

# 장고 차근차근 시작하기

Second Edition

당신의 파이썬/장고 페이스메이커가 되겠습니다. ;)

EP31. 다양한 구동환경을 위한 settings/requirements.txt 분기

requirements.txt

# 의존성있는 라이브러리 관리

- 프로젝트 세팅할 때마다, 일일인 명시해서 설치하는 것은 너무 번거롭습니다.
  - 보통 설치할 라이브러리 개수만 해도 수백개.

# 예를 들어, 샘플 → 개발용

```
셸입력> pip3 install django django-debug-toolbar django-extensions django-bootstrap3 pilkit \
    django-imagekit django-polymorphic django-allauth django-taggit \
    django-taggit-templatetags pylibmc django-filter django-admin-honeypot \
    django-admin-tools feedparser lxml pyyaml pytz pillow wand qrcode django-bootstrap3 \
    django-crispy-forms django-widget-tweaks==1.4.1 jinja2 django-jinja requests \
    beautifulsoup4 nbconvert jupyter_client pdfwr pypdf2 reportlab django-ses
```

# 예를 들어, 샘플 → 배포용

```
셸입력> pip3 install django django-extensions django-bootstrap3 pilkit \
    django-imagekit django-polymorphic django-allauth django-taggit \
    django-taggit-templatetags pylibmc django-filter django-admin-honeypot \
    django-admin-tools feedparser lxml pyyaml pytz pillow wand qrcode django-bootstrap3 \
    django-crispy-forms django-widget-tweaks==1.4.1 jinja2 django-jinja requests \
    beautifulsoup4 nbconvert jupyter_client pdfwr pypdf2 reportlab django-ses \
    psycpg2 uwsgi
```

# requirements.txt

- pip에서는 설치할 패키지 목록을 파일을 통한 지정 지원
  - 일반적인 파일명이 requirements.txt
  - 다른 파일명/경로이어도 Don't care.

```
django==2.1.1  
django-debug-toolbar  
django-extensions  
django-bootstrap3
```

```
셸> pip install -r requirements.txt
```

# 실행환경에 따라 다양한 패키지 목록이 필요

- WHY?

- 실행환경 별로 필요한 라이브러리가 다를 수 있습니다.
  - 클라우드 별로, 혹은 DB별로 등등.
- 같은 프로젝트를 하는 개발팀/개발자마다 다른 라이브러리로 개발 중일 수도 있겠죠.

# requirements.txt를 만들어본다면?

다른 파일을 포함할 수 있습니다. -r 옵션

- 공통
- 개발용
- 서비스 2.0 개발용
- 배포용 (공통)
- 배포용 (AWS)
- 배포용 (Azure)
- 배포용 (Heroku)

# requirements.txt를 만들어본다면?

- requirements.txt
- reqs 디렉토리
  - common.txt : 공통
  - dev.txt : 현재 개발용
  - dev\_2.0.txt : 서비스 2.0 개발용
  - prod\_common.txt : 배포용 (공통)
  - prod\_aws.txt : 배포용 (AWS)
  - prod\_azure.txt : 배포용 (Azure)
  - prod\_heroku.txt : 배포용 (Heroku)

settings 모듈



# settings란?

- 다양한 프로젝트 설정을 담는 파이썬 소스 파일
  - 장고 앱 설정, DB 설정, 캐시 설정 등등
  - 디폴트 설정([django/conf/global\\_settings.py](https://github.com/django/django/blob/master/django/conf/global_settings.py)) 을 기본으로 깔고, 지정 settings를 통해 필요한 설정을 재정의
- 장고 프로젝트 구동 시에 필히 DJANGO\_SETTINGS\_MODULE 환경변수를 통해, settings의 위치를 알려줘야합니다.

manage.py x

```
1  #!/usr/bin/env python
2  import os
3  import sys
4
5  if __name__ == '__main__':
6      os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'askcompany.settings')
7      try:
8          from django.core.management import execute_from_command_line
9      except ImportError as exc:
10         raise ImportError(
```

wsgi.py x

```
12  from django.core.wsgi import get_wsgi_application
13
14  os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'askcompany.settings')
15
16  application = get_wsgi_application()
```

# setdefault 동작은?

os.environ은 dict과 유사한 인터페이스

```
dict.setdefault(key, default=None)
```

```
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'askcompany.settings')
```

