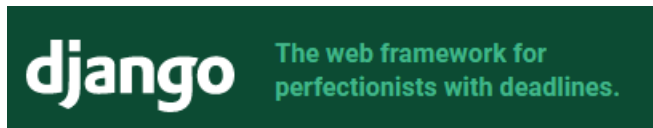


2주차 django 개발 시작

파이썬 언어로 빠르게 웹 사이트를 만들기 위해 django 프레임워크를 사용한다.

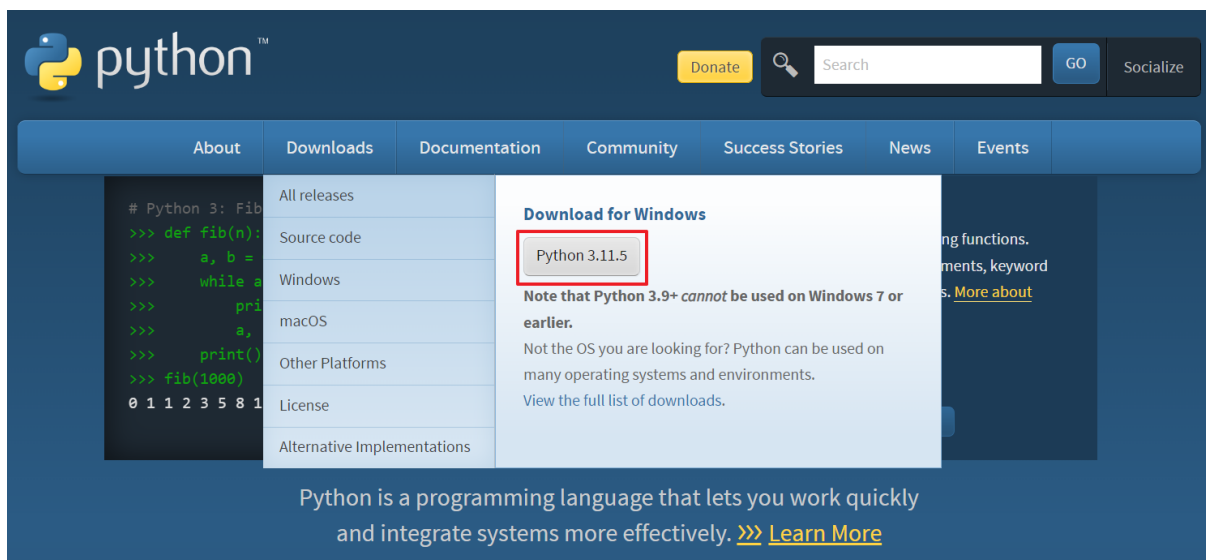
장고 사이트(<https://www.djangoproject.com>)에 접속해보면 좌측 상단에 아래와 같은 이미지가 보인다. 번역을 해보면 ‘마감 시간이 있는 완벽주의자를 위한 웹 프레임워크’ 이다. 장고는 파이썬 웹 프레임워크로 빠르며, 안심할 수 있고 확장성이 뛰어나다. 또한 무료이고 오픈소스다.



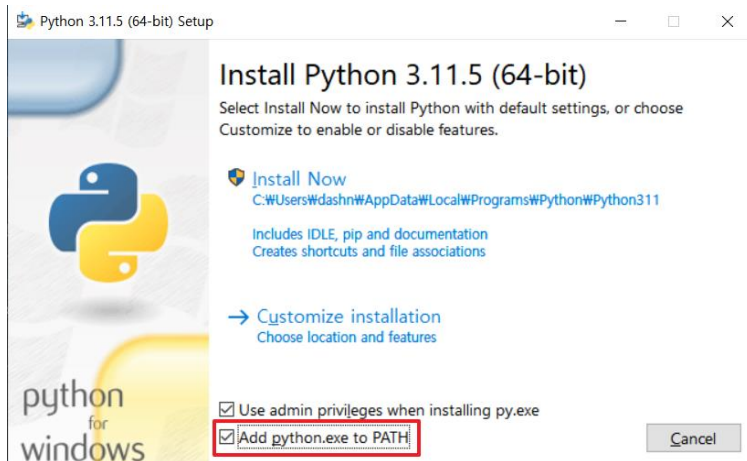
기본적인 파이썬 문법을 이해할 정도면 django로 개발을 하는데 큰 무리는 없다. 다른 프로그래밍 언어를 공부한 경험이 있다면 조건문, 반복문 정도까지의 문법을 살펴보면 된다. w3schools 사이트 (<https://www.w3schools.com/python/default.asp>)나 생활코딩 또는 개발서를 참고하자.

2-1) django 개발을 위한 준비

파이썬 공식 사이트(<https://www.python.org>)에 접속하여 OS에 맞는 버전을 다운 받아 설치한다. 본 문서는 windows를 기준으로 작성한다. 본 자료에서는 **Python 3.11.5** 버전을 사용하며, 무조건 최신 버전을 다운로드 받기 보다는 **Python 3.11.x** 버전을 사용하는 게 안정적이다.



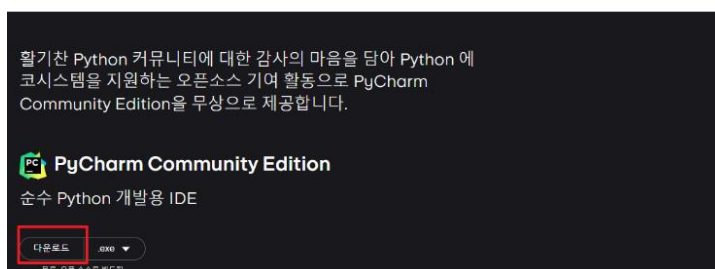
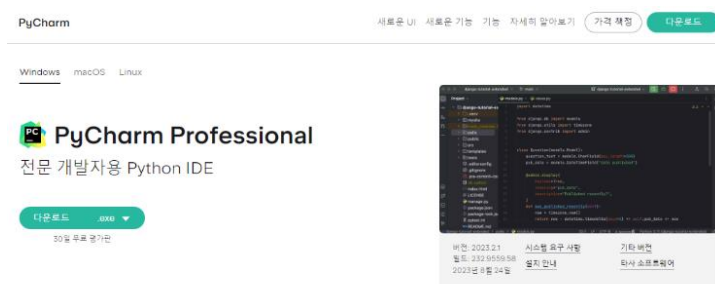
다운로드가 완료되면 설치 파일을 실행한다. **Install Now**를 클릭하기 전에 하단에 있는 **Add python.exe to PATH** 체크박스에 **V** 체크를 선택하자.



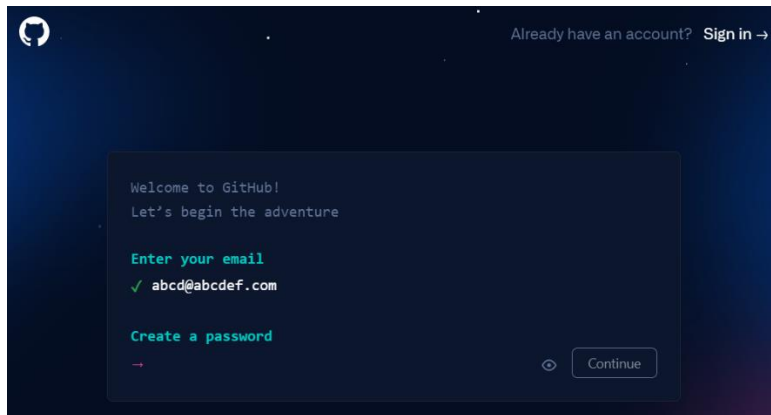
설치 후 cmd 명령 프롬프트 창을 열어 `python -V` 명령어를 입력해서 설치한 버전이 정상적으로 출력되는 지 확인하자. 알파벳 V는 대문자이다. 만약 기존에 설치한 다른 버전이 보인다면 환경변수의 path 설정을 해주면 된다.



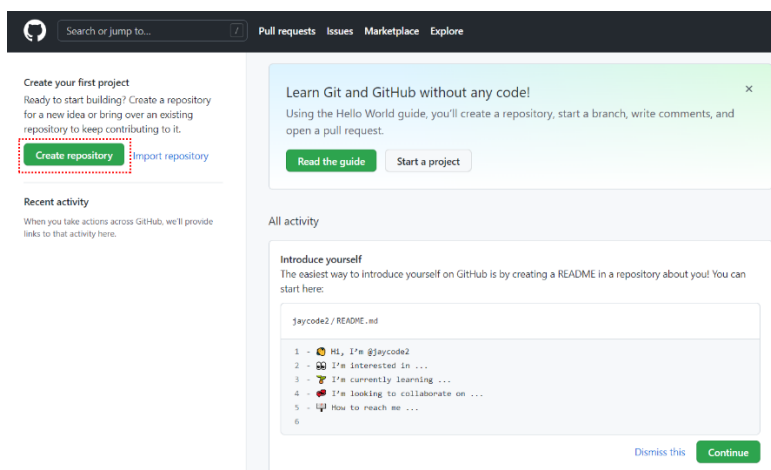
다음으로 파이썬 개발을 편하게 해주는 IDE인 파이참(<https://www.jetbrains.com/ko-kr/pycharm/download/?section=windows>)을 설치한다. 파이참은 유료인 Professional 버전과 무료인 Community 버전이 있다. 여기서는 아래로 스크롤을 내려서 무료 버전인 Community Edition Version을 다운로드 받아 설치하자. 파일명: pycharm-community-2023.2.1.설치도중 installation options화면이 있는데 4가지 체크박스가 보이면 모두 체크를 하고 다음으로 넘어가자. Mac OS라면 자신의 환경에 맞는 dmg 파일을 다운로드 받아 설치하자.



