**Flask creating**

**Python Version**

We recommend using the latest version of Python. Flask supports Python 3.7 and newer.

## Virtual environments

Use a virtual environment to manage the dependencies for your project, both in development and in production.

What problem does a virtual environment solve? The more Python projects you have, the more likely it is that you need to work with different versions of Python libraries, or even Python itself. Newer versions of libraries for one project can break compatibility in another project.

Virtual environments are independent groups of Python libraries, one for each project. Packages installed for one project will not affect other projects or the operating system’s packages.

Python comes bundled with the **[venv](https://docs.python.org/3/library/venv.html" \l "module-venv" \o "(in Python v3.11))** module to create virtual environments.

### **Create an environment**

Create a project folder and a venv folder within:

> mkdir myproject

> cd myproject

> py -3 -m venv venv

### **Activate the environment**

Before you work on your project, activate the corresponding environment:

> venv\Scripts\activate

Your shell prompt will change to show the name of the activated environment.

## Install Flask

Within the activated environment, use the following command to install Flask:

$ pip install Flask

## A Minimal Application

A minimal Flask application looks something like this:

**from** flask **import** Flask

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

@app.route**(**"/"**)**

**def** hello\_world**():**

**return** "<p>Hello, World!</p>"

So what did that code do?

1. First we imported the [**Flask**](https://flask.palletsprojects.com/en/2.2.x/api/#flask.Flask) class. An instance of this class will be our WSGI application.
2. Next we create an instance of this class. The first argument is the name of the application’s module or package. \_\_name\_\_ is a convenient shortcut for this that is appropriate for most cases. This is needed so that Flask knows where to look for resources such as templates and static files.
3. We then use the [**route()**](https://flask.palletsprojects.com/en/2.2.x/api/#flask.Flask.route) decorator to tell Flask what URL should trigger our function.
4. The function returns the message we want to display in the user’s browser. The default content type is HTML, so HTML in the string will be rendered by the browser.

Save it as hello.py or something similar. Make sure to not call your application flask.py because this would conflict with Flask itself.

To run the application, use the flask command or python -m flask. You need to tell the Flask where your application is with the --app option.

$ flask --app hello run

\* Serving Flask app 'hello'

\* Running on http://127.0.0.1:5000 (Press CTRL+C to quit)

* APP deploy in Azure Webapp

If you want to deploy your webapp in azure portal than login azure-cli and us command

# az webapp up --name zeef --resource-group webf --plan webapzeeflaskcheck --sku F1 --location eastus --runtime "python:3.11"

Git Push or Clone

If create repository in your GitHub account and use your https GitHub link and clone new folder and your document past on their add commit and push your repository following commands

* git clone <https://github.com/Mdzeeshan10/Web_app_in_Flask_Python.git> (your https github url)
* git status (status check)
* git add . (you provide . means all files folder in in this folder add)
* git commit -m “My first commit” (commit code)
* git push origine master (after this command it says your github login credential provide then your code is on github)

if your already folder then you want to your folder in git hub then follow below commands

* git init (it is create your .git file in you folder)
* git status (check status)
* git add . (all file adds in your git)
* git commit -m “first commit” (commits your code as you want)
* git remote add origine <https://github.com/Mdzeeshan10/Web_app.git> (your https github url)
* git push -u origine master (your all folder puch in your github)

# Large File Transfer to git hub in local to github

* git init
* git remote add origin
* git lfs install
* git lfs track \*.h5  \*.pkl
* git status(to confirm status)
* cat .gitattributes(to confirm that file name is coming in gitattributes)
* git add .gitattributes \*.h5 \*.pkl
* git commit -m "message"
* git push origin branch\_name
* git lfs push origin branch\_name

To create new branch:

git branch branch\_name

to go to the new branch:

git checkout branch\_name