

# 플라스크 웹 프로그램

# 플라스크 개요

---

## 플라스크의 개요

- 파이썬으로 만든 마이크로 웹 프레임워크
- 기본적으로 DB기능이 포함되어 있지 않는 등 최소한의 기능만 제공
- 최소한의 규약만 있어서 앱 구성도 자유롭게 결정
- 데이터베이스 기능 등 확장 기능을 많이 지원
- 확장 기능은 플라스크 자체에 구현되어 있는 것처럼 간단하게 이용
- 필요에 따라 다양한 확장 기능을 추가해서 크고 작은 웹 앱 개발을 비롯  
해 다양한 모델을 만들 수 있게 설계

# 플라스크 개요

## 파이썬 웹 프레임워크 비교

### ◦ 파이썬에서 이용할 수 있는 웹 프레임워크

프레임워크	공식 사이트	라이선스	초기 개발자	최초 릴리스	템플릿 엔진	O/R 매퍼
장고	<a href="http://www.djangoproject.com">www.djangoproject.com</a>	BSD License	Adrian Holovaty, Simon Willison	2005년	Django Template	Django O/R 매퍼
플라스크	<a href="http://palletsprojects.com/p/flask">palletsprojects.com/p/flask</a>	BSD license	Armin Ronacher	2010년	Jinja2	없음
보틀	<a href="http://bottlepy.org/docs/dev">bottlepy.org/docs/dev</a>	MIT License	Marcel Hellkamp	2009년	Simple Template Engine	없음
FastAPI	<a href="http://fastapi.tiangolo.com">fastapi.tiangolo.com</a>	MIT License	Sebastian Ramirez	2018년	Jinja2	없음

# 플라스크 개요

## 플라스크 설치하기

- pip install 명령어를 실행
- pip list 명령어를 사용

패키지	설명
click	명령어 라인용 프레임워크. 플라스크의 커스텀 명령어를 사용한다.
itsdangerous	안전하게 데이터를 서명해 데이터의 정합성을 확보한다. 플라스크의 세션이나 쿠키(Cookie)를 보호하기 위해서 사용한다.
Jinja2	디폴트 HTML 템플릿 엔진. 다른 템플릿 엔진을 사용할 수도 있다.
MarkupSafe	인젝션 공격을 회피하기 위해 템플릿을 렌더링할 때에 신뢰할 수 없는 입력을 취소한다.
Werkzeug	WSGI 툴킷으로 플라스크의 코어 구현은 Werkzeug( <a href="http://werkzeug.palletsprojects.com">werkzeug.palletsprojects.com</a> )를 바탕으로 만들어져 있다.

# 플라스크 개요

## 플라스크 설치하기

- 최초 플라스크 설치 : pip install 명령어를 실행(아니콘다 설치시 자동 Flask 설치되어 있음)
- 작업 명령창 실행 : Anaconda Prompt **관리자 권한**으로 실행
- 작업 디렉토리 작성 : mkdir flaskbook



```
관리자: Anaconda Prompt

(base) C:\WINDOWS\system32>cd ..
(base) C:\Windows>cd ..
(base) C:\>mkdir flaskbook
(base) C:\>cd flaskbook
(base) C:\flaskbook>dir
C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: 9CB4-C50F

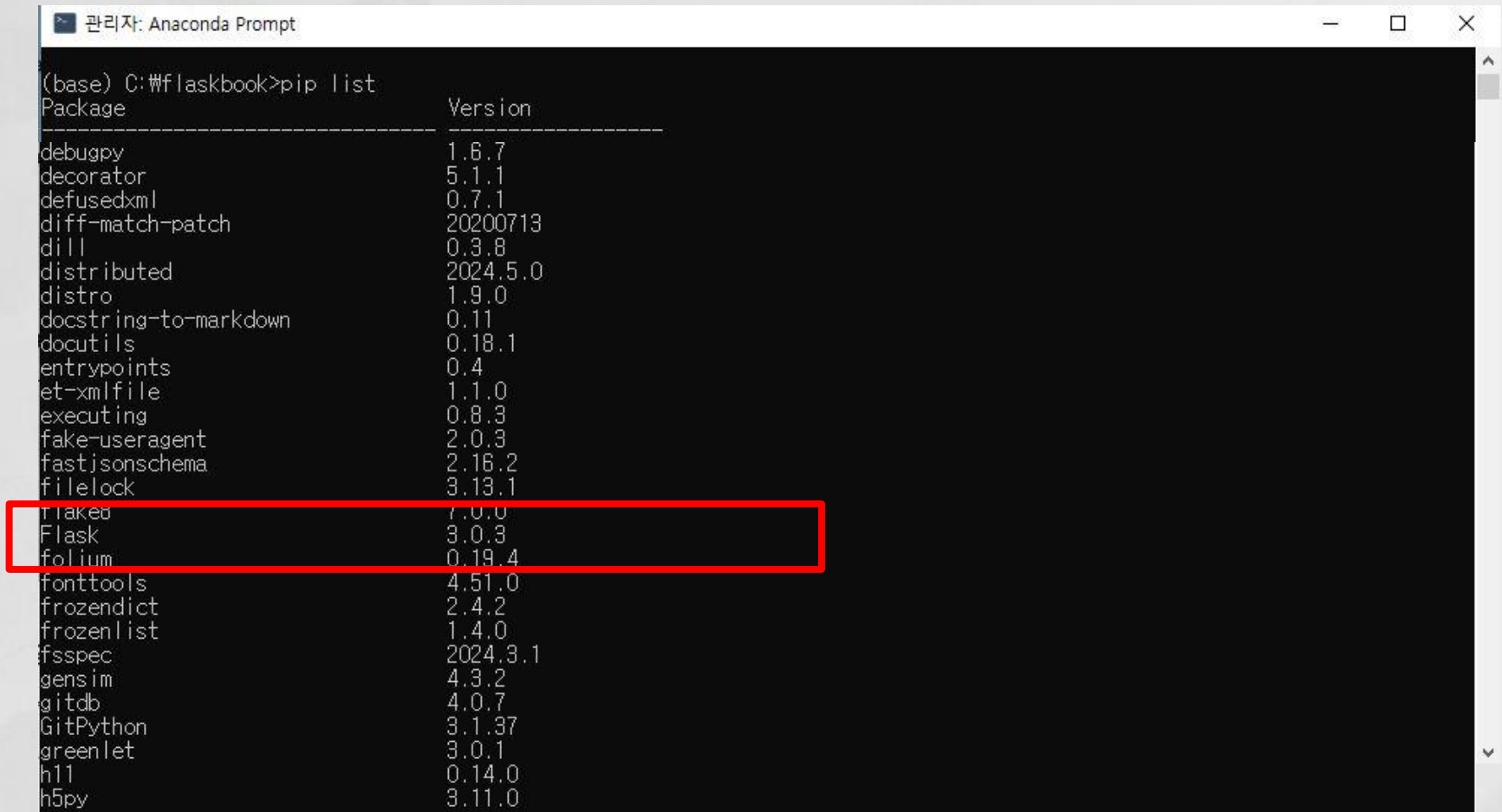
C:\flaskbook 디렉터리

2025-01-31 오전 01:41 <DIR> .
2025-01-31 오전 01:41 <DIR> ..
                0개 파일                0 바이트
                2개 디렉터리   7,078,297,600 바이트 남음
```