Github 계정이 없다면 공식 사이트(<https://github.com/>)에 접속하여 계정을 생성한다. 여러분이 작성한 코드를 깃허브에 올려야 하니 계정정보를 기억해두자.



로그인을 하면 아래와 같은 화면이 나온다. 왼쪽 상단에 있는 Create repository 버튼을 클릭해서 새로운 저장소를 만든다.



repository를 만들기 위한 입력 값을 채운다.

저장소 공개 항목에서 public을 선택하면 모든 사람에게 소스 코드가 공개된다. 개인적인 프로젝트라면 private를 선택한다. public을 선택하는 경우 개발 중 보안과 관련된 설정 시 항상 조심해야 한다.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner * jaycode2 Repository name * django_lec ✓

Great repository names are short and memorable. Need inspiration? How about [musical-waffle?](#)

Description (optional)
django project

☒ Public
Anyone on the internet can see this repository. You choose who can commit.

☐ Private
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☒ Add a README file
This is where you can write a long description for your project. [Learn more.](#)

☒ Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: Python

☐ Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

This will set main as the default branch. Change the default name in your [settings](#).

[Create repository](#)

작성이 다 되었다면 Create repository 버튼을 누른다.

Search or jump to... [Pull requests](#) [Issues](#) [Marketplace](#) [Explore](#)

[jaycode2](#) / [django_lec](#) Public

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

main 1 branch 0 tags [Go to file](#) [Add file](#) [Code](#)

jaycode2 Initial commit 28e2faf now 1 commit

File	Commit	Time
.gitignore	Initial commit	now
README.md	Initial commit	now

README.md

django_lec

django project

생성한 저장소를 로컬 PC에 Clone 한다. 개발하고 있는 PC와 깃허브 저장소를 연결한다고 보면 된다. 우측의 Code를 클릭한 뒤 저장소의 URL을 복사한다.

Search or jump to... [Pull requests](#) [Issues](#) [Marketplace](#) [Explore](#)

[jaycode2](#) / [django_lec](#) Public

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

main 1 branch 0 tags [Go to file](#) [Add file](#) [Code](#)

jaycode2 Initial commit 28e2faf now 1 commit

File	Commit	Time
.gitignore	Initial commit	now
README.md	Initial commit	now

README.md

django_lec

django project

Clone

HTTPS [SSH](#) [GitHub CLI](#)

https://github.com/jaycode2/django_lec.git [Copy](#)

Use Git or checkout with SVN using the web URL.

[Open with GitHub Desktop](#)

[Download ZIP](#)

로컬 PC에 Clone하기 위한 폴더를 생성하고 복사한 URL을 명령어와 함께 입력한다. 저장소를 private으로 생성했다면 깃허브 사용자 이름과 비밀번호를 추가로 입력한다.

C:₩경로 상태에서 아래 명령어를 입력한다(아래 화면 참고)

`mkdir github`

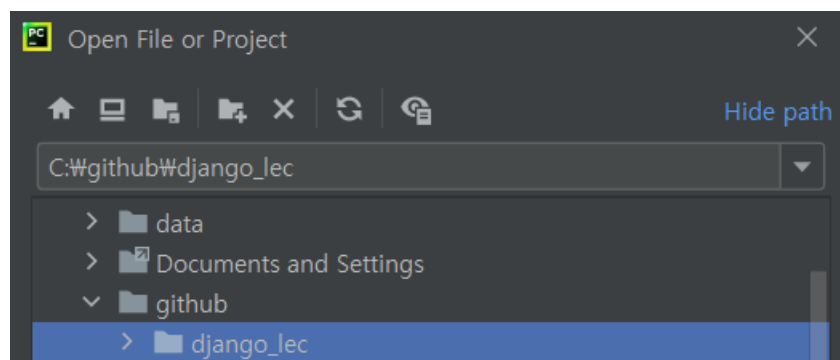
`cd github`

`git clone` 복사한경로(우클릭 하면 복사한 내용이 붙여넣기 된다)

```
C:₩>mkdir github
C:₩>cd github
C:₩github>git clone https://github.com/jaycode2/django_lec.git
Cloning into 'django_lec'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
C:₩github>
```

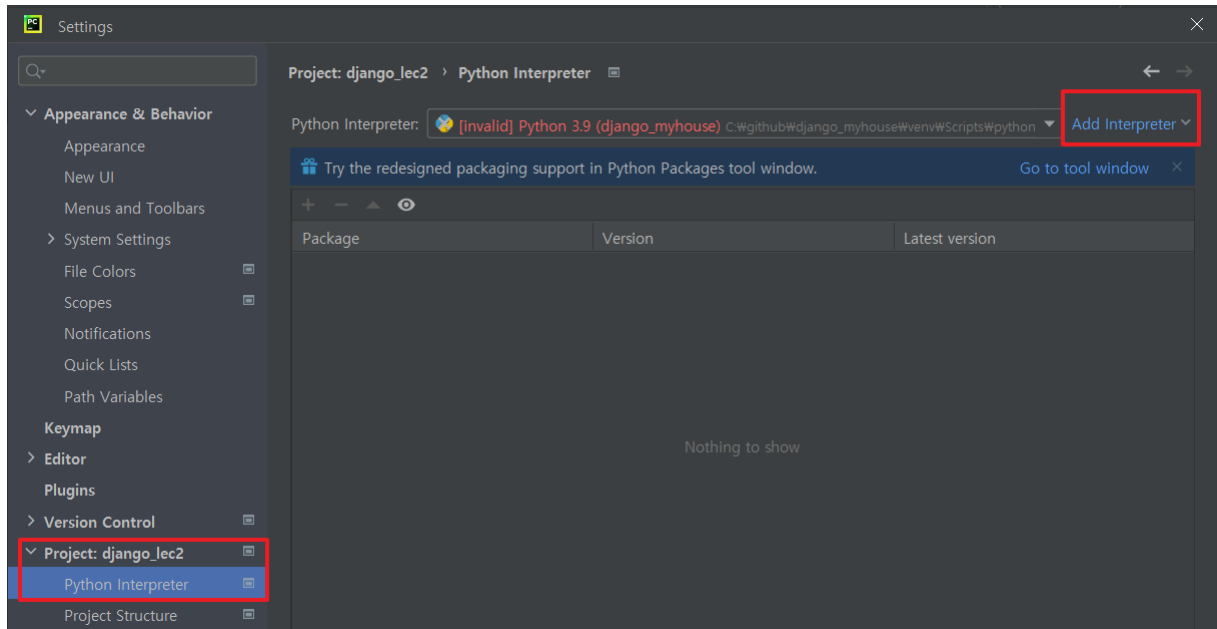
2-2) 파이참(PyCharm) 개발 환경 설정

파이참을 실행하고 좌측 상단의 [File > Open]을 선택한다. 깃허브에서 Clone한 경로를 찾아 선택하고 OK버튼을 누른다.

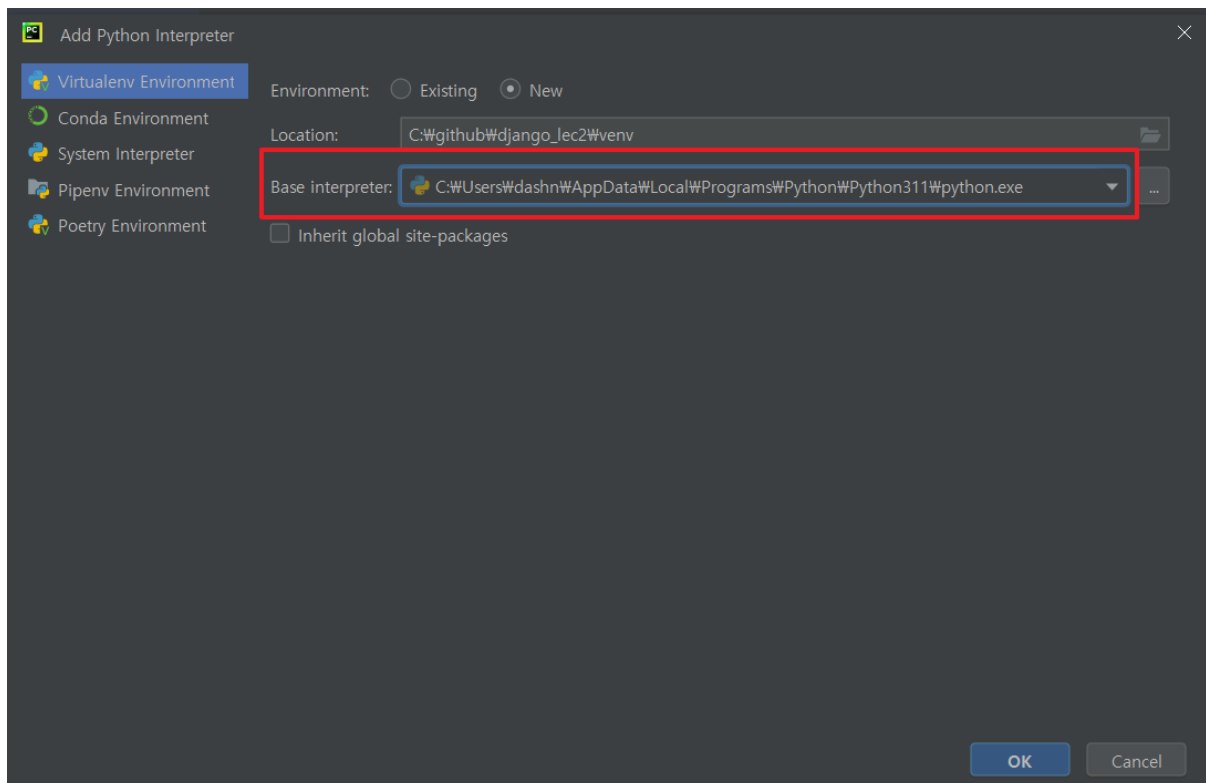


프로젝트를 위한 가상환경을 만든다. 여러 개의 프로젝트를 진행할 경우 각 프로젝트 별로 개발 환경이 다를 수 있는데 이때 가상 환경을 이용하면 하나의 PC에서 독립된 여러 개의 작업이 가능하다.

