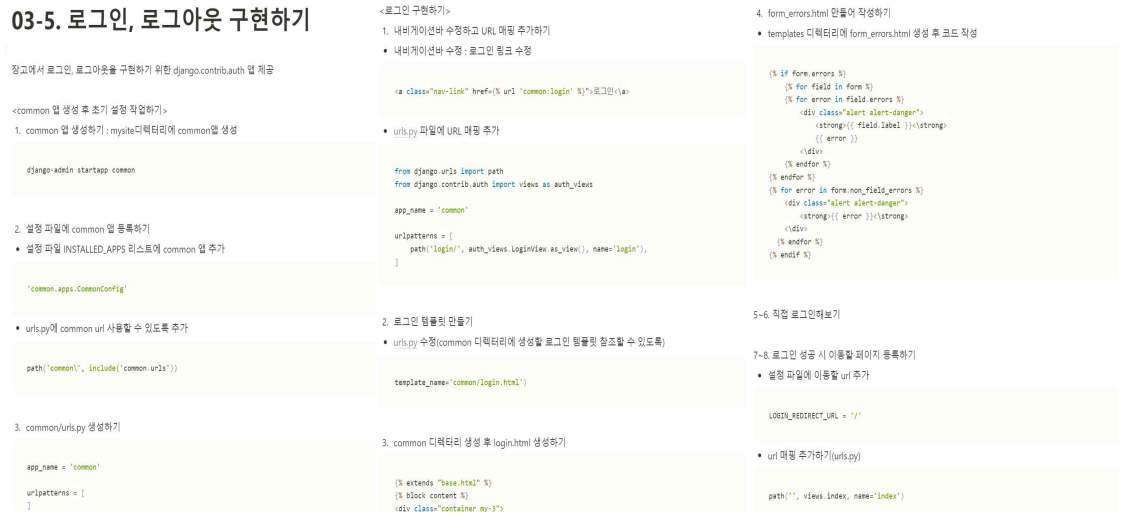
<유용한 파이썬 패키지 조사>

미디어커뮤니케이션학과 2020114614 신채원

1. django(장고)

파이썬에서 많이 쓰이는 웹 프레임워크. 장고에 있는 기능들을 활용해 웹페이지를 만들 수 있다.

예시 코드: 장고를 활용한 웹페이지 로그인/로그아웃 구현(예전에 본인이 했던 노션 필기본 캡처)



03-5. 로그인, 로그아웃 구현하기

장고에서 로그인, 로그아웃을 구현하기 위한 django.contrib.auth 앱 제공

<common 앱 생성 후 초기 설정 작업하기>

1. common 앱 생성하기 : mysite디렉터리에 common앱 생성

```
django-admin startapp common
```

2. 설정 파일에 common 앱 등록하기

- 설정 파일 INSTALLED_APPS 리스트에 common 앱 추가

```
'common.apps.CommonConfig'
```

- urls.py에 common url 사용할 수 있도록 추가

```
path('common\', include('common.urls'))
```

3. common/urls.py 생성하기

```
app_name = 'common'

urlpatterns = [
]
```

<로그인 구현하기>

1. 내비게이션바 수정하고 URL 매핑 추가하기

- 내비게이션바 수정 : 로그인 링크 수정

```
<a class="nav-link" href={% url 'common:login' %}>로그인</a>
```

- urls.py 파일에 URL 매핑 추가

```
from django.urls import path
from django.contrib.auth import views as auth_views

app_name = 'common'

urlpatterns = [
    path('login/', auth_views.LoginView.as_view(), name='login'),
]
```

2. 로그인 템플릿 만들기

- urls.py 수정(common 디렉터리에 생성할 로그인 템플릿 참조할 수 있도록)

```
template_name='common/login.html')
```

3. common 디렉터리 생성 후 login.html 생성하기

```
{% extends "base.html" %}
{% block content %}
<div class="container my-3">
```

4. form_errors.html 만들어 작성하기

- templates 디렉터리에 form_errors.html 생성 후 코드 작성

```
{% if form.errors %}
    {% for field in form %}
    {% for error in field.errors %}
        <div class="alert alert-danger">
            <strong>{{ field.label }}</strong>
            {{ error }}
        </div>
    {% endfor %}
{% endfor %}
{% for error in form.non_field_errors %}
    <div class="alert alert-danger">
        <strong>{{ error }}</strong>
    </div>
 {% endfor %}
{% endif %}
```

5~6. 직접 로그인해보기

7~8. 로그인 성공 시 이동할 페이지 등록하기

- 설정 파일에 이동할 url 추가

```
LOGIN_REDIRECT_URL = '/'
```

- url 매핑 추가하기(urls.py)

```
path('', views.index, name='index')
```

더불어 내가 파이썬으로 구현해보고 싶은 것 또한 이것과 관련되어 있다. 장고 책을 한 권 떼면서 웹페이지 하나를 만들어 보았는데, 그동안 C언어나 자바 문법을 공부하면서 내가 별 찍기를 하고, 스택에 데이터를 쌓을 줄 앎으로써 실제로 구현할 수 있는 게 무엇일까 하는 생각들이 들었었는데, 장고를 활용해 웹페이지 만들기를 하니 내가 짜는 코드들의 결과물들이 바로바로 보여서 성취감이 들었다. 아직 초보라 엄청 예쁜 웹페이지를 만들지는 못하지만, 이후에 나올 파이썬의 대쉬 패키지나 자바스크립트 언어를 더 배워서 예쁜 디자인들이 많이 들어간 내 웹페이지를 만들어 보고 싶다.