# 플라스크 개요

## 플라스크 설치하기

- o pip list 명령어를 사용



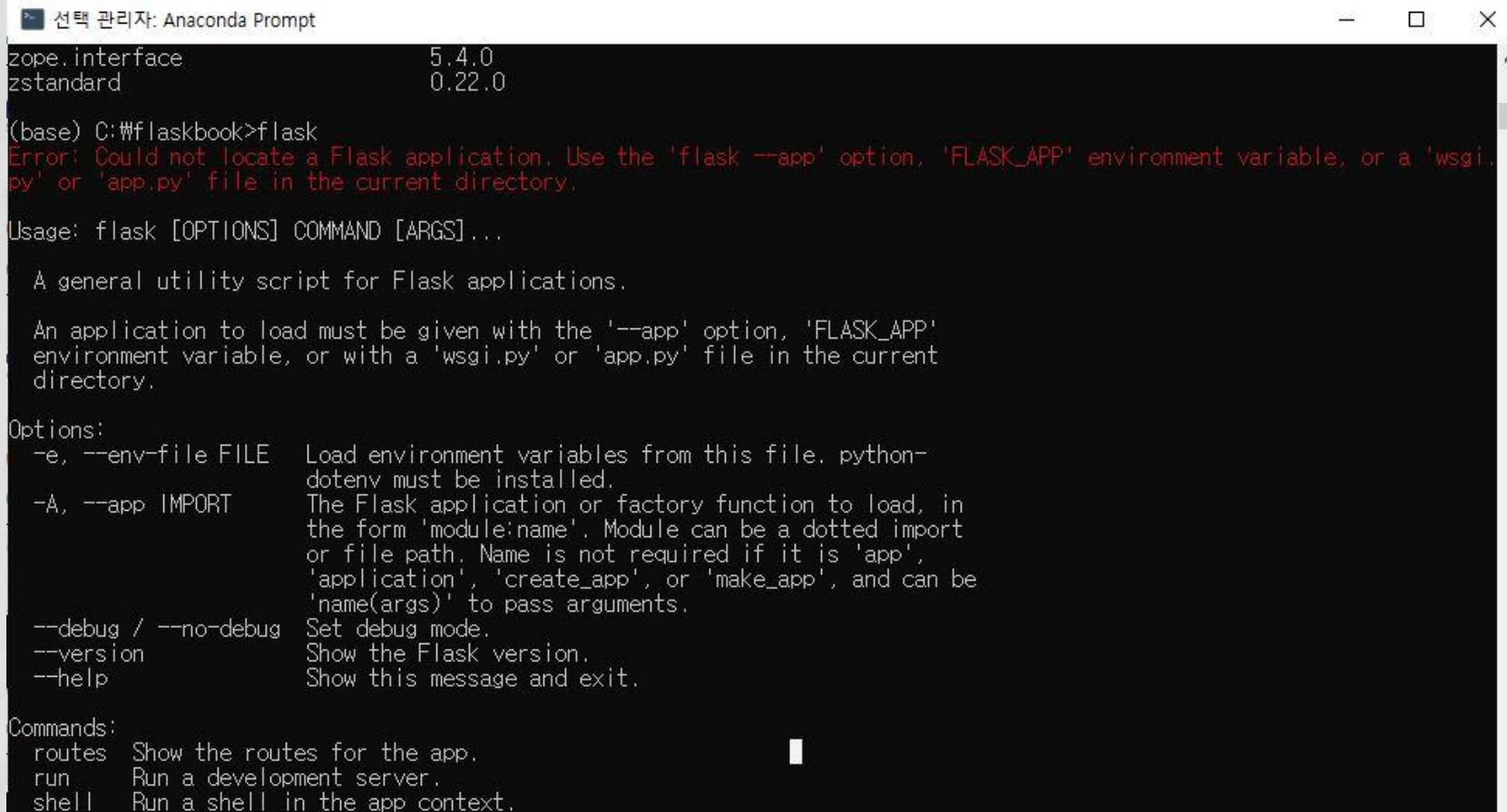
```
관리자: Anaconda Prompt

(base) C:\wflaskbook>pip list
Package                                Version
-----
debugpy                                1.6.7
decorator                              5.1.1
defusedxml                             0.7.1
diff-match-patch                       20200713
dill                                    0.3.8
distributed                            2024.5.0
distro                                  1.9.0
docstring-to-markdown                  0.11
docutils                               0.18.1
entrypoints                            0.4
et-xmlfile                             1.1.0
executing                              0.8.3
fake-useragent                         2.0.3
fastjsonschema                         2.16.2
filelock                               3.13.1
Flake8                                 7.0.0
Flask                                  3.0.3
folium                                 0.19.4
fonttools                              4.51.0
frozendict                             2.4.2
frozenlist                             1.4.0
fsspec                                  2024.3.1
gensim                                  4.3.2
gitdb                                   4.0.7
GitPython                              3.1.37
greenlet                               3.0.1
h11                                     0.14.0
h5py                                    3.11.0
```

