From January 6th to 19th, I participated in the Tech Academy's Python Live Project. The goal of the project was to develop a web app using the Django Framework that helped users explore their interests around a specific hobby (I chose Anime Film). Participating in this project I learned how to use an API, a database with Models, and the Django Template Language among other tools in order to spin up a working application. On the Project Management Board I completed 8 Stories total (including the Prep Story). Along the way I also participated in a series of standup meetings in practice of the SCRUM process. Here is the summary of my learning experience proceeding through each story and the SCRUM process.

Prep Story: Planning

Here I gained experience in planning for a project by developing a concept for an App (Tracking Anime Movies). I picked an API (iMDB Alternative) with a vast collection of data on movies. I decided that one part of my app would be used for keeping track of movies you already have access to and another part would be used to search for movies you were curious about.

Story 1: Build the Basic App

In this story I learned how to set up a Django Web App and connect the various moving parts that allow you to render a single page effectively (Django Template Language). I registered the App in the main settings, extended the home page from the base template and wrote a views function to render the home page also connecting it through the URL's file. Definitely a lot to keep track of, I learned that sketching everything out on paper really helps.

Story 2: Creating the Model

In this story I devised a relevant set of data useful for organizing a collection of movies. I created a movie object, a model form and a views function to render it in a user friendly interface.

Story 3: Index Page

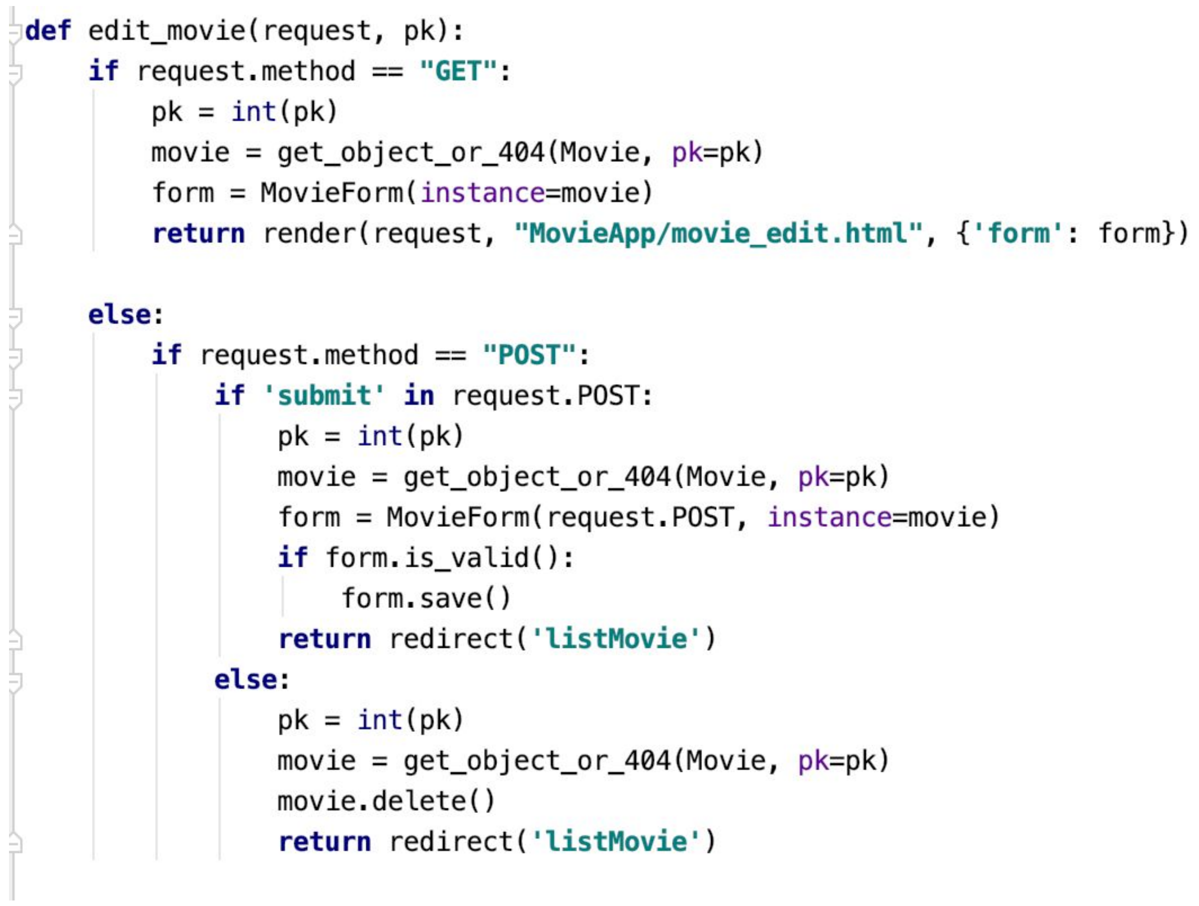
Here I learned how to display the collection of movies with a Model Manager. Using the Manager I retrieved all the movies from my database and using DTL variables I displayed them.

