Project Layout

Create a project directory and enter it:

```
$ mkdir flask-tutorial
$ cd flask-tutorial
```

Then follow the installation instructions to set up a Python virtual environment and install Flask for your project.

The tutorial will assume you're working from the `flask-tutorial` directory from now on. The file names at the top of each code block are relative to this directory.

A Flask application can be as simple as a single file.

`hello.py`

```
from flask import Flask

app = Flask(__name__)


@app.route('/')
def hello():
    return 'Hello, World!'
```

However, as a project gets bigger, it becomes overwhelming to keep all the code in one file. Python projects use *packages* to organize code into multiple modules that can be imported where needed, and the tutorial will do this as well.

The project directory will contain:

- `flaskr/`, a Python package containing your application code and files.
- `tests/`, a directory containing test modules.
- `venv/`, a Python virtual environment where Flask and other dependencies are installed.
- Installation files telling Python how to install your project.
- Version control config, such as git. You should make a habit of using some type of version control for all your projects, no matter the size.
- Any other project files you might add in the future.

By the end, your project layout will look like this:

```
/home/user/Projects/flask-tutorial
├── flaskr/
│   ├── __init__.py
│   ├── db.py
│   ├── schema.sql
│   ├── auth.py
│   ├── blog.py
│   ├── templates/
│   │   ├── base.html
│   │   ├── auth/
│   │   │   ├── login.html
│   │   │   └── register.html
│   │   └── blog/
│   │       ├── create.html
│   │       ├── index.html
│   │       └── update.html
│   └── static/
│       └── style.css
├── tests/
│   ├── conftest.py
│   ├── data.sql
│   ├── test_factory.py
│   ├── test_db.py
│   ├── test_auth.py
│   └── test_blog.py
├── venv/
├── setup.py
└── MANIFEST.in
```

If you're using version control, the following files that are generated while running your project should be ignored. There may be other files based on the editor you use. In general, ignore files that you didn't write. For example, with git:

`.gitignore`

```
venv/

*.pyc
__pycache__/

instance/

.pytest_cache/
.coverage
htmlcov/

dist/
```

```
build/
*.egg-info/
```

Continue to Application Setup.