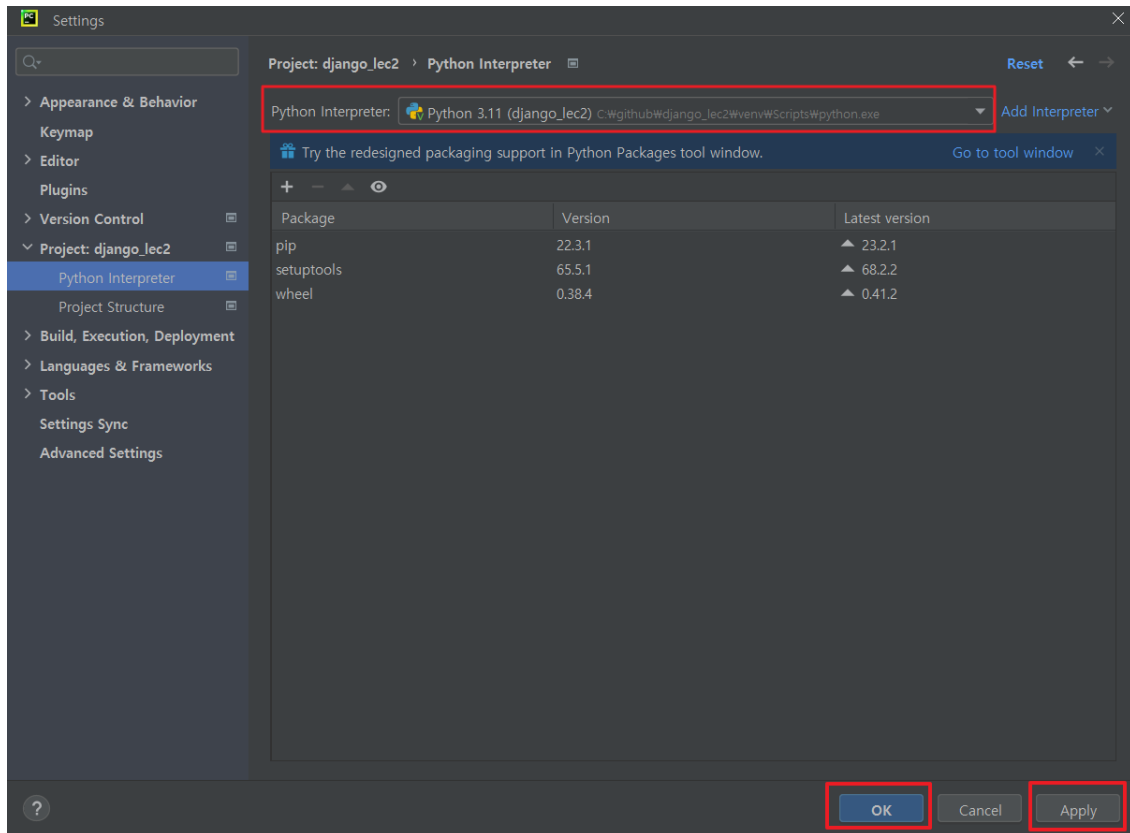
가상 환경을 만드는 방법은 여러 가지가 있지만 이 프로젝트에서는 파이참에서 제공하는 기능을 활용한다. 파이참에서 [File > Settings > Project: 프로젝트명] 을 선택하고 우측 설정 버튼에서 Add Interpreter를 클릭한다.



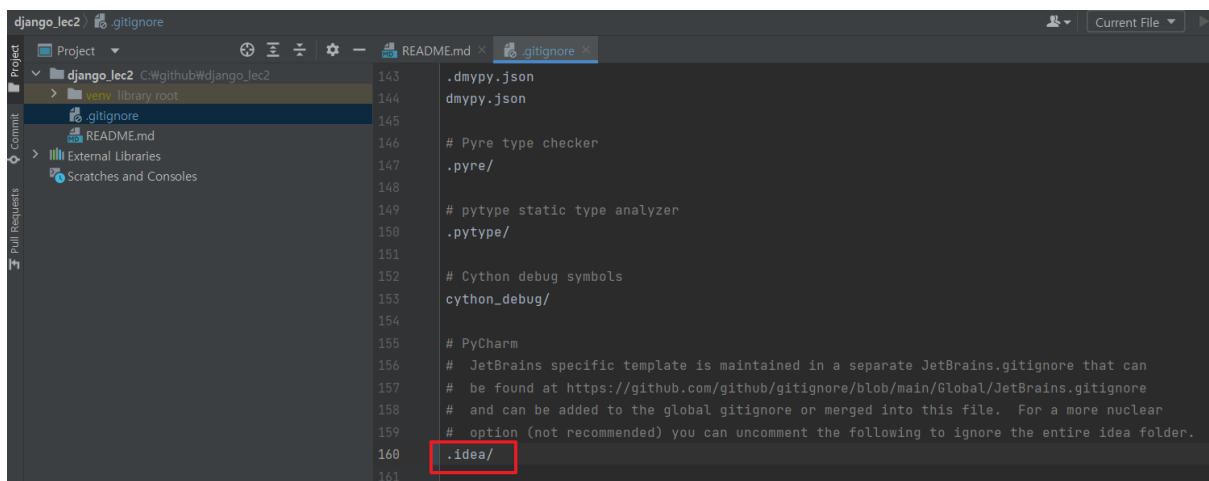
New environment가 기본적으로 설정되어 있고 경로도 나온다. Base Interpreter에 설치한 파이썬 경로를 선택하고 OK버튼을 선택하면 가상 환경이 venv 폴더 안에 생성된다.



Python Interpreter에 pip와 setuptools가 보이면 정상이다. 우측 하단의 Apply를 누르고 OK를 클릭한다.



생성된 파일 중 .gitignore 맨 하단에 아래 코드를 활성화한다. .idea는 파이참에서 설정한 내용을 담은 폴더인데, 버전 관리를 할 필요가 없다. 오히려 다른 PC에서 작업할 때 프로젝트 설정이 맞지 않아서 문제가 생길 수도 있다.



가상환경에서 django를 설치하기 위해 프로젝트 경로에서 venv\Scripts\activate.bat를 입력한다. Mac을 사용한다면 source venv/bin/activate를 입력한다. 프롬프트 앞에 (venv)가 보이면 가상환경이 정상적으로 실행되었다는 의미다.

```
C:\github\django_lec>venv\Scripts\activate.bat
(venv) C:\github\django_lec>_
```

pip list를 입력하면 현재 설치된 패키지가 보인다. Pip 버전을 업데이트 하라는 안내가 뜨는 경우 초록색 명령어를 그대로 입력하면 업데이트가 된다. python.exe -m pip install --upgrade pip

```
(venv) C:\github\django_lec2\venv\Scripts>pip list
Package      Version
-----
pip          22.3.1
setuptools   65.5.1
wheel        0.38.4

[notice] A new release of pip available: 22.3.1 -> 23.2.1
[notice] To update, run: python.exe -m pip install --upgrade pip

(venv) C:\github\django_lec2\venv\Scripts>
```

```
(venv) C:\github\django_lec2\venv\Scripts>python.exe -m pip install --upgrade pip
Requirement already satisfied: pip in c:\github\django_lec2\venv\lib\site-packages (22.3.1)
Collecting pip
  Downloading pip-23.2.1-py3-none-any.whl (2.1 MB)
    ----- 2.1/2.1 MB 6.1 MB/s eta 0:00:00
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 22.3.1
    Uninstalling pip-22.3.1:
      Successfully uninstalled pip-22.3.1
Successfully installed pip-23.2.1

(venv) C:\github\django_lec2\venv\Scripts>
```

명령 프롬프트 창에서 상위 디렉토리로 이동하고 싶다면 cd .. 명령어를 입력하면 된다.

```
(venv) C:\github\django_lec2\venv\Scripts>cd ..
(venv) C:\github\django_lec2\venv>cd ..
(venv) C:\github\django_lec2>
```


django 설치를 위해 프로젝트 경로로 돌아와서, 명령 프롬프트 창에 `pip install django`를 입력한다.

```
C:\WINDOWS\system32\cmd.exe - pip install django
(venv) C:\github#django Lec2>pip install django
Collecting django
  Obtaining dependency information for django from https://files.pythonhosted.org/package
b43a830fdf776edd60c2e25c7c5f4d23cc243/Django-4.2.5-py3-none-any.whl.metadata
  Downloading Django-4.2.5-py3-none-any.whl.metadata (4.1 kB)
Collecting asgiref<4,>=3.6.0 (from django)
```

