

# Python at Reclaim Hosting

Before starting this tutorial, make sure your Flask app is fully functioning and complete, with all files in **one project folder** (it may contain subfolders) on your own computer.

Make the following **change** in the `.py` file that runs your Flask app:

```
routes.py
1  from flask import Flask, render_template
2  app = Flask(__name__)
3
4  # web host requires application in passenger_wsgi
5  application = app
6
```

Below the line that says:

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

**Add** this line:

```
application = app
```

**Save the .py file.** Why you must do this is explained on the last page of this document.

## requirements.txt

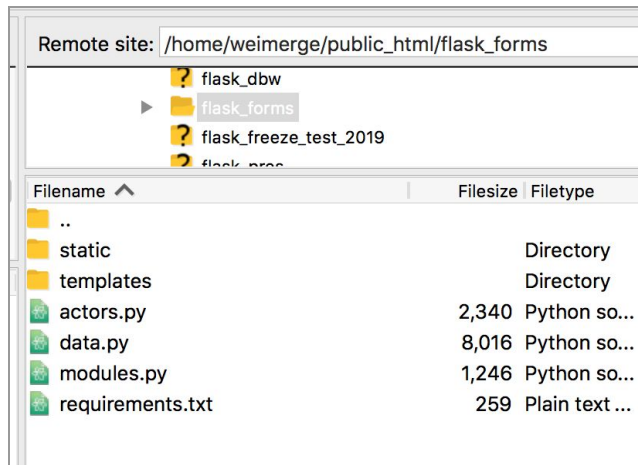
Before you upload your folder to your server, create a *requirements.txt* file. This must be done in Terminal, in the folder where your *env/* folder is. After you have this file, **copy it** into your project folder — the folder you will upload to the server. To create this file, at the command prompt:

```
pip freeze > requirements.txt
```

Optional reading: [Why and how to make a requirements.txt](#)

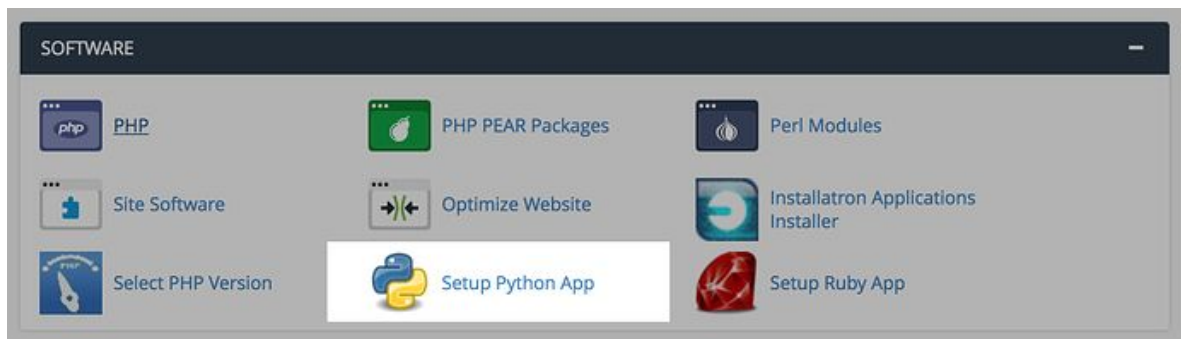
## Put your project on your server

Use your normal method to *copy* or upload **that project folder** into the *public\_html* directory on the server at Reclaim. You may rename the project folder if you like. **Make sure the folder name has NO SPACES and NO PUNCTUATION (other than an underscore) and NO UPPERCASE LETTERS.** (My folder below is named *flask\_forms*.)



## Set up the Python app on the server

1. Log in at **Reclaim Hosting** (assuming that is where your domain is hosted).
2. Open **cPanel**.
3. Click this ("Setup Python App"):



4. Click this:



5. Fill in the following:
  - a. Python version: Select the highest number
  - b. Application root: *public\_html/your\_folder\_name/*
  - c. Application URL: *your\_folder\_name/*
  - d. Application startup file: The name of your Flask app file (the .py file that runs it)
  - e. Application Entry point: *application*
  - f. Passenger log file: Fill as shown below

WEB APPLICATIONS
+ CREATE APPLICATION
CANCEL
CREATE

Python version
3.7.3

Application root
public\_html/flaskform/

It is a physical address to your application on a server that corresponds with its URI. Upload your application files here.

