Flask-Migrate

事前準備

Flask-Migrate をインストール（まだなら）：

```
pip install Flask-Migrate
```

app.pyに以下を追加

```
from flask_migrate import Migrate
migrate = Migrate(app, db)
```

プロジェクトのルートで以下を打つ（= ターミナル・コマンドプロンプト上）

```
flask db init
```

※これは「コンソール（ターミナル）」で打ちます。Python対話モードではありません。

カラム追加時の操作（毎回）

※下記2つも コンソール（ターミナル）で打つコマンド です。

モデルにカラムを追加したあと：

```
flask db migrate -m "add age column to user"
```

追加を実際のDBに反映

```
flask db upgrade
```

既存のレコードに追加したカラムに入る値の処理

SQLiteで ALTER TABLE するときに NOT NULL 制約をつけているのに DEFAULT が指定されていないと、
SQLite「既存レコードにNULLを入れるわけにはいかんよ！」→

```
OperationalError: Cannot add a NOT NULL column with default value NULL
```

該当のマイグレーションファイルを開きます

（例：migrations/versions/bdd19fa63016_add_column_puroduct_number_tb.py） op.add_column のところを以下のように編集：

```
with op.batch_alter_table('product_number') as batch_op:
    batch_op.add_column(sa.Column('serial_no', sa.String(length=20),
nullable=False, server_default=''))
```

◇ server_default='' を指定することで、SQLiteのエラーを回避できます。

保存したら再度実行

```
flask db upgrade
```

SQLiteにおける注意点

SQLiteではDDLがトランザクションに巻き込まれない SQLiteでは、以下のようなテーブル構造を変更するコマンド（DDL）がトランザクションとして一貫して扱われないという制限があります： ALTER TABLE CREATE TABLE DROP TABLE ADD COLUMN など そのため、Alembicが複数のDDL文を順に実行しようとしていた 最初のカラム追加だけは成功していた（確定済み） その後の別処理（たとえば外部キーエラーや重複追加）でエラーが出た でもすでに前半のDDL（ADD COLUMN）はロールバックされずに残っている という結果が起こりえます。