설치가 완료되면 `pip list`를 다시 입력해서 Django가 보이는 지 확인한다.

```
(venv) C:\github#django Lec2>pip list
Package      Version
-----
asgiref      3.7.2
Django       4.2.5
pip          23.2.1
setuptools   65.5.1
sqlparse     0.4.4
tzdata       2023.3
wheel        0.38.4
```

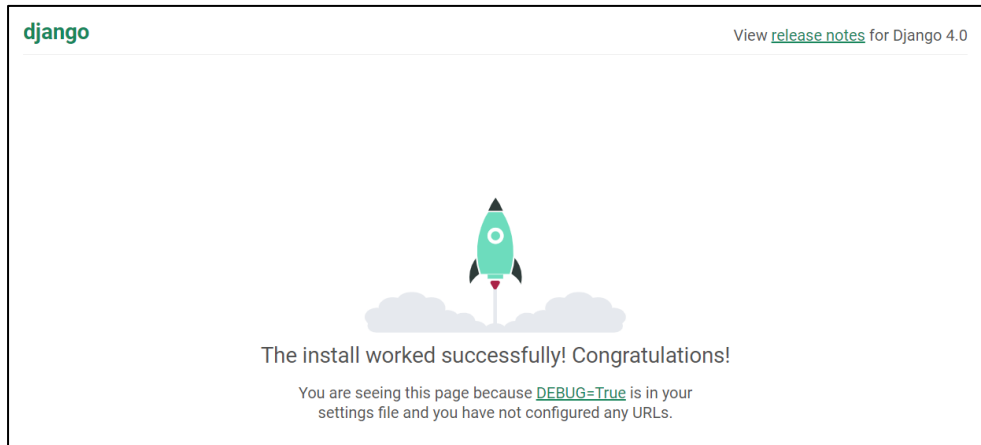
django 프로젝트 생성을 위해 `django-admin startproject my_project .` 을 입력한다. 마지막에 마침표(.)까지 꼭 넣어야 한다. 다음으로 `python manage.py runserver`로 서버를 실행한다.

```
(venv) C:\github#django Lec2>django-admin startproject my_project .
(venv) C:\github#django Lec2>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin,
auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
December 30, 2021 - 09:47:55
Django version 4.0, using settings 'my_project.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

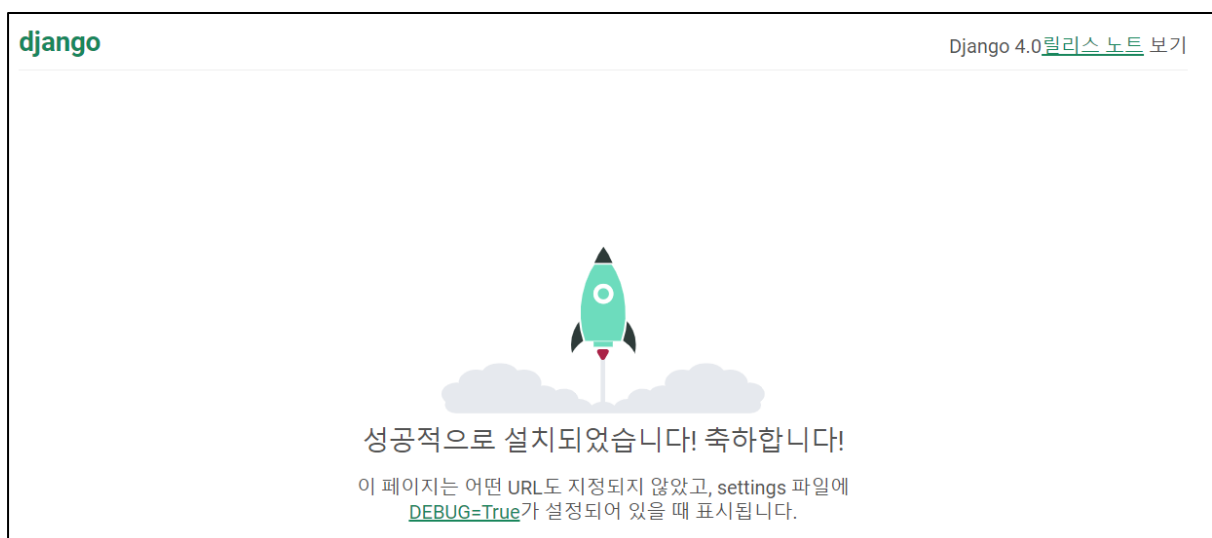
마이그레이션에 대한 내용이 나오지만 일단은 개발 서버가 실행된 모습을 화면으로 확인해보자. 브라우저 열어 <http://127.0.0.1:8000/> 또는 localhost:8000/를 입력한다.



개발 서버를 실행했는데 화면이 영어로 보인다. 파이참 my_project/setting.py 파일을 열어 언어와 시간을 설정해준다.

```
LANGUAGE_CODE = 'ko-kr'
TIME_ZONE = 'Asia/Seoul'
USE_I18N = True
USE_TZ = True
```

브라우저를 새로고침하면 한글로 변경된 화면이 출력된다. 화면이 정상적으로 보이지 않을 경우 터미널에서 `python manage.py runserver`를 입력해서 서버를 다시 구동한다. 서버 구동을 종료하려면 `Ctrl + C`를 누르면 된다.

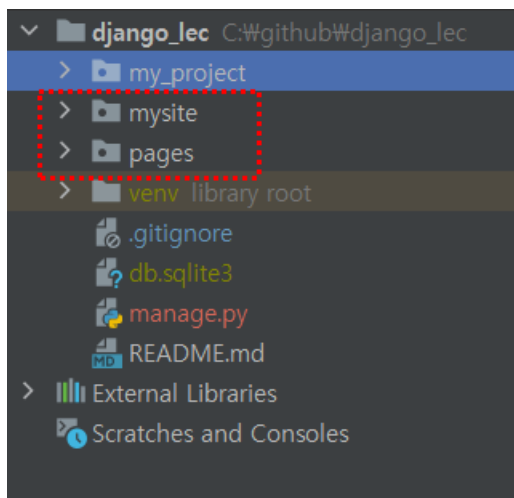


앞서 적용되지 않은 18개의 마이그레이션에 대한 처리를 해주자. migration 개수는 버전에 따라 다를 수 있다. `python manage.py migrate` 를 입력한다.

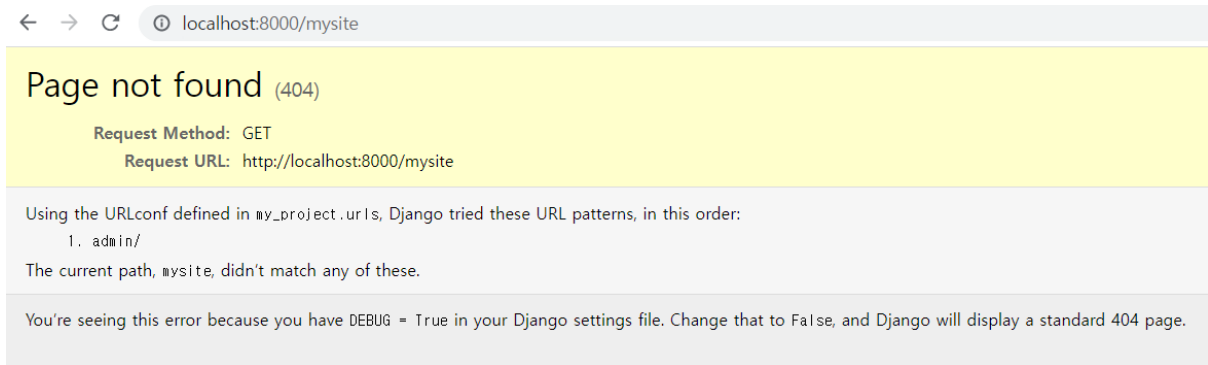
```
(venv) C:\github\django_lect>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
```

2-3) 장고로 개발 시작하기

앞서 my_project 프로젝트를 생성했지만 아직 아무 기능이 없는 상태다. 기능을 추가하기 위해서는 앱(App)을 추가해야 한다. 장고 프로젝트는 이 앱의 모음이라고 이해하면 된다. 하나의 프로젝트에 여러 개의 앱을 개발할 수 있다. python manage.py startapp mysite를 입력하여 mysite 앱을 만들자. 개별 페이지를 작성하기 위한 python manage.py startapp pages도 입력한다. 모든 작업은 venv 가상 환경에서 해야 한다는 점을 유의하자.



전형적이지만 화면에 Hello World를 출력해보자. 서버를 구동한 뒤에 앱을 만들었으니 브라우저에 <http://localhost:8000/mysite> 를 입력해보자. <http://localhost:8000>만 입력했을 때와 달리 404 오류가 발생한다. 404 오류는 요청한 페이지를 찾을 수 없을 경우 출력된다.



우리는 앱을 생성했지만 URL에 대해 매핑하는 작업을 아직 하지 않았다. `mysite > views.py` 에 아래 코드를 입력하자.