2. Pillow(필로우)

이미지 라이브러리이다. 썸네일 제작도 가능하며, 파일 형식을 변경하거나 필터 적용도 할 수 있고, 해당 이미지 파일을 띄워서 볼 수도 있다. 특히 많은 이미지 파일들을 한꺼번에 동일 변경 적용할 때 유용하게 쓰이는 패키지이다.

예시 코드(이미지 파일 띄우기):

```
from PIL import Image

im = Image.open("kittems.jpg")
im.show()
print(im.format, im.size, im.mode)
```

3. Dash(대쉬)

최신 라이브러리로, 순수 파이썬에서만 구동되는 데이터 시각화에 특화되어 있는 앱이다. Flask나 Plotly.js, React.js와 함께 사용하는데, 따로 자바스크립트 구문을 넣지 않고도 원하는 것들을 대시 보드로 구성할 수 있다.

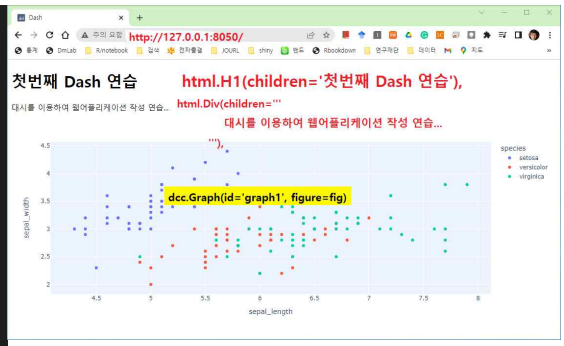
예시 코드(대시를 활용해 웹 어플리케이션에 그래프 띄우기):

```
from dash import Dash, html, dcc
import plotly.express as px
import pandas as pd

df = px.data.iris()
fig = px.scatter(df, x="sepal_length", y="sepal_width",
                 color="species")

app = Dash(__name__)
app.layout = html.Div(children=[
    html.H1(children='첫번째 Dash 연습'),
    html.Div(children='''
        대시를 이용하여 웹어플리케이션 작성 연습...
    '''),
    dcc.Graph(
        id='graph1',
        figure=fig
    )
])

if __name__ == '__main__':
    app.run_server(debug=True)
```



4. zipfile(파일 압축)

어떤 파일들을 압축하거나, 그 압축파일을 다시 해제하거나 파일들의 정보를 읽는 등 압축 파일에 관한 전반에 대해 다양하게 활용되는 함수이다.

예시 코드: 개별 파일 압축하기(주의점: ZipFile함수 인자에 작업 경로를 넣어 버리면 파일과 함께 작업 경로까지 압축해버리기 때문에, 작업디렉터리를 이동해서 작업하는 것이 좋다고 한다.)

```
os.chdir("C:/Users/User/Desktop/")

my_zip = zipfile.ZipFile("test_zip.zip", 'w')
my_zip.write('test1.txt')
my_zip.close()
```

5. Numpy(넘파이)

과학 계산을 위한 라이브러리로서, 행렬/배열 처리 및 연산을 수행하고 난수를 생성하는 기능도 가지고 있다.

예시 코드(넘파이를 이용해 여러 가지 배열 생성하기):

```
#리스트에서 행렬/배열 생성
import numpy as np
a = [[1,2,3], [4,5,6]]
b = np.array(a)
print(b)

#2 특수한 배열의 생성
print(np.arrange(10)) #1씩 증가하는 1차원 배열
print(np.arrange(5,10)) #시작이 5부터
print(np.zeros((2,2)))  #영행렬
print(np.full((2,3),5)) #모든 원소가 5인 2*3행렬
print(np.eye(3)) #단위행렬

#3. 배열의 차원 변환
a = np.arrange(20)
print(a)   #한 줄로 길게 출력됨
b = a.reshape((4,5))
print(b)   #4*5의 2차원 배열로 변환

#4. numpy 슬라이싱/인덱싱
lst = [
    [1,2,3],
    [4,5,6],
    [7,8,9]
]
arr= np.array(lst)
a = arr[0:2, 0:2]
print(a)   #1,2,4,5 만 출력
a = arr[1:, 1:]
print(a)   #5,6,8,9 만 출력
```