A screenshot of a computer

AI-generated content may be incorrect.

* To run the flask, we need Python. So first, let’s download the python executable file from python.org.
* Now, Need to install **PIP.**

1. pip(pip Install Packages) is the package installer for Python. You can use it to install packages from the Python Package Index and other indexes.
2. Download pip file and run this command on terminal. Python get-pip.py

* Let’s create virtual environment and check how to activate and deactivate this environment.

Python -m <environment Name> env

For example: python -m sunil env

– Here sunil is the virtual environment, created under your project,Now got your project >> .env >> script >>activate and press enter, once you entered environment will be started.

* To check, how many packages are installed in this environment, use below command.

Pip list

* To install new packages or library, use the command below.

Pip install <package name>

* Now, suppose you have to share this complete project setup along with all installed libraries and packages to someone else, in this case we will create one file requirement.txt using below command.

Pip freeze > requirements.txt

* How to deactivate this environment, use the command below.

Deactivate

* + Now Install flask using below command.

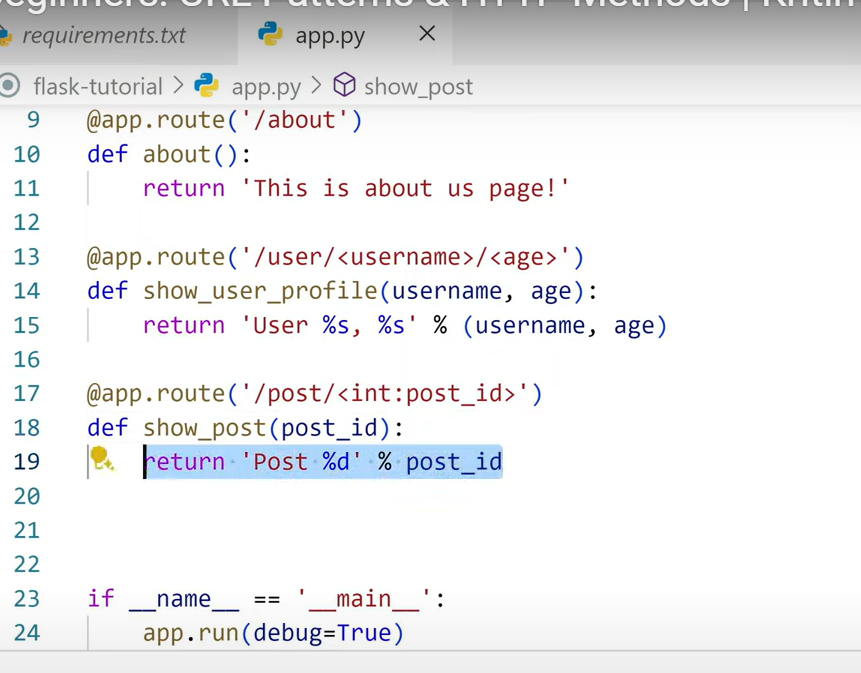
Pip install flask

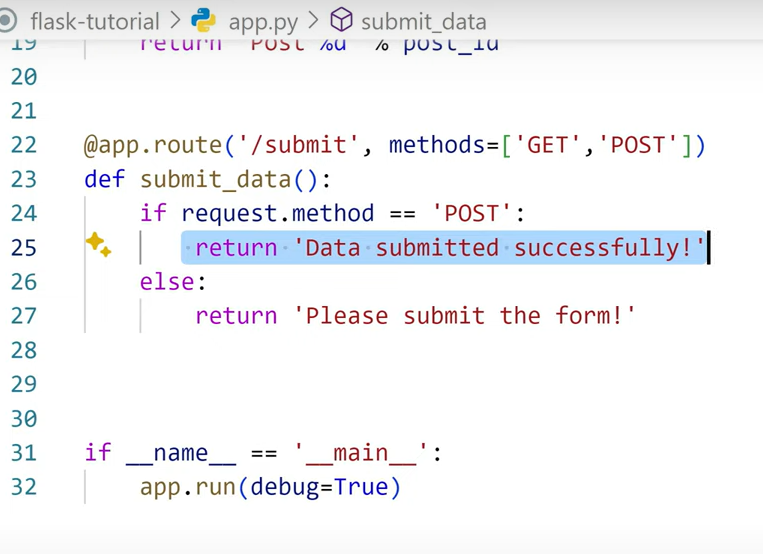
* Now, we will crate one file app.py under environment and execute Hello world.there are some steps we need to do in this file.
  + Import flask
  + Create one instance of flask
  + Create route
  + Write function and put the code whatever required functionality.
  + Check condition, if route is matched the run app.run()
  + Now, you can execute this file app.py from terminal, and then u will get localhost url and then u need to visit that url to get Flask output.