```
from django.http import HttpResponse

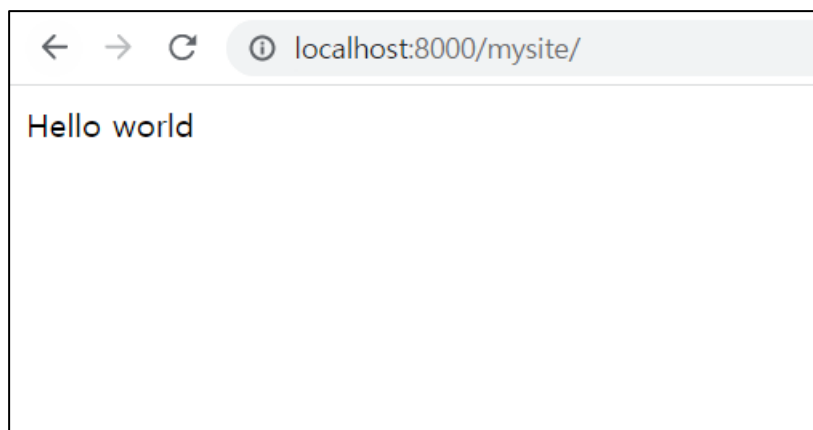
def index(request):
    return HttpResponse("Hello world")
```

`my_project > urls.py`에 아래 점선 사각형 코드도 추가한다.

```
from django.contrib import admin
from django.urls import path
from mysite import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('mysite/', views.index),
]
```

`python manage.py runserver`를 입력하여 서버를 재구동하고 브라우저에 다시 URL을 입력하면 Hello world가 출력된다.



위 흐름은 개발 과정에서 계속 반복되므로 이해를 해두자. 브라우저에서 로컬 서버로 localhost:8000/mysite 페이지 호출을 요청하면 urls.py에서 mysite 매핑을 확인하고 views.py 에서 index 함수를 호출해서 그 결과를 다시 브라우저에 출력하는 형태다.

우리가 수정한 my_project > urls.py는 가장 상위의 URL 매핑 파일이다. 개별 앱을 수정할 때마다 프로젝트 전체에 영향을 주는 건 위험도가 크기 때문에 기존 소스를 일부 수정해보자. my_project > urls.py 에서 from mysite import views를 삭제하고 노란색 강조색 부분을 수정한다.

```
from django.contrib import admin
from django.urls import path, include
from mysite import views # 삭제

urlpatterns = [
    path('admin/', admin.site.urls),
    path('mysite/', include('mysite.urls')),
]
```

include 안에 있는 mysite 앱에는 아직 urls.py 파일이 없다. my_project > urls.py 파일을 복사해서 mysite에도 urls.py 파일을 생성하고 아래와 같이 내용을 입력한다. <http://localhost:8000/mysite/> 에서 동일하게 Hello World가 출력된다면 정상이다.

```
from django.urls import path

from . import views

urlpatterns = [
    path('', views.index),
]
```

2-4) 장고 모델(django model)

장고는 모델(Model)을 사용해서 데이터베이스를 처리한다. 데이터베이스를 다루기 위해서는 보통 SQL에 대한 공부가 필요하지만 장고는 SQL 작성없이 데이터를 처리한다. (참고 url: <https://docs.djangoproject.com/en/4.0/intro/tutorial02/>) 추가로 ORM에 대한 개념을 이해하면 좋다.

mysite/models.py 파일에 웹사이트 본문에 해당하는 모델을 정의한다.

```
from django.db import models

class MainContent(models.Model):
    title = models.CharField(max_length=200)
    content = models.TextField()
    pub_date = models.DateTimeField('date published')
```

장고 모델에서 사용하는 Field 타입에 대한 더 많은 내용은 다음 URL을 참고하자. (참고 url: <https://docs.djangoproject.com/en/3.0/ref/models/fields/#field-types>)

가상환경 터미널에서 python manage.py makemigrations를 입력한다.

```
(venv) C:\github\django Lec>python manage.py makemigrations
No changes detected
```

models.py를 수정했는데도 변경이 없다고 하는 건 my_project의 setting.py에 앱을 등록하지 않았기 때문이다. setting.py에 생성한 앱을 추가한다.

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'mysite',
    'pages',
]
```

다시 python manage.py makemigrations를 입력하면 이번에는 정상적으로 처리가 된 모습이다.

```
(venv) C:\github\django Lec>python manage.py makemigrations
Migrations for 'mysite':
  mysite\migrations\0001_initial.py
  - Create model MainContent
```

.gitignore는 깃에서 관리하지 않을 목록을 관리한다. 아래와 같이 추가하면 깃에서는 migrations의 내용을 관리하지 않는다.

```
# PyCharm
.idea/
migrations/
```

실제 데이터베이스 모델에 적용하기 위해 `python manage.py migrate` 명령어를 입력한다.

```
(venv) C:\github\django lec>python manage.py makemigrations
Migrations for 'mysite':
  mysite\migrations\0001_initial.py
    - Create model MainContent

(venv) C:\github\django lec>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, mysite, sessions
Running migrations:
  Applying mysite.0001_initial... OK
```

2-5) 관리자(Admin) 페이지에서 MainContent 작성하기

장고를 사용하면 관리자 페이지를 쉽게 구성할 수 있다. `admin.py`에 간단한 수정으로 사이트를 쉽게 관리한다(참고 url: <https://docs.djangoproject.com/en/4.0/intro/tutorial02/>) `python manage.py createsuperuser` 명령어를 입력한다. 사용자 이름, 이메일 주소를 입력하고 비밀번호까지 넣으면 관리자 계정이 생성된다.

```
(venv) C:\github\django lec>python manage.py createsuperuser
사용자 이름 (leave blank to use ' '): admin
이메일 주소: admin@myproject.com
Password:
Password (again): _
```

서버를 구동하고 <http://localhost:8000/admin/> 경로로 접속하면 아래와 같은 화면이 나온다.



The image shows the Django Admin login interface. It has a dark blue header with the text 'Django 관리'. Below the header, there are two input fields: '사용자 이름:' (Username) with the value 'admin' and '비밀번호:' (Password) with four dots. At the bottom, there is a blue button labeled '로그인' (Login).

로그인해서 사용자 항목을 선택하면 가입한 관리자 계정의 정보가 보인다.

Django 관리

환영합니다

홈 > 인증 및 권한 > 사용자(들)

Start typing to filter...

인증 및 권한

그룹 + 추가

사용자(들) + 추가

변경할 사용자 선택

Q

검색

액션:

실행

1 중 아무것도 선택되지 않았습니다.

<input type="checkbox"/>	사용자 이름	이메일 주소	이름	성	스태프 권한
<input type="checkbox"/>	admin	admin@myproject.com			✔

1 사용자

관리자 페이지에서 MainContent를 조작할 수 있다. mysite/admin.py에서 아래와 같이 입력 후 관리자 페이지를 새로고침 한다.

```
from django.contrib import admin
from .models import MainContent

admin.site.register(MainContent)
```

Django 관리

환영합니다. ADMIN 사이트 보기 / 비밀번호 변경 / 로그아웃

홈 > Mysite > Main contents

Start typing to filter...

MYSITE

Main contents + 추가

인증 및 권한

그룹 + 추가

사용자(들) + 추가

변경할 main content 선택

MAIN CONTENT 추가 +

0 main contents

추가 버튼을 누르고 제목, 내용 그리고 날짜를 입력한 뒤 저장을 누른다.

Django 관리

환영합니다. ADMIN 사이트 보기 / 비밀번호 변경 / 로그아웃

홈 > Mysite > Main contents > main content 추가

필터에 타이핑 시작...

MYSITE

Main contents + 추가

인증 및 권한

그룹 + 추가

사용자(들) + 추가

main content 추가

Title:

첫 번째 상품등록

Content:

첫 번째 상품입니다.

상품의 사이즈는 L 입니다.

Date published:

날짜:

2023-09-13

오늘 📅

시각:

22:45:01

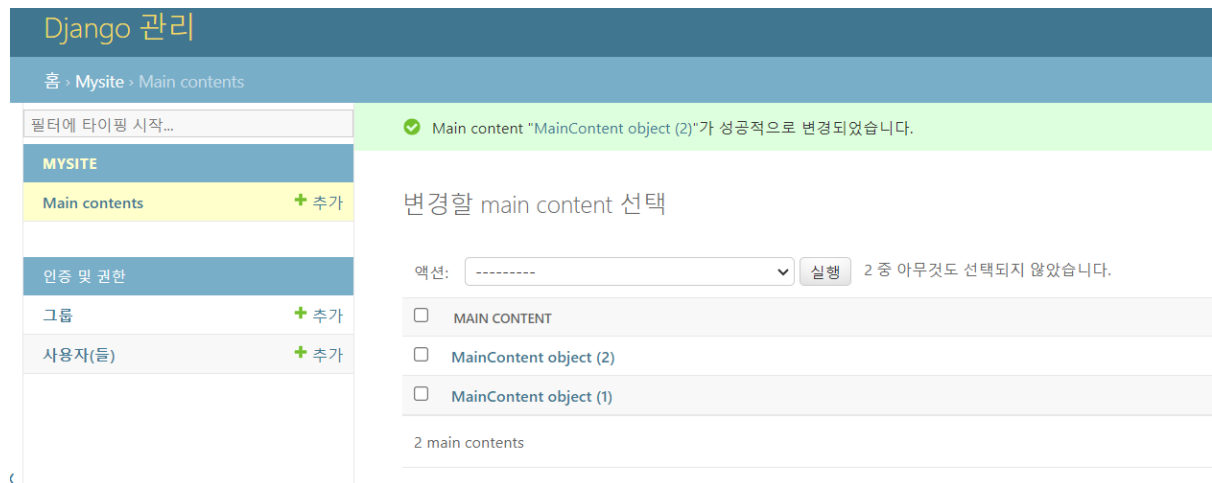
현재 ⌚

저장

저장 및 다른 이름으로 추가

저장 및 편집 계속

Content를 몇 개 더 작성해서 목록이 쌓이는지 확인해보자.



