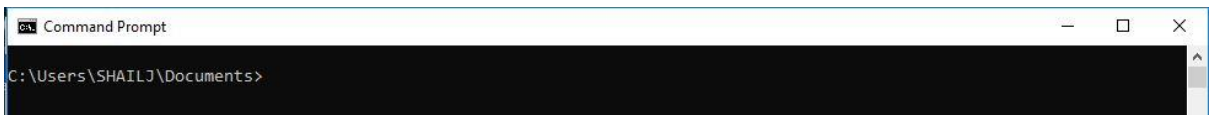
BASIC FLASK APP DEMO – Creating a data collection form

Pre-requirements – Python 3 installed (able to run through the command prompt and use the pip command), VSCode installed.

For this tutorial I will be developing the code in VSCode and running in a virtual environment, I found this to be the easiest IDE when working with Flask as the full project requires Python, HTML and CSS files.

Creating the folder and installing the packages

1. Open Command Prompt
2. C:
3. cd (copy route to where you want to create new folder from file explorer) **OR** if you know the path type it as in my example : cd Users\{USERNAME}\Documents



4. command prompt should now look like mine above
5. mkdir Flask_App_Demo
6. cd Flask_App_Demo
7. https://github.com/J1-Shail/Flask_App_Demo/blob/master/requirements.txt save this text file to this folder as requirements.txt (This is to install all packages required for the App)
8. pip install virtualenv
9. py -3 -m venv Venv *(Venv is the name of this virtual environment being created, you could call it what you want)*
10. Venv\scripts\activate.bat
11. pip install -r requirements.txt

Open VSCode and open the Flask_App_Demo folder

1. **Click new file and create a file called app.py** this is where we will put all the python code for the application. On bigger projects it is better to split the code up. For more in depth Tutorials you can work through either of these links:
    - https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world
    - https://www.youtube.com/playlist?list=PL-osiE80TeTs4UjLw5MM6OjgkjFeUxCYH
      (Corey Schafer on Youtube)
    - https://flask.palletsprojects.com/en/1.1.x/quickstart/

2. In VSCode Explorer on the left side of the screen hover over Flask_App_Demo and **create a new folder called templates**. Within this templates folder **create the files output.html, home.html and layout.html** and copy code from my github (https://github.com/J1-Shail/Flask_App_Demo).

3. Now **create a folder called static** and within this folder **create the file main.css**, copy my code from github.

4. The rest of the code is now in the app.py file **DO NOT COPY AND PASTE THIS FROM MY GITHUB, WE WILL WORK THROUGH THIS TOGETHER**. At the top of the file we import all required packages.

```
from flask import Flask, url_for, render_template, redirect
from flask_sqlalchemy import SQLAlchemy
from flask_wtf import FlaskForm
from wtforms import SelectField, StringField, SubmitField
from wtforms.fields.html5 import DateField
from wtforms.validators import DataRequired, Email
from flask_bootstrap import Bootstrap
```

5. Create an instance of the Flask class

```
app = Flask(__name__)
```

6. Set the secret key, this can be anything and is required to use a lot of features in flask. To generate a secret key for future projects you could use the secrets module.

```
app.secret_key = 'SecretKey'
```

7. Set SQLAlchemy URI for connection to the database, this determines what the database will be called when created and then create the database instance

```
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///info.db'
db = SQLAlchemy(app)
```

8. Create the bootstrap instance, this is used in the home.html file to use quick_form for rendering the form.

```
bootstrap = Bootstrap(app)
```

9. Generate the database model

```
class Info(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(30), nullable=False)
    d_o_b = db.Column(db.Datetime(), index=True, nullable=False)
    email = db.Column(db.String(120), nullable=False)
```

```
    city = db.Column(db.String(),  nullable=False)

    def __repr__(self):
        return f"Info('{self.name}', '{self.d_o_b}', '{self.email}',
'{self.city}')"
```

10. A list of tuples for the options in the City Select field for the form, 1st value is what is saved to the database, 2nd value is what is displayed in the form

```
Cities = [
    ('CAM', 'Cambridge'),
    ('LDN', 'London')
    ]
```

11. Create the flask form fields

```
class DataForm(FlaskForm):

    name_input = StringField(label='Name', validators=[DataRequired()])

    date_of_birth = DateField('Date of Birth', format='%Y-%m-%d',
validators=[DataRequired()])

    email_address = StringField(label='Email Address',
validators=[DataRequired(), Email()])

    city_input = SelectField('City', choices=cities)

    submit = SubmitField('Submit')
```

12. Use the route() decorator for the URL address that should trigger the function.
redirect(url_for()) will run the app route stated in url_for()
render_template() will render the html template that is entered, you can also enter variables into the render_template() command and these can then be called within the html template with Jinja2.

```
@app.route('/', methods=['GET', 'POST'])
@app.route('/home', methods=['GET', 'POST'])
def home():
    form = DataForm()
    if form.validate_on_submit():
        data = Info(name=form.name_input.data,
        d_o_b=form.date_of_birth.data, email=form.email_address.data,
        city=form.city_input.data)
        db.session.add(data)
        db.session.commit()
```

```
        return redirect(url_for('output'))
    return render_template('home.html', form=form)


@app.route('/output')

def output():

    details = Info.query.order_by(Info.id.desc()).first()


    name_details = details.name

    dob_details = details.d_o_b

    email_details = details.email

    city_details = details.city


    return render_template('output.html', name=name_details, dob=dob_details,
    email=email_details, city=city_details)
```

13. Finally, you need to call the app to run, debug=True is used when in development mode for debugging

```
if __name__ == "__main__":
        db.create_all()
        app.run(debug=True)
```

14. Running the code

- Open Command Prompt
- Navigate to the directory where the project is saved
- Activate Virtual Environment Venv\scripts\activate.bat
- Type the following: python app.py
- Open chrome browser and go to 127.0.0.1:5000 (command prompt will tell you where the app is running)
- You should now see the data form