# 플라스크 개요

## 플라스크 명령어

- flask 또는 flask --help 명령어로 옵션을 확인



```
선택 관리자: Anaconda Prompt
zope.interface          5.4.0
zstandard               0.22.0

(base) C:\flaskbook>flask
Error: Could not locate a Flask application. Use the 'flask --app' option, 'FLASK_APP' environment variable, or a 'wsgi.py' or 'app.py' file in the current directory.

Usage: flask [OPTIONS] COMMAND [ARGS]...

  A general utility script for Flask applications.

  An application to load must be given with the '--app' option, 'FLASK_APP' environment variable, or with a 'wsgi.py' or 'app.py' file in the current directory.

Options:
  -e, --env-file FILE      Load environment variables from this file. python-dotenv must be installed.
  -A, --app IMPORT          The Flask application or factory function to load, in the form 'module:name'. Module can be a dotted import or file path. Name is not required if it is 'app', 'application', 'create_app', or 'make_app', and can be 'name(args)' to pass arguments.
  --debug / --no-debug     Set debug mode.
  --version                 Show the Flask version.
  --help                    Show this message and exit.

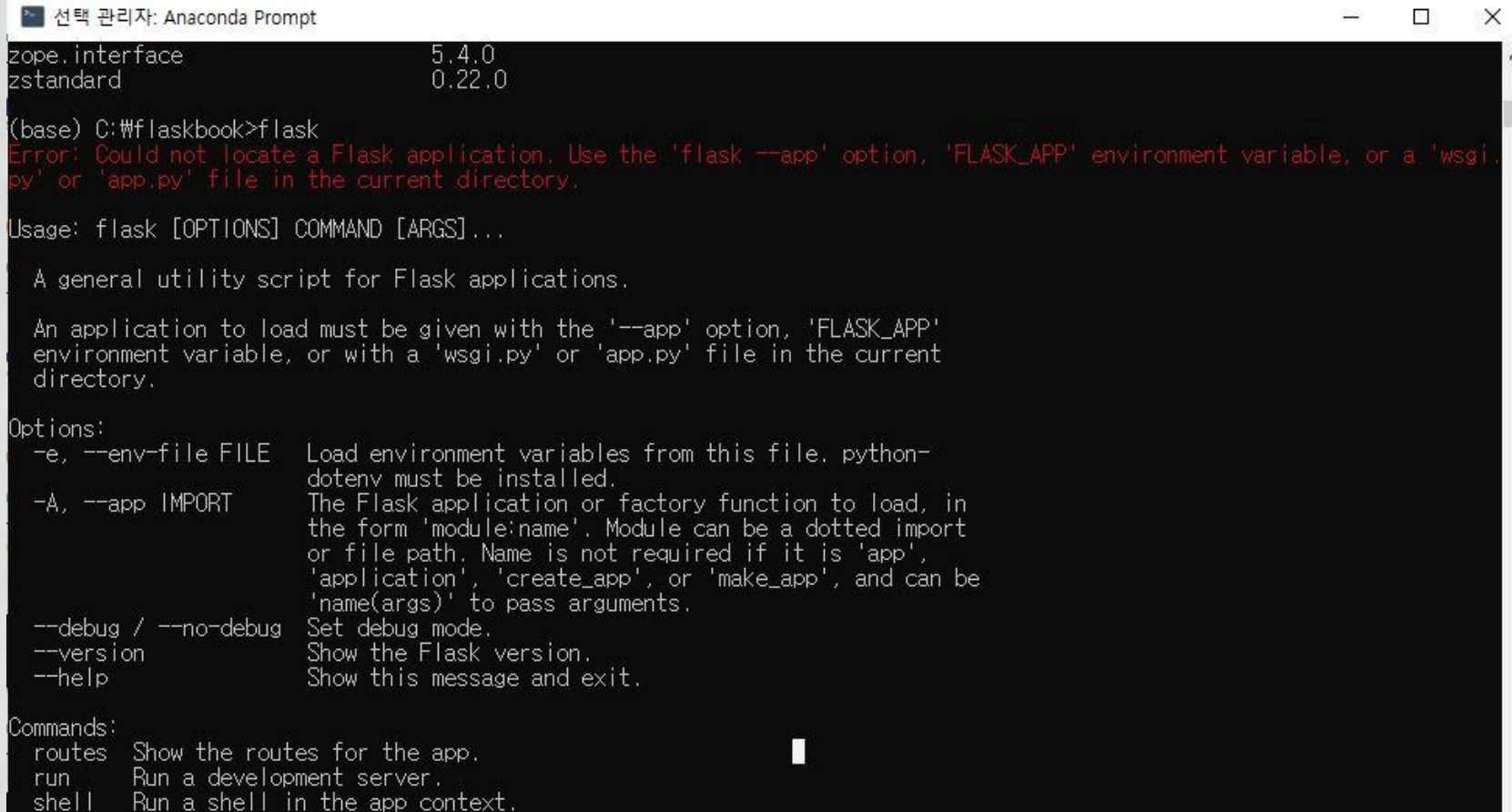
Commands:
  routes  Show the routes for the app.
  run     Run a development server.
  shell   Run a shell in the app context.
```



# 플라스크 개요

## 플라스크 명령어

- flask 또는 flask --help 명령어로 옵션을 확인



```
선택 관리자: Anaconda Prompt
zope.interface      5.4.0
zstandard           0.22.0

(base) C:\flaskbook>flask
Error: Could not locate a Flask application. Use the 'flask --app' option, 'FLASK_APP' environment variable, or a 'wsgi.py' or 'app.py' file in the current directory.

Usage: flask [OPTIONS] COMMAND [ARGS]...

  A general utility script for Flask applications.

  An application to load must be given with the '--app' option, 'FLASK_APP' environment variable, or with a 'wsgi.py' or 'app.py' file in the current directory.

Options:
  -e, --env-file FILE  Load environment variables from this file. python-dotenv must be installed.
  -A, --app IMPORT      The Flask application or factory function to load, in the form 'module:name'. Module can be a dotted import or file path. Name is not required if it is 'app', 'application', 'create_app', or 'make_app', and can be 'name(args)' to pass arguments.
  --debug / --no-debug  Set debug mode.
  --version             Show the Flask version.
  --help               Show this message and exit.

Commands:
  routes  Show the routes for the app.
  run     Run a development server.
  shell   Run a shell in the app context.
```



# 플라스크 개요

## 플라스크 명령어

### ◦ flask run --help 명령어로 옵션을 확인

```
선택 관리자: Anaconda Prompt
(base) C:\wflaskbook>flask run --help
Usage: flask run [OPTIONS]

Run a local development server.

This server is for development purposes only. It does not provide the
stability, security, or performance of production WSGI servers.

The reloader and debugger are enabled by default with the '--debug' option.

Options:
  --debug / --no-debug           Set debug mode.
  -h, --host TEXT                The interface to bind to.
  -p, --port INTEGER            The port to bind to.
  --cert PATH                   Specify a certificate file to use HTTPS.
  --key FILE                    The key file to use when specifying a
                                certificate.
  --reload / --no-reload        Enable or disable the reloader. By default
                                the reloader is active if debug is enabled.
  --debugger / --no-debugger    Enable or disable the debugger. By default
                                the debugger is active if debug is enabled.
  --with-threads / --without-threads
                                Enable or disable multithreading.
  --extra-files PATH            Extra files that trigger a reload on change.
                                Multiple paths are separated by ';'.
  --exclude-patterns PATH       Files matching these fnmatch patterns will
                                not trigger a reload on change. Multiple
                                patterns are separated by ';'.
  --help                        Show this message and exit.
```