2-6) 작성한 MainContent 화면에 출력하기

<http://localhost:8000/mysite/> 입력 시 브라우저에 관리자 페이지에서 등록한 내용을 출력해보자.

mysite/views.py를 수정한다.

```
from django.http import HttpResponse # 삭제
from django.shortcuts import render
from .models import MainContent

def index(request):
    return HttpResponse("Hello world") # 삭제

    content_list = MainContent.objects.order_by('-pub_date')
    context = {'content_list': content_list}
    return render(request, 'mysite/content_list.html', context)
```

질문 목록은 `MainContent.objects.order_by('-pub_date')`으로 출력한다. `pub_date` 앞에 마이너스 기호(-)를 붙이면 역순으로 정렬된다. 가장 최신 콘텐츠를 상단에 노출하기 위함이다. `render` 함수는 `content_list`의 데이터를 `mysite/content_list.html` 파일에 적용 후 HTML을 리턴 한다.

(수정 후)

```
1  from django.shortcuts import render
2  from .models import MainContent
   1 usage
3  def index(request):
4      content_list = MainContent.objects.order_by('-pub_date')
5      context = {'content_list': content_list}
6      return render(request, template_name: 'mysite/content_list.html', context)
```

mysite/content_list.html와 같은 파일을 템플릿이라고 한다. 템플릿을 저장할 디렉토리를 만들기 전에 my_project/settings.py 파일을 일부 수정하자.

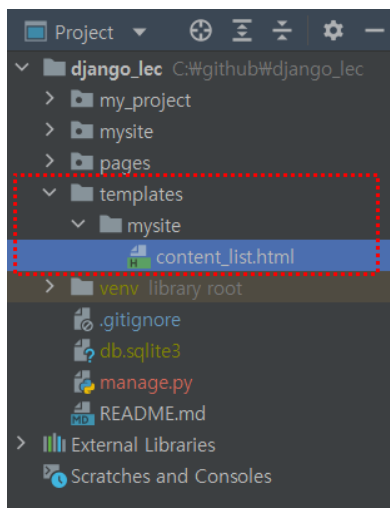
```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [BASE_DIR / 'templates'],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    ],
]
```

터미널에서 프로젝트 하위에 templates 디렉토리를 생성한다.

```
(venv) C:\github\django Lec>mkdir templates
```

!!!! 금방 끝나겠다!!

파이참에서 templates 디렉토리를 우클릭 한 뒤 [New > directory]를 선택하여 mysite 폴더를 만든다. mysite 폴더를 우클릭 하여 [New > HTML File]을 선택하여 파일명을 content_list라고 입력한다.



templates/mysite/content_list.html에 다음 내용을 작성한다.

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <title>mysite</title>
</head>
<body>
<h1>mysite</h1>
{% if content_list %}
    {% for content in content_list %}
        <hr/>
        <h3><a href="/mysite/{{ content.id }}/"/>{{ content.title }}</a></h3>
        <h5>{{ content.content }}</h5>
    {% endfor %}
{% else %}
    <p>content가 없습니다.</p>
{% endif %}
</body>
</html>
```

템플릿을 보면 {% %}로 둘러 쌓인 문장이 있는데 이를 템플릿 태그라고 한다. {% if content_list %}로 콘텐츠 목록이 있는지 검증하고 없다면 {% else %}로 content가 없다는 내용을 출력한다. {% for content in content_list %}는 반복문으로 이해하면 된다(참고 url: <https://docs.djangoproject.com/en/4.0/intro/tutorial03/>)

개발 서버를 구동하고 브라우저에 <http://localhost:8000/mysite/> 를 입력해서 MainContent 내용을 확인한다.

mysite

[두 번째 상품을 등록합니다.](#)

두 번째 상품은 다음과 같습니다 상품의 특징은 BB입니다.

[상품 소개1](#)

첫 번째 상품입니다. 이 상품의 특징은 AA 입니다.

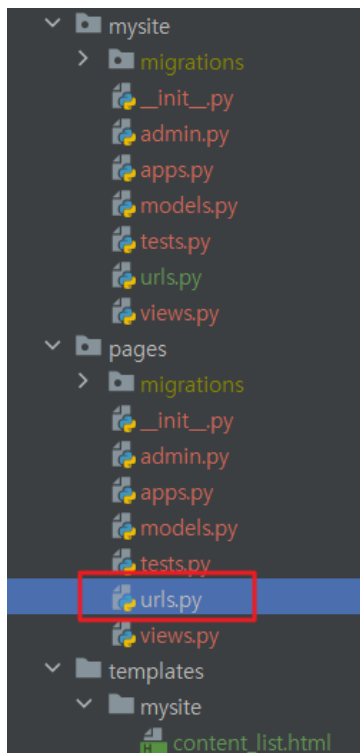
2-7) 메인 페이지와 회사소개 페이지 만들기

홈페이지 메인과 회사소개 페이지를 추가해보자. my_project/urls.py에 pages를 사용하기 위해 url을 매핑한다.

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('mysite/', include('mysite.urls')),
    path('', include('pages.urls')),
```

mysite앱의 urls.py파일을 pages 앱 하위 동일한 위치에 복사 붙여넣기하고 아래와 같이 메인 페이지와 회사소개를 위한 path를 설정한다.



```
from django.urls import path

from . import views

urlpatterns = [
    path('', views.mainpage),
    path('company/', views.company),
]
```

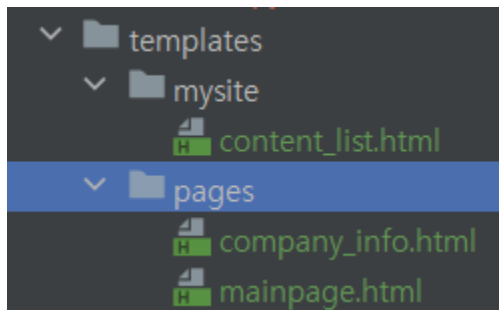
pages/views.py에 아래와 같이 입력한다.

```
from django.shortcuts import render

def mainpage(request):
    return render(request, 'pages/mainpage.html')

def company(request):
    return render(request, 'pages/company_info.html')
```

templates 디렉토리 아래에 pages 폴더를 만들고 mainpage.html, company_info.html 파일을 생성한다.



pages/mainpage.html

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <title>메인 페이지</title>
</head>
<body>
<nav>
    <a href="/company/">company info</a>
    <a href="/mysite/">product</a>
</nav>

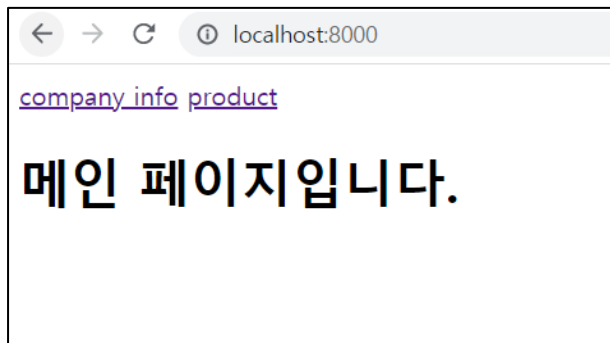
<h1>메인 페이지입니다.</h1>
</body>
</html>
```

pages/company_info.html

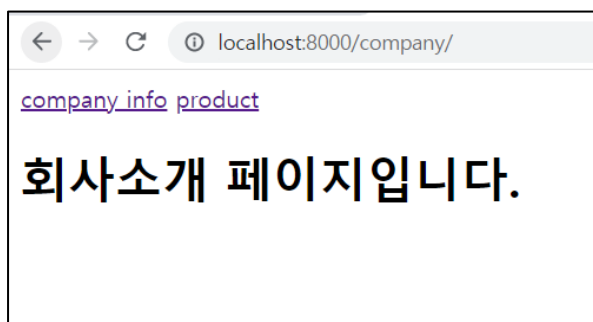
```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <title>회사소개</title>
</head>
<body>
<nav>
  <a href="/company/">company info</a>
  <a href="/mysite/">product</a>
</nav>