Python app.py

from flask import Flask  
app = Flask(\_\_name\_\_)  
  
@app.route('/')  
def hello\_world():  
 return 'Kaisa hai bhai, this hai kya?'  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run(debug = True)

* + **Routing Concept:**





* How to use Template in Flask: (Jina Template Engine)
  + We will create a templates folder under the environment, and inside template will create html file.
  + To call this HTML file in Fask, we will use **render\_template(‘filename.html’)**
  + But before using this render\_template, we need to import in app.py file.
* from flask import Flask, request, render\_template

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

@app.route('/about-us')  
def about\_us():  
 return render\_template('about\_us.html')

if \_\_name\_\_ == '\_\_main\_\_':  
 app.run(debug = True)

* How to pass or print List using loop value in template:
* @app.route('/list')  
  def hello\_world():  
   items = ['Python', 'Django', 'Flask', 'FastAPI']  
   return render\_template('python\_course.html', course\_name=items)

In Template file(python\_course.html)

<!-- For loop logic of jinja template -->  
{% for course\_name\_item in course\_name %}  
 <li>{{course\_name\_item}}</li>  
{% endfor %}

**How to add static files(CSS, JS, images) in your Flask App.**

* + Create static folder in ur app. And then create images, js and css folder under static folder.
  + Link, your images, js and css path into template file.
  + Use url\_for(‘static’, filename=<path of filename>”)

**Flask WTForms, Form Handling, Validation error handling:**

* + WTForms is a Python library that facilitates form creation and validation in web applications. Flask-WTF integrates WTForms with the Flask web framework, providing features like CSRF protection, and easy access to request data.
  + Install Flask-WTF and WTForms using pip:

**Flask with SQLAlchemy: Setting Up & Connecting to a Database:**

Flask-SQLAlchemy is an extension for Flask that adds support for SQLAlchemy to your application. It simplifies using SQLAlchemy with Flask by setting up common objects and patterns for using those objects, such as a session tied to each web request, models, and engines.

**Installation:** pip install -U Flask-SQLAlchemy

**To create a database and table using SQLAlchemy in Flask, below are the Key Concepts:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Explanation step by step:**

**1. Set up your Flask project**

Make sure you have Flask and SQLAlchemy installed.

pip install Flask

pip install flask\_sqlalchemy

**2. Create your Flask app**: Create a Python file (e.g., app.py) and initialize your Flask app with SQLAlchemy:

from flask import Flask

from flask\_sqlalchemy import SQLAlchemy

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

# Step 1: Configure the SQLAlchemy database URI

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///mydatabase.db' # or use MySQL/PostgreSQL URI

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

# Step 2: Initialize SQLAlchemy

db = SQLAlchemy(app)

**3. Define your database model (table)**

Each class you define inherits from db.Model and represents a table.

# Step 3: Create a model class

class User(db.Model):

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

name = db.Column(db.String(100), nullable=False)

email = db.Column(db.String(120), unique=True, nullable=False)

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

return f'<User {self.name}>'

**4. Create the database and tables**

You now need to create the actual database file and tables. This is done using db.create\_all(). Add this block at the end of app.py:

# Step 4: Create the tables

with app.app\_context():

db.create\_all()

run python app.py