# 위 코드는 아래 코드와 같은 동작을 합니다.

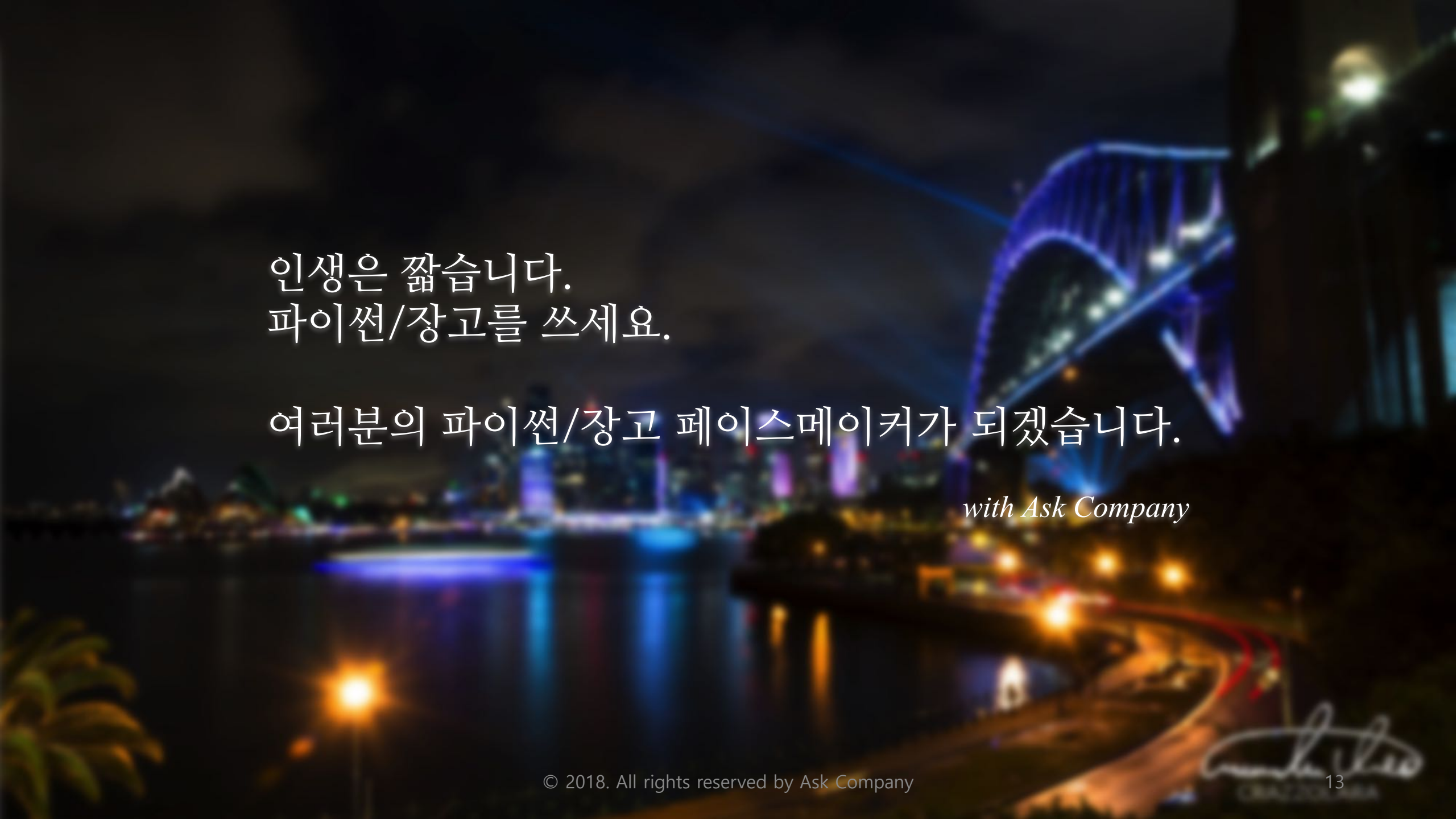
```
if 'DJANGO_SETTINGS_MODULE' not in os.environ:  
    os.environ['DJANGO_SETTINGS_MODULE'] = 'askcompany.settings'
```

# settings를 지정하는 2가지 방법

- 1) DJANGO\_SETTINGS\_MODULE 환경변수로 지정하기
  - 주의) OS마다/배포하는 방법마다 환경변수 세팅방법이 다릅니다.
  - 별도로 지정하지 않으면, manage.py/wsgi.py에 세팅된 설정값이 적용
- 2) manage.py 명령에서 --settings 옵션을 통해 지정하기
  - 환경변수 설정에 우선합니다.
  - 셸 > python manage.py 명령 `--settings=askcompany.settings.prod_heroku`

# settings를 파이썬 패키지로 만들기

- 주의
  - settings.py 내 BASE\_DIR 설정은 상대경로로 프로젝트 ROOT 경로를 계산.
- 이전
  - 프로젝트/settings.py
- 이후
  - 프로젝트/settings/
    - \_\_init\_\_.py
    - common.py
    - dev.py
    - prod\_common.py
    - prod\_aws.py
    - prod\_heroku.py

A nighttime photograph of a cityscape featuring a bridge with blue lights and a curved road with yellow lights. The background is dark with some city lights visible.

인생은 짧습니다.  
파이썬/장고를 쓰세요.

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

*with Ask Company*