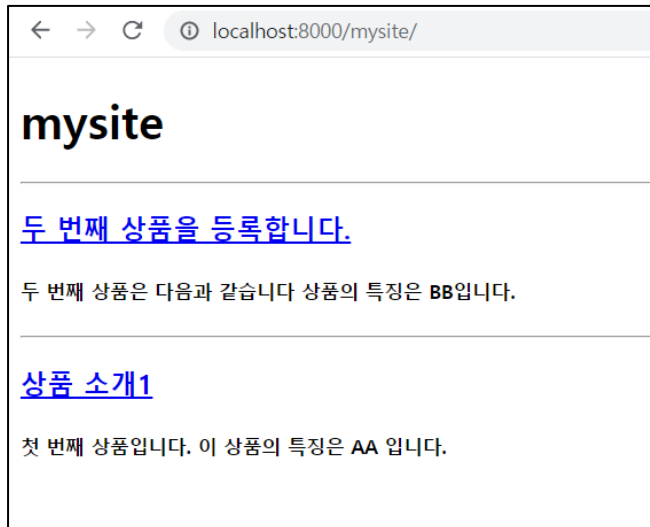
<h1>회사소개 페이지입니다.</h1>
</body>
</html>
```

개발 서버를 재구동하고 화면에서 결과를 확인한다.



company_info와 product도 정상 출력되는지 확인하자.

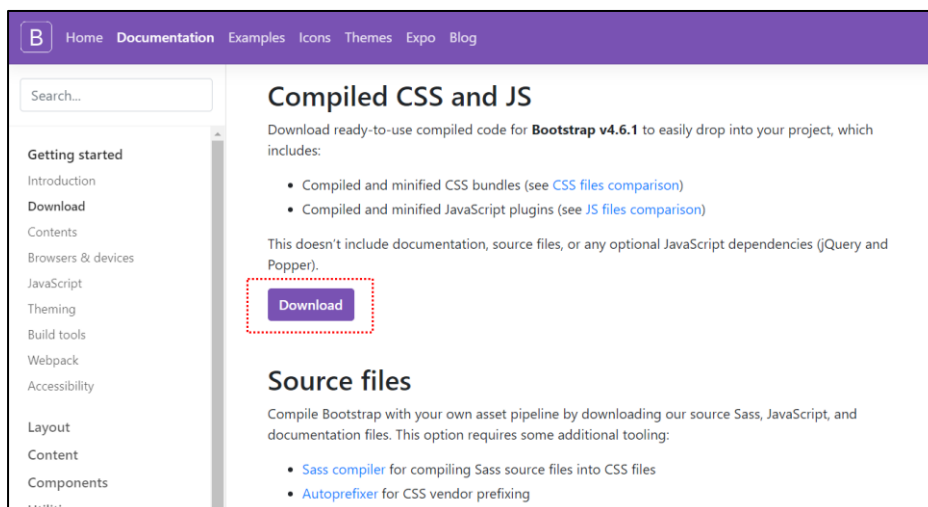




2-8) 부트스트랩 적용하기

디자인 요소를 HTML과 CSS로 하나씩 작업하는 건 굉장히 오랜 시간이 걸린다. bootstrap 템플릿을 활용하여 깔끔한 페이지로 바꿔보자. 먼저 부트스트랩을 다운 받는다(url:

<https://getbootstrap.com/docs/4.6/getting-started/download/>)



my_project/settings.py 파일을 수정한다(참고 url:

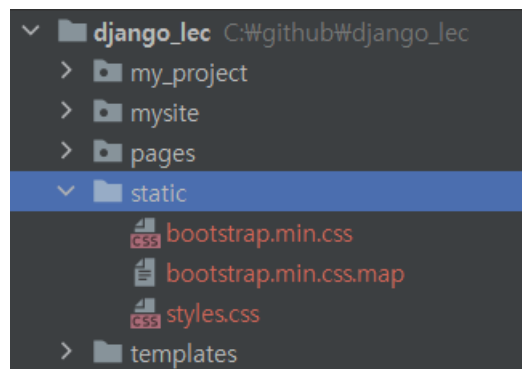
<https://docs.djangoproject.com/en/4.0/howto/static-files/>)

```
# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.0/howto/static-files/

STATIC_URL = 'static/'
STATICFILES_DIRS = (
    BASE_DIR / 'static',
)
```

static 폴더를 아래와 같이 생성하고 bootstrap.min.css 파일과 bootstrap.min.css.map 파일을 그 안에 넣는다.

```
(venv) C:\github\django Lec>mkdir static
```



<https://startbootstrap.com/> 사이트에서 무료 템플릿을 다운로드 받는다. 본 강의에서는 <https://startbootstrap.com/template/small-business>를 사용했다. 무료로 다운로드를 받아 index.html의 내용을 전체 복사해서 content_list.html에 붙여 넣고 일부 수정을 한다. static 경로에 styles.css 파일도 추가한다. 기존 context_list.html에 있는 내용 중 데이터를 처리하는 부분은 다시 적용을 해야하니 따로 복사해둔다.

상단

```
<!DOCTYPE html>
<% load static %>
<html lang="ko">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no" />
    <meta name="description" content="" />
```



```

<meta name="author" content="" />
<title>Small Business - Start Bootstrap Template</title>
<!-- Favicon-->
<link rel="icon" type="image/x-icon" href="assets/favicon.ico" />
<!-- Core theme CSS (includes Bootstrap)-->
<link rel="stylesheet" type="text/css" href="{% static 'styles.css' %}">
<link rel="stylesheet" type="text/css" href="{% static 'bootstrap.min.css' %}">
</head>

```

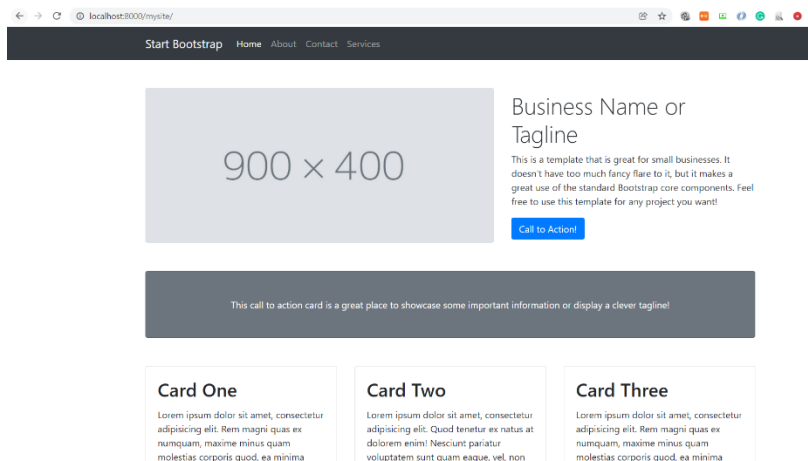
하단 일부 삭제