# 플라스크 개요

## 플라스크 명령어

- flask run 명령어 실행(또는 flask --debug run) 하면 웹 서버가 실행
- <http://127.0.0.1:5000> 에서 웹서버 실행
- --host와 --port 옵션으로 호스트(서버)와 포트를 지정해서 실행

옵션	내용
-h 또는 --host	호스트를 지정한다.
-p 또는 --port	포트를 지정한다.
--reload --no-reload	오토 리로드를 on/off한다. 코드를 편집할 때 자동으로 반영시키려면 on으로 한다. 디버그 모드 시에는 기본적으로 on이 된다.
--debugger --no-debugger	디버거를 on/off한다. 디버그 모드일 때는 기본적으로 on이 된다.
--help	명령어 옵션을 표시한다.

# 플라스크 개요

## 플라스크 명령어

- flask routes 명령어 실행하면 app의 라우팅 정보를 출력
- 라우팅이란 요청한 곳의 URL과 실제로 처리하는 함수를 연결하는 작업
- 루트를 추가하여 flask routes 명령어를 실행하면 추가한 루트의 연결 정보를 확인
- 엔드포인트(Endpoint)는 일반적으로 API에 접근하기 위한 URL을 가리킴
- 플라스크에서는 URI와 연결된 함수명 또는 함수에 붙인 이름을 가리

Endpoint	Methods	Rule
index	GET	/
static	GET	/static/<path:filename>

항목	설명
Endpoint	URL에 접근할 때 실행할 함수 또는 지정한 이름. static은 정적 파일용의 엔드포인트로, 항상 고정으로 존재한다.
Methods	사용할 HTTP 메서드. 지정이 없는 경우는 GET이 기본으로 된다.
Rule	사용할 URL의 규칙

# 플라스크 개요

---

## 플라스크 명령어

- flask shell 명령어 실행하면 플라스크 app의 컨텍스트(실행 환경)에서 파이썬 인터랙티브 셸을 사용
- 디버깅이나 테스트를 할 때에 유용

# 플라스크 개요

---

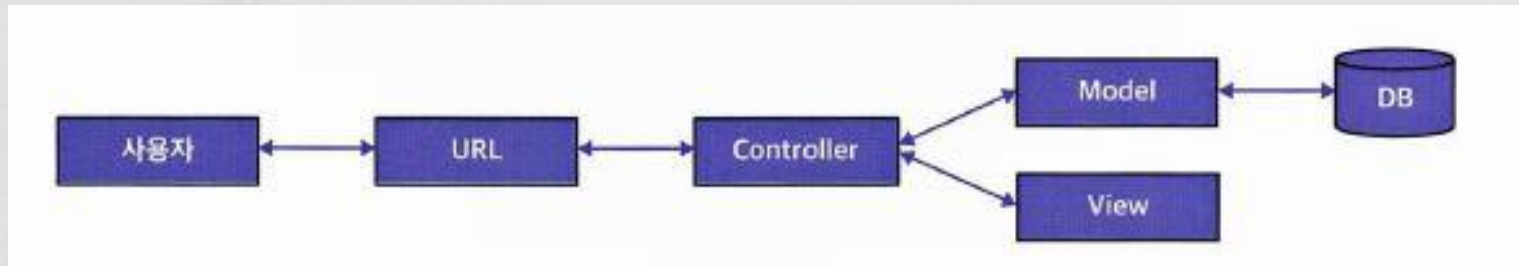
## 플라스크 명령어

- flask shell 명령어 실행하면 플라스크 app의 컨텍스트(실행 환경)에서 파이썬 인터랙티브 셸을 사용
- 디버깅이나 테스트를 할 때에 유용

# 플라스크 개요

## MVC(Model, View, Control) 모델

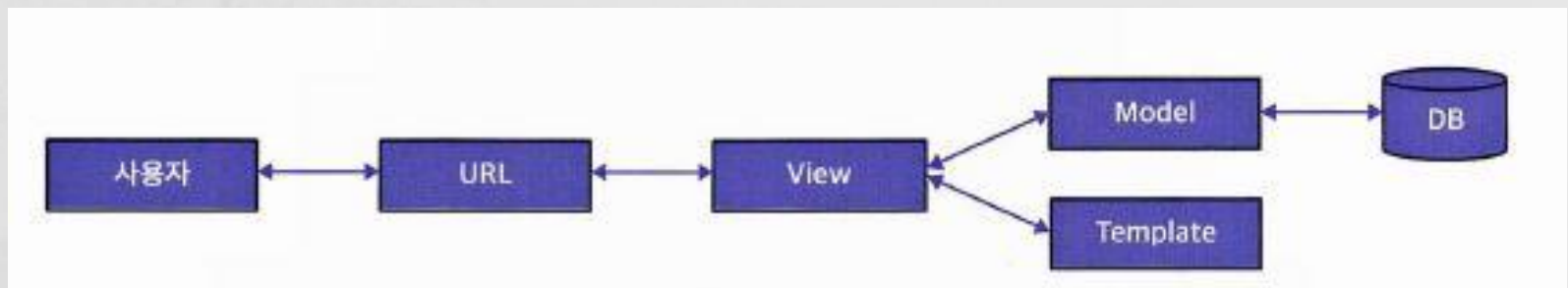
- 일반적인 MVC 패턴
- 개발 시 일반적으로 언급되는 MVC(Model-View-Controller) 패턴이란 데이터(Model), 사용자 인터페이스(View), 데이터를 처리하는 로직(Controller)을 구분해서 한 요소가 다른 요소들에 영향을 주지 않도록 설계하는 방식



# 플라스크 개요

## MVT(Model, View, Template) 모델

- 플라스크는 사용자 인터페이스를 가진 app을 구현하기 위한 디자인 패턴으로서 MVT ( Model, View, Template ) 모델을 채용
- Model : 실무(비즈니스)로직을 담당(데이터)
- View : 입력을 받아 Model과 Template을 제어
- Template : 입출력을 담당(사용자 인터페이스 UI)





