pip install flask

pip install flask\_sqlalchemy

pip install flask\_migrate

pip install psycopg2

**Init git repo and create .gitignore**

Okay, when our most important dependencies ready, we should initialize the git repository and specify the files that git should ignore.

Open the terminal and type:

git init

Next, create the .gitignore file in the project's dir, and add a few paths, that we should ignore:

\*\*/\_\_pycache\_\_/\*

migrations/\*

.vscode

**Create app.py and import deps**

It's the first step when we will really start coding with Python.

We need to create a file named "app.py", and import dependencies that we already installed.

import psycopg2

from flask import Flask

from flask\_sqlalchemy import SQLAlchemy

from flask\_migrate import Migrate

**Export flask variable**

When we finished creating the app.py file, we should export the path to our flask's app.

After that, the flask will know what file should run as an application.

Open the terminal and type:

export FLASK\_APP=app.py

**Define app and configure DB connection**

In the next step, we should define our flask application and create the app variable.

The second what we should do is to create code that will configure a connection with our database.

In the file named "app.py", we should add our app declaration and connection string below the imports.

app = Flask(\_\_name\_\_)

app.config['SQLALCHEMY\_TRACK\_MODIFICATIONS'] = False

app.config['SQLALCHEMY\_DATABASE\_URI'] = "postgresql://user:password@localhost:5432/dbname"

**Add app run**

In this step, we will focus on the app.run command.

That will let us start the application from the command line.

We will not use it in the first lesson, but we should add it to the next ones.

On the bottom of the app.py file, we should add app.run(debug=True) if the \_\_name\_\_ is equal to ‚\_\_main\_\_'.

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

**Create models directory**

Great! Now we can move forward and focus on the models for the database.

In the first lesson, we will create the only model named "prices", that will let us use the SQL table with the same name.

First of all, we need to create a directory named "models", add it to your project.

**Create prices.py in models directory and import DB from app**

Now, when we have created a necessary directory, we can go into the file.

First, create the file named prices.py (add it in the directory called "models").

Next, you should go into the prices.py file, and import variable named "db" from the app.

from app import db

**Create a model in prices.py**

This step is the biggest part of our today's coding, and the most important.

We need to create the whole model that we will use for the migrations.

We should start by creating a class named "PriceMode", and pass "db.Model", as a param.

The next step is to declare a variable named "\_\_tablename\_\_" with value "prices".

Below that line, we need to create columns that we will use inside our table.

After that, we should create a function named "\_\_init\_\_", and assign our columns to the self.

Let's take a look at how I did it in the example below.

class PriceModel(db.Model):

\_\_tablename\_\_ = 'prices'

id = db.Column(db.Integer, primary\_key=True)

company = db.Column(db.String())

date = db.Column(db.String())

openPrice = db.Column(db.Integer())

highPrice = db.Column(db.Integer())

lowPrice = db.Column(db.Integer())

closePrice = db.Column(db.Integer())

volume = db.Column(db.Integer())

def \_\_init\_\_(self, company, date, openPrice, highPrice, lowPrice, closePrice, volume):

self.company = company

self.date = date

self.openPrice = openPrice

self.highPrice = highPrice

self.lowPrice = lowPrice

self.closePrice = closePrice

self.volume = volume

def \_\_repr\_\_(self):

return f"<Price for {self.company} in day {self.date} is {self.closePrice}>"

**Import models in app.py and add migrate**

It's the last step of coding now, and we run the code!

We should come back to the file named "app.py" and import our PriceModel.

After importing the model, we should declare a variable named "migrate", and assing the Migrate function that we introduced before.

To the function named "Migrate", we need to pass the app and db variables as params.

This code should be after variable named "db", and before the last if-statement.

Let's take a look at the example below.

from models.prices import PriceModel

migrate = Migrate(app, db)

**Init DB**

We are done with the code!

Now we should run it and check if everything is fine.

The first what you need to do is initialize DB by using this method in the terminal:

flask DB init

**Migrate and upgrade**

If all went well, you should have some \_\_pycache\_\_ directories and new dir named "migrations".

If yes, you can go into the DB migrations.

The first of all what you need to do is to migrate, open the terminal, and type:

flask db migrate

After migration, you need to upgrade your with the results, open the terminal and type:

flask db upgrade