```

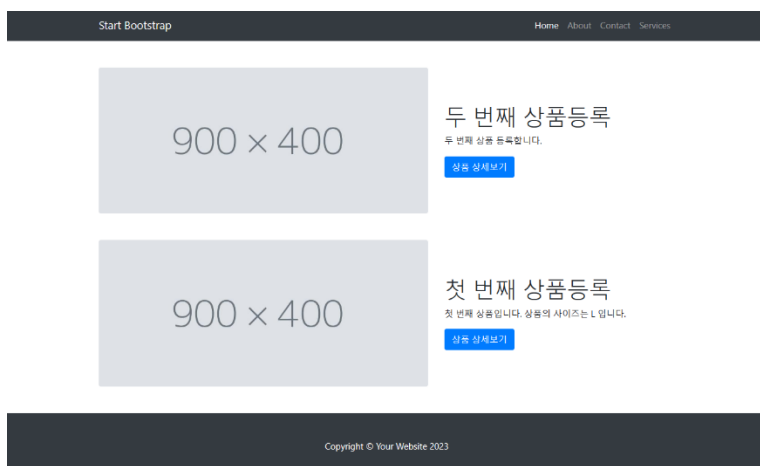
<!-- Bootstrap core JS-->
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>
<!-- Core theme JS-->
<script src="js/scripts.js"></script>

```

서버를 재구동하고 localhost:8000/mysite에서 템플릿이 정상적으로 적용됐는지 확인하자.



복사해두었던 context_list.html 파일의 내용을 활용하여 템플릿에 다시 데이터를 출력해보자. 아래와 같은 결과가 나오도록 수행해보자.



작업이 완료되었다면 github에 커밋을 한다.

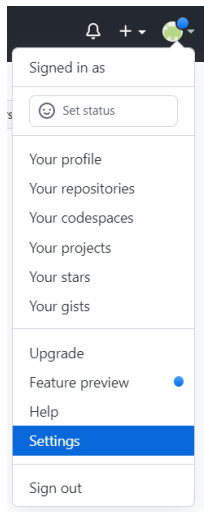
```
(venv) C:\github\django_lec>git add .  
(venv) C:\github\django_lec>git commit -m "2주차 django 개발 시작"
```

마지막으로 git push 명령어를 입력했을 때 로그인 창이 출력된다. 정상적으로 깃 로그인을 한 줄 알았는데 로그인 실패가 떴다.

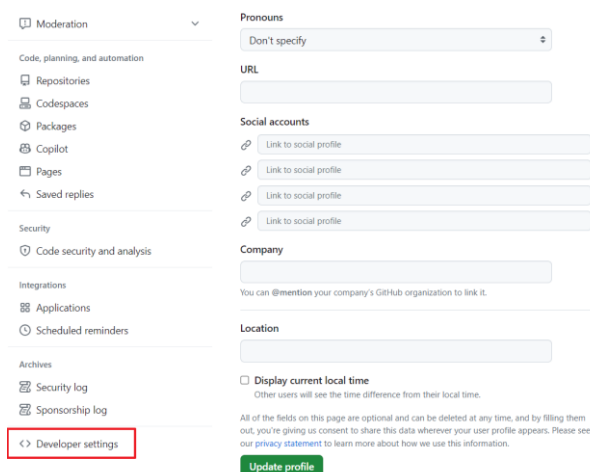
```
(venv) C:\github\django_lec2>git push  
Ligon failed, use ctrl+c to cancel basic credential prompt.  
Username for 'https://github.com': jaycode2  
Password for 'https://jaycode2@github.com':  
remote: Support for password authentication was removed on August 13, 2021.  
remote: Please see https://docs.github.com/en/get-started/getting-started-with-git/about-remote-repositories#cloning-with-https-urls for information on currently recommended modes of authentication.  
fatal: Authentication failed for 'https://github.com/jaycode2/django_lec2.git/'
```

내용을 보면 2021년 8월 13일부터 git 작업을 인증할 때 계정 암호를 인정하지 않는다는 의미다. 대신 개인 토큰을 사용해야 한다.

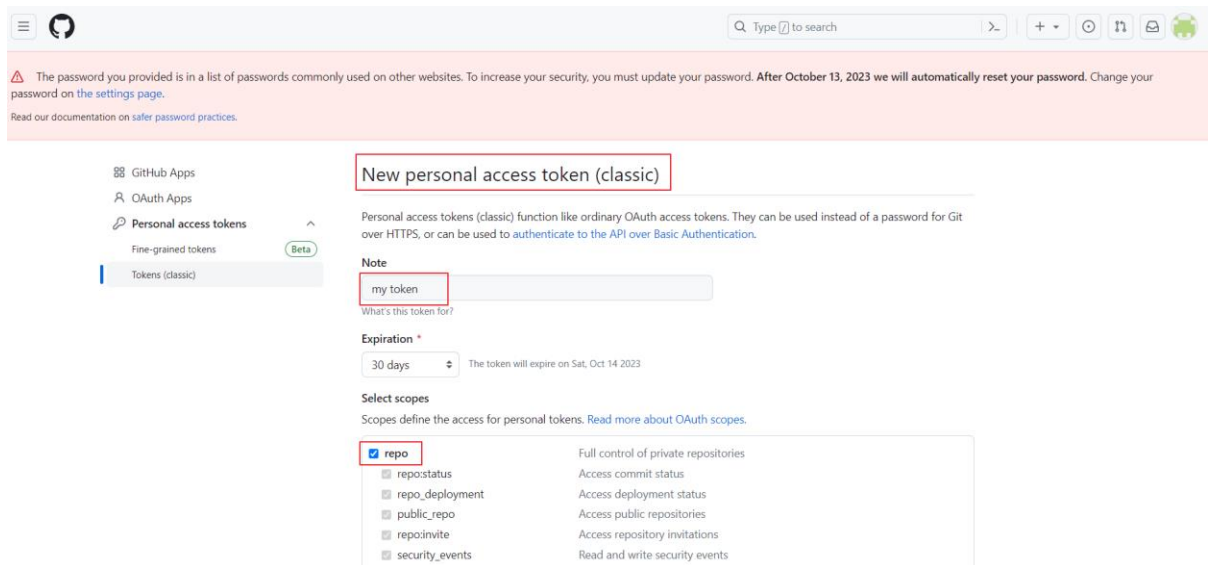
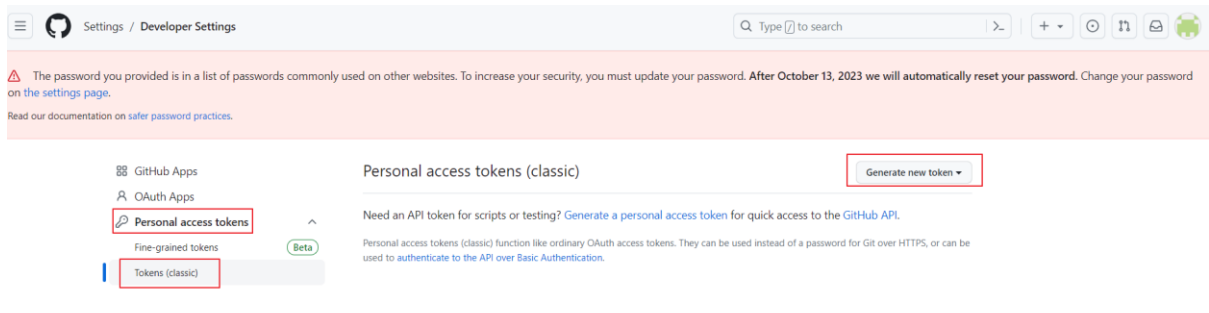
Github 사이트에 접속해서 로그인을 하고 우측 상단의 메뉴 중 Settings를 선택한다.



좌측 하단의 Developers settings를 선택하고 Personal access tokens를 클릭한다.



Tokens > 우측 Generate new token을 선택하고 필요한 항목을 채운다.

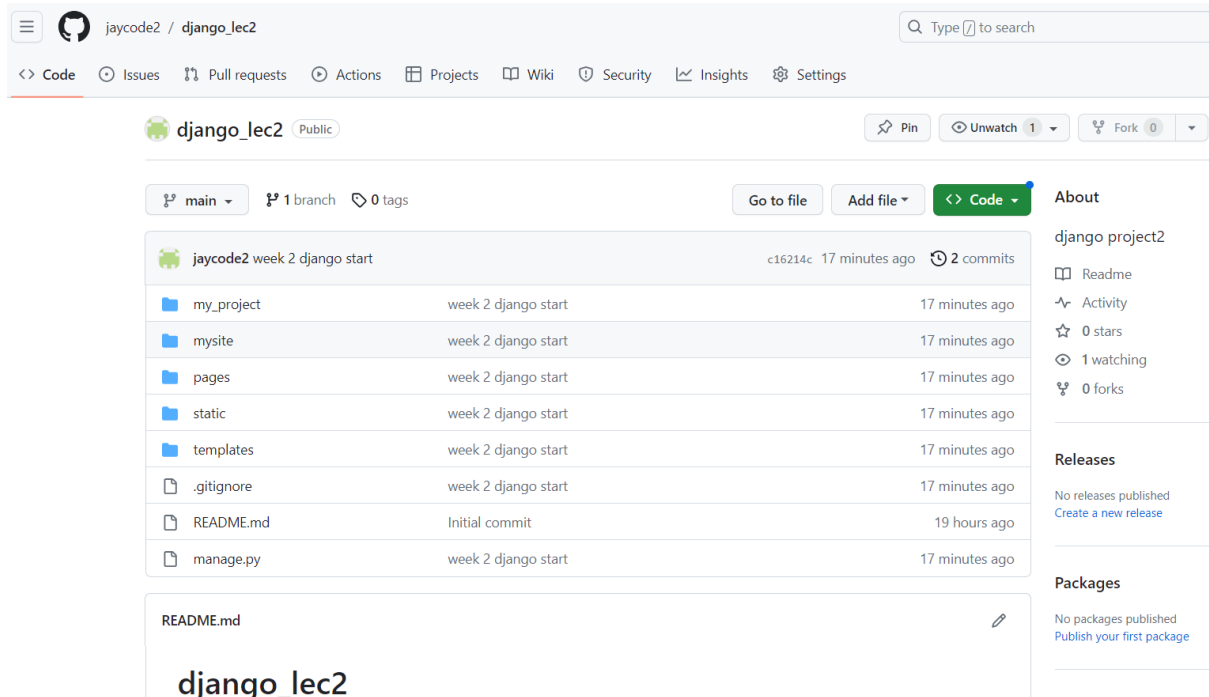


Token이 생성되면 따로 기록을 해두자. 만약 token이 기억나지 않는다면 token을 다시 생성하면 된다.

명령 프롬프트 창에 다시 git push를 입력하고 비밀번호에 생성된 토큰을 입력한다. 비밀번호 입력창에 토큰을 복사한 상태에서 우클릭을 하면 눈에는 보이지 않지만 입력이 된다.

```
(venv) C:\#github#django_lec2>git push
Logon failed, use ctrl+c to cancel basic credential prompt.
Username for 'https://github.com': jaycode2
Password for 'https://jaycode2@github.com':
Enumerating objects: 34, done.
Counting objects: 100% (34/34), done.
```

내 깃허브 경로에서 정상적으로 커밋이 되었는지 확인한다.



The screenshot shows a GitHub repository page for 'django_lec2' by user 'jaycode2'. The repository is public and has 1 branch and 0 tags. The file list shows the following files and folders:

File/Folder	Commit Message	Commit Hash	Time Ago
my_project	week 2 django start	c16214c	17 minutes ago
mysite	week 2 django start		17 minutes ago
pages	week 2 django start		17 minutes ago
static	week 2 django start		17 minutes ago
templates	week 2 django start		17 minutes ago
.gitignore	week 2 django start		17 minutes ago
README.md	Initial commit		19 hours ago
manage.py	week 2 django start		17 minutes ago

The README.md file is open, showing the title 'django_lec2'.

지금까지 django 개발을 위한 환경설정부터 데이터를 화면에 출력하는 기본적인 개발을 진행했다. 해당자료와 django 튜토리얼(<https://docs.djangoproject.com/en/4.0/intro/tutorial01/>), 공식문서를 활용하여 나만의 웹 페이지를 만들어보자(<https://docs.djangoproject.com/en/4.0/>). 앞으로 진행되는 차수에서는 추가 미션카드를 통해 완성도를 높여갈 예정이니 웹 페이지의 전체적인 틀을 잡는데 집중해 보자.