# 플라스크 작성

## 작업 디렉터리 만들기

- flaskbook 작업 디렉터리에 여러 app을 추가하기 위해 apps폴더와

c:\flaskbook\apps\minimalapp 디렉터를 작성

```
관리자: Anaconda Prompt

(base) C:\flaskbook>dir
C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: 9CB4-C50F

C:\flaskbook 디렉터리

2025-01-31 오전 01:41 <DIR> .
2025-01-31 오전 01:41 <DIR> ..
0개 파일 0 바이트
2개 디렉터리 7,118,237,696 바이트 남음

(base) C:\flaskbook>mkdir apps
(base) C:\flaskbook>cd apps
(base) C:\flaskbook\apps>mkdir minimalapp
(base) C:\flaskbook\apps>cd minimalapp
(base) C:\flaskbook\apps\minimalapp>dir
C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: 9CB4-C50F

C:\flaskbook\apps\minimalapp 디렉터리

2025-01-31 오전 02:40 <DIR> .
2025-01-31 오전 02:40 <DIR> ..
0개 파일 0 바이트
2개 디렉터리 7,117,451,264 바이트 남음

(base) C:\flaskbook\apps\minimalapp>
```

```
(base) C:\>tree flaskbook
폴더 PATH의 목록입니다.
볼륨 일련 번호가 00000085 9CB4:C50F입니다.
C:\FLASKBOOK
├── apps
│   └── minimalapp
```

# 플라스크 작성

## 웹 서버 작성

- o c:\flaskbook\apps\minimalapp 디렉터리에 app.py 작성

```
app.py - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

# flask 클래스를 import 한다
from flask import Flask

# flask 클래스를 인스턴스화한다
app = Flask(__name__)

# URL과 실행할 함수를 매핑한다
@app.route("/")
def index():
    return "Hello, FlaskBook!"
```

# 플라스크 작성

## 웹 서버 작성

- c:\flaskbook\apps\minimalapp 디렉터리에서
- flask run 명령어로 실행

```
(base) C:\>cd flaskbook
(base) C:\flaskbook>cd apps
(base) C:\flaskbook\apps>cd minimalapp
(base) C:\flaskbook\apps\minimalapp>dir
C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: 9CB4-C50F

C:\flaskbook\apps\minimalapp 디렉터리

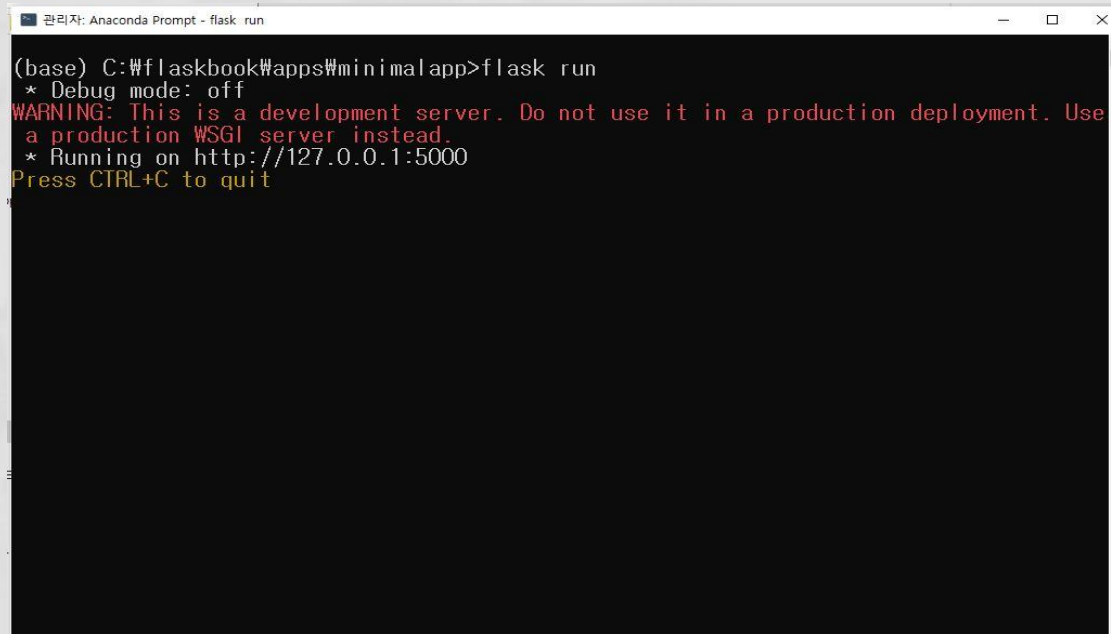
2025-01-31 오전 02:55 <DIR>          .
2025-01-31 오전 02:55 <DIR>          ..
2025-01-31 오전 02:53          236 app.py
                   1개 파일           236 바이트
                   2개 디렉터리 7,122,976,768 바이트 남음

(base) C:\flaskbook\apps\minimalapp>
```

# 플라스크 작성

## 웹 서버 실행

- c:\flaskbook\apps\minimalapp 디렉터리에서
- flask run 명령어서 실행
- (base) C:\flaskbook\apps\minimalapp>flask run