Story 4: Details Page

I created a separate details page for displaying the Details of a single movie in a more presentable manner. I used the same techniques from the previous story (Model Manager with DTL variables) also I learned how to use the pk (primary key) as a variable in the URL in order to tailor the Details Page for the specific movie selected.

<u>Story 5: Edit and Delete Functions</u>

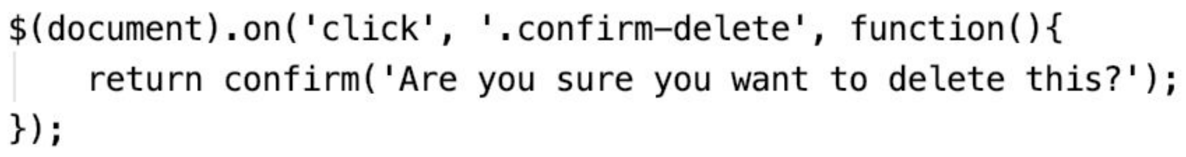
Things got juicy in this story. I created a form that allows you to delete any individual movie as well as edit/update any info pertaining to it. I was able to implement edit and delete functionality in the same template by utilizing logic in the views function (See Below). I also learned how to use Bootstrap's Crispy Forms to present a more sleek looking interface. One challenge I had was using Javascript to implement a confirmation alert for the Delete button (Also Below).



```python
def edit_movie(request, pk):
    if request.method == "GET":
        pk = int(pk)
        movie = get_object_or_404(Movie, pk=pk)
        form = MovieForm(instance=movie)
        return render(request, "MovieApp/movie_edit.html", {'form': form})

    else:
        if request.method == "POST":
            if 'submit' in request.POST:
                pk = int(pk)
                movie = get_object_or_404(Movie, pk=pk)
                form = MovieForm(request.POST, instance=movie)
                if form.is_valid():
                    form.save()
                return redirect('listMovie')
            else:
                pk = int(pk)
                movie = get_object_or_404(Movie, pk=pk)
                movie.delete()
                return redirect('listMovie')
```

Javascript:



