<https://medium.com/free-code-camp/how-to-build-a-web-app-using-pythons-flask-and-google-app-engine-52b1bb82b221>

How to build a web app using Python’s Flask

## A small tutorial project for learning Flask, APIs, and Google App Engine for beginners

If you want to build web apps in a very short amount of time using Python, then [Flask](http://flask.pocoo.org/)is a fantastic option.

Flask is a small and powerful web framework (also known as “[microframework](https://en.wikipedia.org/wiki/Microframework" \t "_blank)”)

In this tutorial, we will see how to build a simple **weather app** with some dynamic content using an **API**

To create a weather app, we will need to request an API key from [Open Weather Map](https://openweathermap.org/api) (<https://openweathermap.org/api>)

The free version allows up to 60 calls per minute, which is more than enough for this app

This tutorial will also cover: (1) basic CSS design, (2) basic HTML with Jinja

The steps we’ll take are listed below:

* **Step 0:** Installing Flask (this tutorial doesn’t cover Python and PIP installation)
* **Step 1:**Building the App structure
* **Step 2:** Creating the Main App code with the API request
* **Step 3:**Creating the 2 pages for the App (Main and Result) with [Jinja](http://jinja.pocoo.org/" \t "_blank), HTML, and CSS
* **Step 4:** Deploying and testing on your local laptop

## ****Step 0 — Installing Flask and the libraries we will use in a virtual environment.****

We’ll build this project using a virtual environment. But why do we need one?

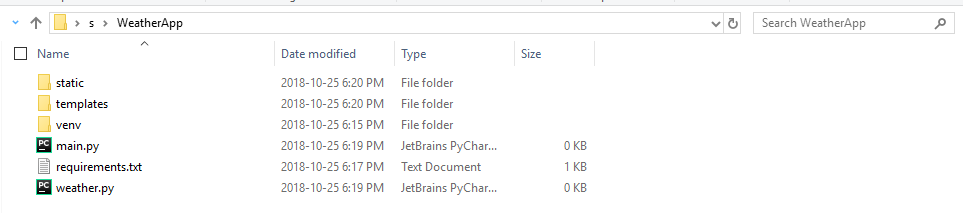
With virtual environments, you create a local environment specific for each projects. You can choose libraries you want to use without impacting your laptop environment. As you code more projects on your laptop, each project will need different libraries. With a different virtual environment for each project, you won’t have conflicts between your system and your projects or between projects.

* Run Command Prompt (cmd.exe) with administrator privileges. Not using admin privileges will prevent you from using pip.
* (Optional) Install virtualenv and virtualenvwrapper-win with PIP. If you already have these system libraries, please jump to the next step.
* Create your folder with the name “WeatherApp” and make a virtual environment with the name “venv” (it can take a bit of time)
* Activate your virtual environment with “call” on Windows (same as “source” for Linux). This step changes your environment from the system to the project local environment.
* Create a requirements.txt file that includes Flask and the other libraries we will need in your WeatherApp folder, then save the file. The requirements file is a great tool to also keep track of the libraries you are using in your project.
* Install the requirements and their dependencies. You are now ready to build your WeatherApp. This is the final step to create your local environment.

## ****Step 1 — Building the App structure****

You have taken care of the local environment. You can now focus on developing your application. This step is to make sure the proper folder and file structure is in place. The next step will take care of the backend code.

* Create two Python files (main.py, weather.py) and two folders (static with a subfolder img, templates).



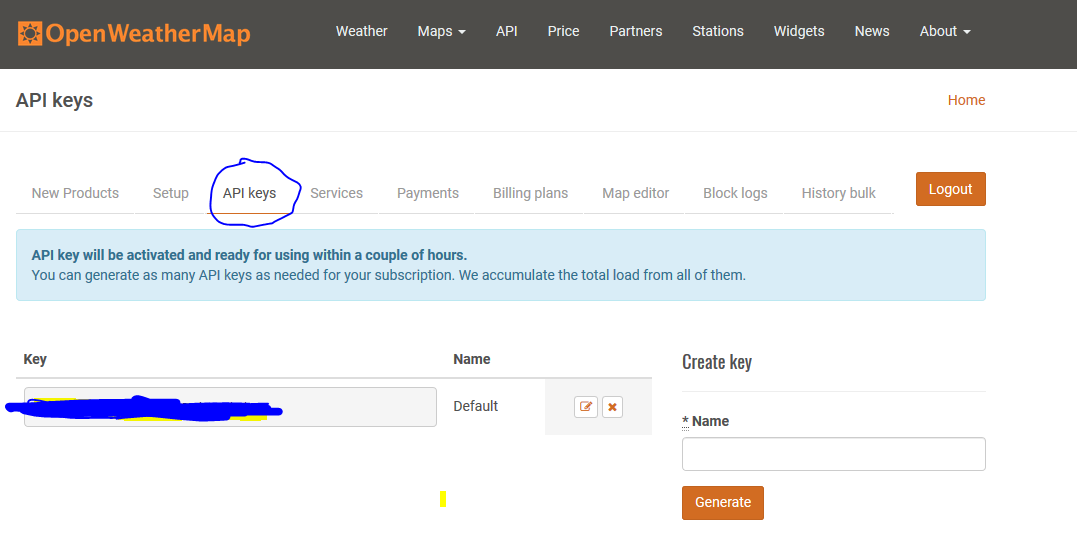
## ****Step 2 — Creating the Main App code with the API request (Backend)****

With the structure set up, you can start coding the backend of your application. Flask’s “Hello world” example only uses one Python file.

This tutorial uses two files to get you comfortable with importing functions to your main app.

1. The main.py is the server that routes the user to the homepage and to the result page.
2. The weather.py file creates a function with API that retrieves the weather data based on the city selected. The function populates the resulting page.

* Edit main.py with the following code and save
* Request a free API key on Open Weather Map



* Edit weather.py with the following code (updating the API\_KEY) and save

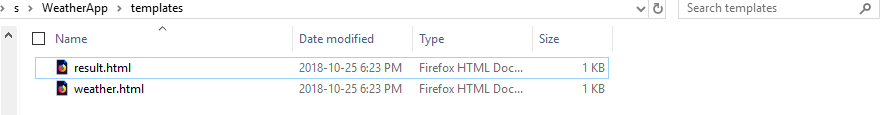
## ****Step 3 — Creating pages with [Jinja](http://jinja.pocoo.org/" \t "_blank), HTML, and CSS (Frontend)****

This step is about creating what the user will see.

The HTML pages weather and result are the one the backend main.py will route to and give the visual structure. The CSS file will bring the final touch. There is no Javascript in this tutorial (the front end is pure HTML and CSS).

It was my first time using the [Jinja2](http://jinja.pocoo.org/)template library to populate the HTML file. It surprised me how easy it was to bring dynamic images or use functions (e.g. rounding weather). Definitely a fantastic template engine.

* Create the first HTML file in the templates folder (weather.html)
* Create the second HTML file in the templates folder (result.html)



Add a CSS file in the static folder (style.css)

* Download the images in the img subfolder in static

Link with the images on [Github](https://github.com/tristanga/WeatherApp_Image" \t "_blank):

## ****Step 4 — Deploying and testing locally****

At this stage, you have set up the environment, the structure, the backend, and the frontend. The only thing left is to launch your app and to enjoy it on your localhost.

* Just launch the main.py with Python

python main.py

* Go to the localhost link proposed on cmd with your Web Browser (Chrome, Mozilla, etc.). You should see your new weather app live on your local laptop :)