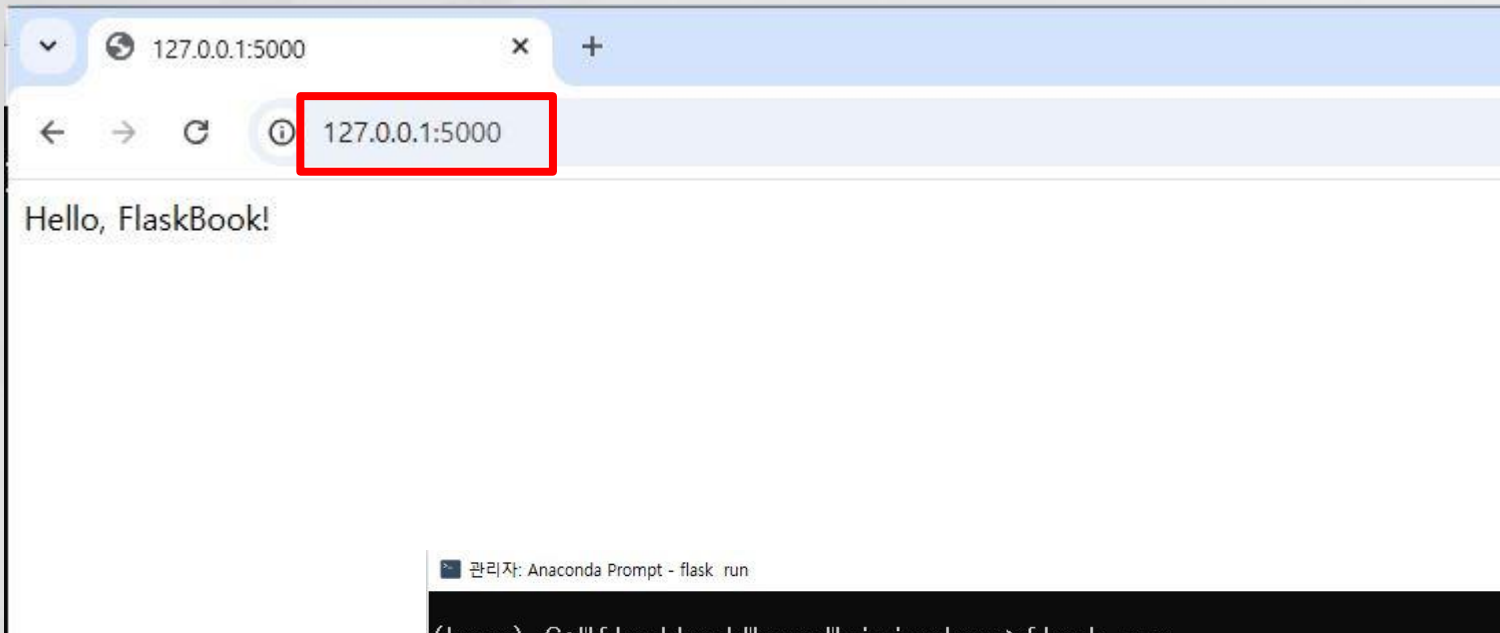


```
관리자: Anaconda Prompt - flask run
(base) C:\flaskbook\apps\minimalapp>flask run
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use
a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

# 플라스크 작성

## 브라우저에서 실행

- 브라우저에서 <http://127.0.0.1:5000> URL에 접근



```
관리자: Anaconda Prompt - flask run
(base) C:\flaskbook\apps\minimalapp>flask run
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use
a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [31/Jan/2025 03:13:31] "GET / HTTP/1.1" 200 -
```

# 플라스크 작성

---

## 라우팅 이용하기

- 라우팅이란 요청한 곳의 URI와 실제로 처리를 담당하는 함수를 연결하는 작업
- 플라스크에서는 함수의 앞에 데코레이터라는 함수 `@app.route()`를 추가함으로써 루트
- `app.py`에 'Hello, World!'를 출력하는 라우팅을 추가

# 플라스크 작성

## 라우팅 이용하기

- o c:\flaskbook\wapps\minimalapp\app.py에 추가 작성

```
app.py - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

# flask 클래스를 인스턴스화한다
app = Flask(__name__)

# URL과 실행할 함수를 매핑한다
@app.route("/")
def index():
    return "Hello, FlaskBook!"

# 루트 추가하기
@app.route("/hello")
def hello():
    return "Hello, World!"
```



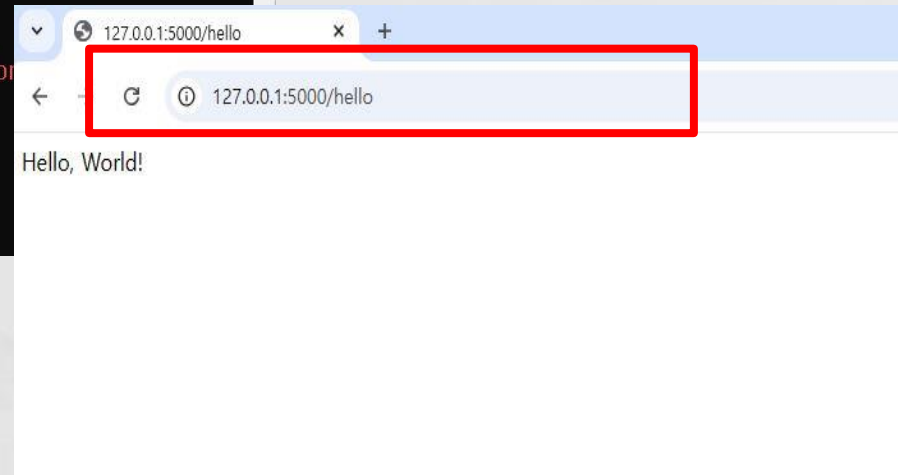
# 플라스크 작성

## 라우팅 이용하기

- (base) C:\flaskbook\apps\minimalapp>flask run
- 브라우저에서 <http://127.0.0.1:5000/hello> URL에 접근

```
관리자: Anaconda Prompt - flask run
(base) C:\flaskbook\apps\minimalapp>flask run
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use
a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [31/Jan/2025 03:13:31] "GET / HTTP/1.1" 200 -

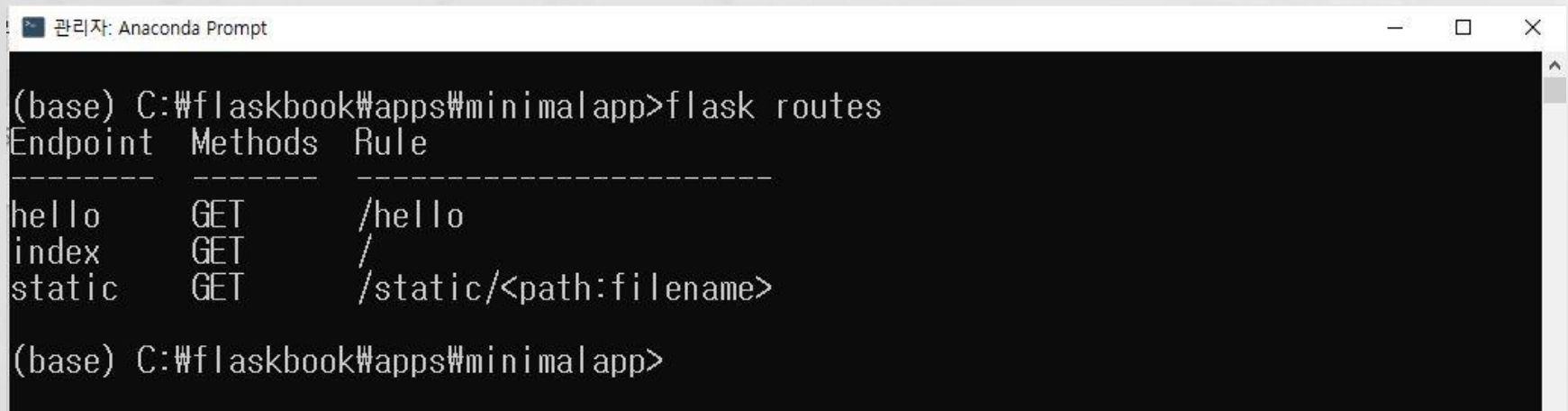
(base) C:\flaskbook\apps\minimalapp>flask run
* Debug mode: off
WARNING: This is a development server. Do not use it in a production
a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [31/Jan/2025 03:38:23] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [31/Jan/2025 03:38:36] "GET /hello HTTP/1.1" 200 -
```



# 플라스크 작성

## 라우팅 이용하기

- flask routes 명령어로 라우팅 정보를 확인
- (base) C:\flaskbook\apps\minimalapp>flask routes