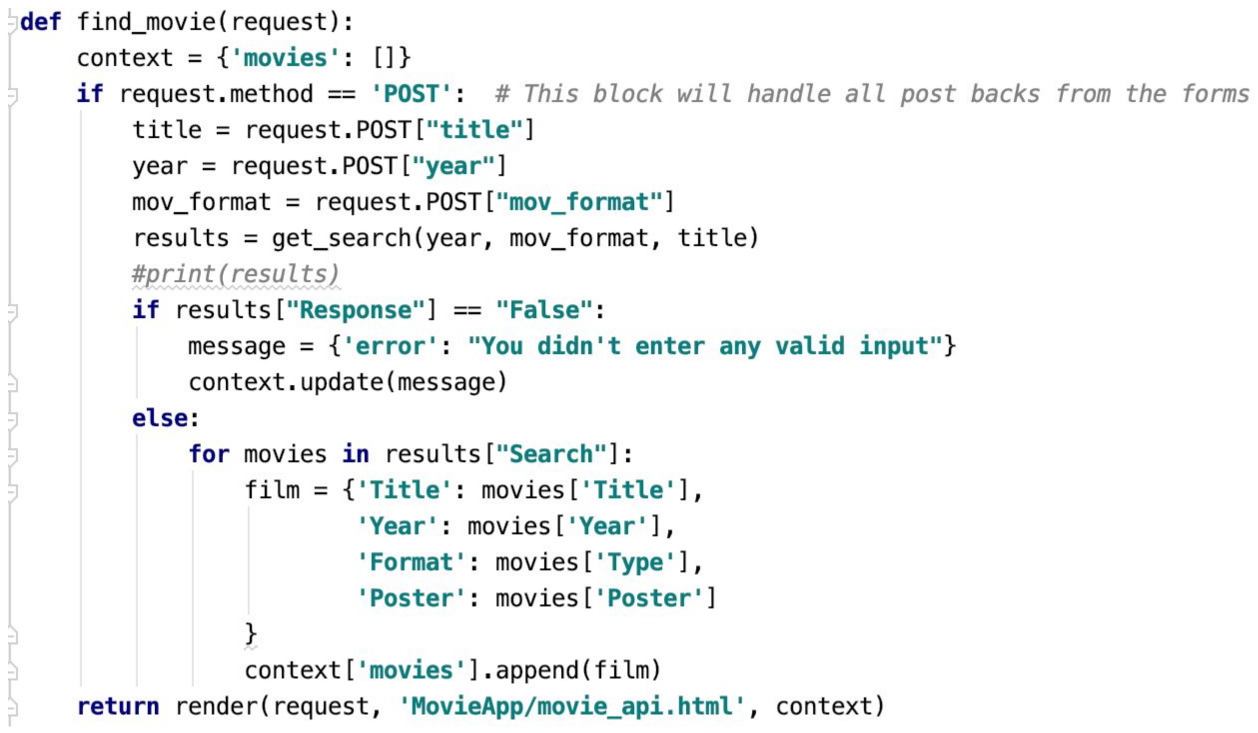
```javascript
$(document).on('click', '.confirm-delete', function(){
    return confirm('Are you sure you want to delete this?');
});
```

<u>Story 6A: Connect to API</u>

I was especially interested in this story, which had prompted me to connect to my API. Here I learned how to write a function for connecting to my API and assign variables to it’s parameters in order to give the user access to them through a template form.

<u>Story 7A: Parse through JSON</u>

Here I learned about JSON responses and how to utilize them in order to display relevant data to the user. I chose to display the data below the form that connects to the API, so that the user can conveniently execute another search. A challenge was getting the page to display an error message when a user inputs invalid information. I solved this in the logic of the views function by probing for a False “Response” from the JSON response (See Below).



```python
def find_movie(request):
    context = {'movies': []}
    if request.method == 'POST':  # This block will handle all post backs from the forms
        title = request.POST["title"]
        year = request.POST["year"]
        mov_format = request.POST["mov_format"]
        results = get_search(year, mov_format, title)
        #print(results)
        if results["Response"] == "False":
            message = {'error': "You didn't enter any valid input"}
            context.update(message)
        else:
            for movies in results["Search"]:
                film = {'Title': movies['Title'],
                        'Year': movies['Year'],
                        'Format': movies['Type'],
                        'Poster': movies['Poster']
                }
                context['movies'].append(film)
    return render(request, 'MovieApp/movie_api.html', context)
```

Scrum

Monday to Friday throughout the two week sprint I participated in a daily standup meeting as a part of the SCRUM Process. Every morning we met for about 10 minutes and discussed what we completed the day before along with the challenges we're currently facing. Additionally, on Fridays we did a Code Retrospective reflecting on what practices proved effective along with what could be improved. Most of the value I gained from these meetings was in the accountability that came with reporting my progress every morning. It was never a good feeling to report that I had only completed a couple aspects of one story rather than the whole task at hand. This motivated me to be more productive in the long haul. Additionally, I found the code retrospectives to be particularly helpful in prompting me to reflect on and improve my coding process. I realized that while researching, I would often delve into "rabbit holes" beyond the depth of knowledge required for the task at hand. While this was informative and possibly benefited my programming career in the long run, this practice impeded my efficiency in the sprint. Overall, I learned a ton throughout the Live Project process and am proud of the App I made.