adjustment to the `crontab` entry to account for the 6 hour offset and I was all set.

```
0 18 * * 1-5 /usr/bin/python3 /root/mortgage-rate-scraper/script.py
```

Recently, I learned the joys of Daylight Savings Time as my script was "delayed" by an hour. Another easy fix to the `crontab` file and we were back in business. Time logic is so fun!

Viewing the Results

The last step in this project was to create a user facing dashboard where I could see today's rates and a simple trend of rates over time. For most of my personal projects, I like to use Metabase for this kind of reporting. It's simple to configure, lightweight, and open source. Metabase also offers an embedding option that I'm going to try out for the first time in this post!

{{< iframe "http://metabase.corvidanalytics.com/public/dashboard/8eef25a5-d8a4-4033-96e6-b828af13b025" 800 600 >}}

Here's a link to the GitHub repository for this project