```
(base) C:\flaskbook\apps\minimalapp>flask routes
Endpoint    Methods    Rule
-----
hello      GET       /hello
index      GET       /
static     GET       /static/<path:filename>

(base) C:\flaskbook\apps\minimalapp>
```

# 플라스크 작성

---

## 라우팅 이용하기

- HTTP 메서드는 클라이언트가 서버에 대해서 요청을 송신할 때에 서버에게 실행하기를 바라는 조작을 전달하기 위해 사용
- HTML 폼에서는 GET 메서드나 POST 메서드
- GET 메서드는 검색 등 리소스를 얻는 경우에 이용
- 폼이 아닌 평소의 브라우징도 GET 메서드 사용
- POST 메서드는 로그인이나 문의 송신 등 폼의 값을 등록.갱신하는 경우에 이용

# 플라스크 작성

---

## 라우팅 이용하기

- 플라스크의 엔드포인트에 이름 붙이기
- 엔드포인트(Endpoint)는 일반적으로 API에 접근하기 위한 URI를 가리킴
- 플라스크에서는 URI 와 연결된 함수명 또는 함수에 붙인 이름을 가리킴
- 엔드포인트명은 기본적으로 `@app.route`로 수식된 함수명
- `@app.route("/", endpoint="endpoint-name")`

```
# 엔드포인트 명 추가하기
@app.route("/hello", methods=["GET"], endpoint="hello-endpoint")
def hello():
    return "Hello, World"
```

# 플라스크 작성

## 라우팅 이용하기

- 플라스크의 엔드포인트에 이름 붙이기
- flask routes 명령어를 실행하면 엔드포인트명 확인

```
관리자: Anaconda Prompt
(base) C:\flaskbook\apps\minimalapp>flask routes
Endpoint      Methods Rule
-----
hello-endpoint GET    /hello
index         GET    /
static        GET    /static/<path:filename>
```

Rule                      Methods                      Endpoint

↓                              ↓                              ↓

```
@app.route("/hello", methods=["GET", ], endpoint="hello-endpoint")
def hello():
```

↑  
엔드포인트를 지정하지 않으면 함수명이 엔드포인트명이 된다

# 플라스크 작성

## 라우팅 이용하기

- Rule에 변수 지정하기
- @app.route 데코레이터의 Rule에 변수를 지정
- 변수는 <변수명> 형식으로 지정

```
# Rule에 변수 지정하기
@app.route("/hello/<name>", methods=["GET"], endpoint="hello-endpoint")
def hello(name):
    return f"Hello, {name}"
```

- 옵션에서 컨버터라는 타입 정의를 이용해 <컨버터: 변수명>

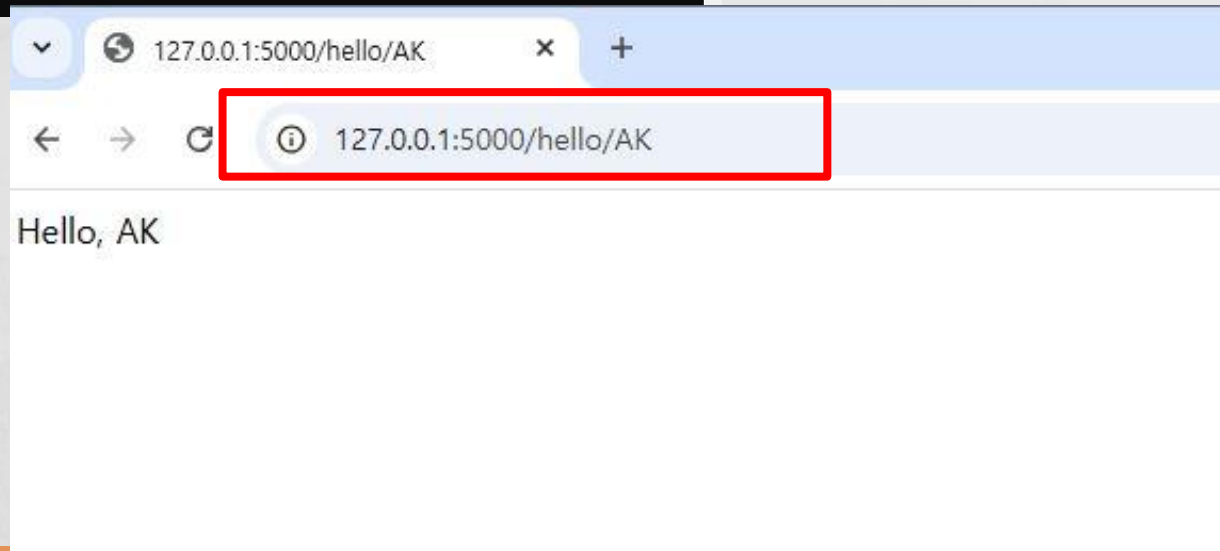
컨버터의 타입	설명
string	슬래시가 없는 텍스트
int	양의 정수
float	양의 부동 상수점
path	슬래시가 있는 텍스트를 허용
uuid	UUID 문자열

# 플라스크 작성

## 라우팅 이용하기

- Rule에 변수 지정하기
- 브라우저에서 <http://127.0.0.1:5000/hello/AK> URL에 접근

```
(base) C:\#flaskbook#apps#minimalapp>flask run
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use
a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [31/Jan/2025 04:19:52] "GET /hello HTTP/1.1" 200 -
127.0.0.1 - - [31/Jan/2025 04:20:02] "GET /hello/AK HTTP/1.1" 200 -
```



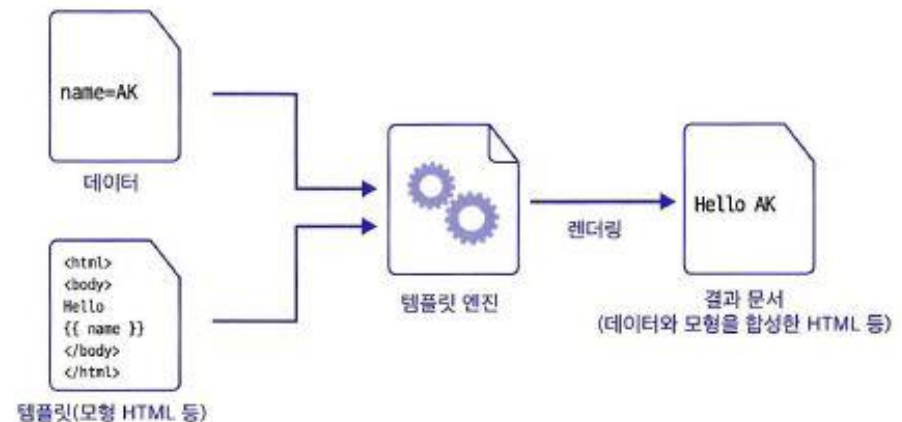


# 플라스크 작성

## 템플릿 엔진 사용하기

- 템플릿 엔진은 템플릿이라는 모형과 데이터를 합성하여 문서를 출력하는 소프트웨어
- 플라스크의 기본 템플릿 엔진은 Jinja2
- 템플릿 엔진을 사용하여 HTML을 렌더링 (그리기)하려면

**render\_template** 함수를 이용



# 플라스크 작성

---

## 템플릿 엔진 사용하기

- 템플릿을 만들고 app 측에서는 `render_template` 함수에 템플릿의 이름과 키워드 인수로서 변수를 넘겨 이용
- `render_template` 함수를 사용해서 템플릿을 이용
- 렌더링을 위해 필요한 템플릿으로서 **HTML 파일을 작성**
- `templates` 디렉토리를 생성하고 거기에 `index.html`을 만듦
- **템플릿에서** `{{ 변수명 }}`이라고 기술하여 Jinja2가 변수를 전개하여 **렌더링**

# 플라스크 작성

## 템플릿 엔진 이용하기

- o **templates** 디렉터리를 생성하고 거기에 **index.html**을 만듦

```
선택 관리자: Anaconda Prompt

(base) C:\flaskbook\apps\minimalapp>mkdir templates

(base) C:\flaskbook\apps\minimalapp>dir
C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: 9CB4-C50F

C:\flaskbook\apps\minimalapp 디렉터리

2025-01-31 오전 04:38 <DIR> .
2025-01-31 오전 04:38 <DIR> ..
2025-01-31 오전 03:36      342 app - 복사본.py
2025-01-31 오전 04:30      650 app.py
2025-01-31 오전 04:38 <DIR> templates
2025-01-31 오전 04:29 <DIR> __pycache__
                2개 파일              992 바이트
                4개 디렉터리  7,099,211,776 바이트 남음
```

# 플라스크 작성

## 템플릿 엔진 이용하기

- o (base) C:\flaskbook\apps\minimalapp\templates/index.html

```
*index.html - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

<!DOCTYPE html>
<html lang="ko">
  <head>
    <meta charset="UTF-8" />
    <title>Name</title>
  </head>
  <body>
    <h1>Name: {{ name }}</h1>
  </body>
</html>
```

# 플라스크 작성

---

## 템플릿 엔진 이용하기

- (base) C:\flaskbook\apps\minimalapp\app.py 수정

```
# flask 클래스를 import 한다
from flask import Flask, render_template
```

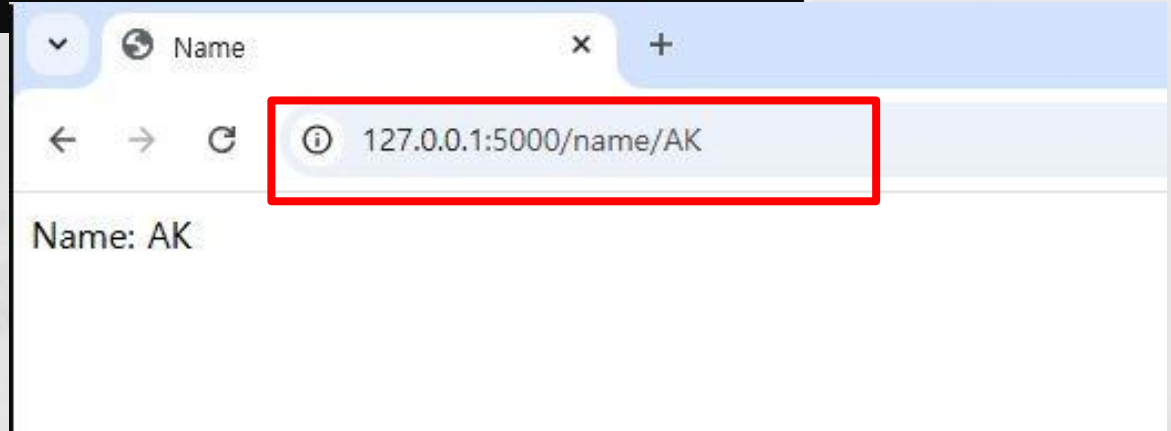
```
# Rule에 변수 지정하기
@app.route("/name/<name>")
def show_name(name):
    return render_template("index.html", name=name)
```

# 플라스크 작성

## 템플릿 엔진 이용하기

- (base) C:\flaskbook\apps\minimalapp>flask run
- 브라우저에서 <http://127.0.0.1:5000/name/AK> URL에 접근

```
(base) C:\flaskbook\apps\minimalapp>flask run
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use
a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [31/Jan/2025 04:52:35] "GET /name/AK HTTP/1.1" 200 -
```



# 플라스크 작성

## 템플릿 엔진 이용하기(Jinja2)

### ◦ 변수의 값 출력하기

- 변수의 값을 표시하려면 {{ }}을 사용

```
<h1>Name: {{ name }}</h1>
```

### ◦ 조건식 if문의 사용법

- 조건식 if 문을 이용하려면 {% %}

```
{% if name %}  
<h1>Name: {{ name }}</h1>  
{% else %}  
<h1>Name:</h1>  
{% endif %}
```

### ◦ 반복 for 문의 사용법

- 반복 for 문을 이용할 때에도 {% %}를 사용

```
<ul>  
  {% for user in users %}  
    <li><a href="{{ user.url }}">{{ user.username }}</a></li>  
  {% endfor %}  
</ul>
```



# 정리

---

## 정리

- 플라스크 개요
- 플라스크 작성