Application URL
weimergeeks.com flaskform/

It is an HTTP/HTTPS link to your application

Application startup file
actors.py

Application Entry point
application

Setup wsgi callable object for your application

Passenger log file
/home/weimerge/logs/passenger.log

You can define the path along with the filename (e.g. /home/weimerge/logs/passenger.log)

? Support

- Click the **CREATE** button (upper right).
- You should get *a confirmation popup* that says your app is configured.  
**If not**, don't panic. Go to the list of your **Web Applications** and click the EDIT icon for your app there:

 Python

WEB APPLICATIONS
+ CREA

App URI	App Root Directory	Status	Actions
weimergeeks.com/flask_db_write	/home/weimerge/public_html/flask_db_write	● started (v3.7.3)	<div>Edit the application</div> <div> <div></div> <div>↺</div> <div>✎</div> <div>🗑</div> </div>

- Stop the app:



- Farther down on the page:

Configuration files

▶ Run Pip Install

+ Add

Execute python script

You can also enter a command to execute (e.g. `/path/to/manage.py migrate`). Note, only python

▶ Run Script

You need to type *requirements.txt* and then click **ADD**. Note: **Make sure you followed the instructions above to create a *requirements.txt* file. Make sure it is in the app's folder.**

10. Click **Run Pip Install** and select *requirements.txt* when it appears.
11. *Wait for a while*. You should get a confirmation popup that says it went fine.
12. Start the app:



13. You should now be able to see your app running at <http://yourdomain.com/foldername/> or <https://yourdomain.com/foldername/>
14. If you get a weird little message ("It works!") *instead of your app's start page*, do the following:
  - a. Replace the Flask app file (the .py file that runs it) on the server with the one on your computer. Just drag and drop, replace.
  - b. **Restart** the application.



Did you mess up? You can instantly *delete* an app with the **DESTROY** button at top right.  
**Warning:** It does NOT ask if you are sure! **Good news:** It will not affect your uploaded files in the folder. It just removes the added bits from Reclaim.

If your app is not working yet, try clicking the **RESTART** button shown above.

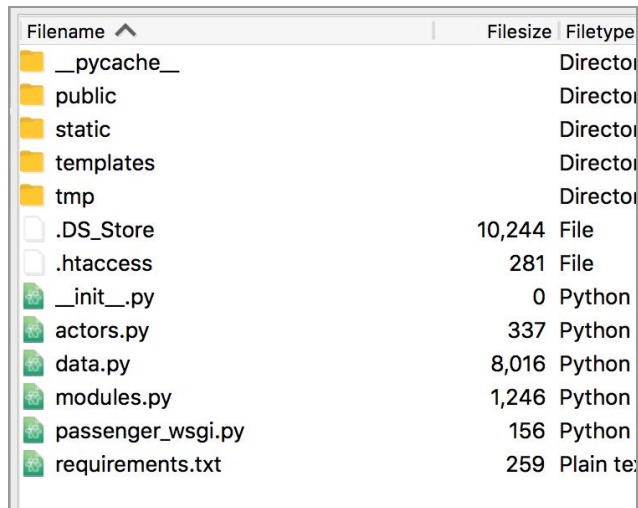
## Other helpful facts

If you replace/update files in your app, you might need to **restart** it. In **cPanel**, open "Setup Python App" to get to the access panel for your app.

You can create *more than one app* using “Setup Python App.” Reclaim does not appear to limit how many Python apps we can have.

## Directory structure

Here’s my directory on Reclaim, viewed in FileZilla, after the above steps:



Filename	Filesize	Filetype
__pycache__		Directory
public		Directory
static		Directory
templates		Directory
tmp		Directory
.DS_Store	10,244	File
.htaccess	281	File
__init__.py	0	Python
actors.py	337	Python
data.py	8,016	Python
modules.py	1,246	Python
passenger_wsgi.py	156	Python
requirements.txt	259	Plain text

These files and folders were **added by** the “Setup Python App” process:

- public/
- tmp/
- .htaccess
- passenger\_wsgi.py

**DO NOT delete or modify** `.htaccess` or `passenger_wsgi.py`!

## Not suitable for commercial projects

As [this fine tutorial](#) points out, “There are better ways to deploy your Flask application.” But as the author also notes, “However, great tools come with a price tag, and you might be already paying for a **shared hosting account**. So let’s use that yo.” We have a shared hosting account at Reclaim, so that’s what we’re using. Yo.

## *Don’t* include your virtual environment folder

If you have a `virtualenv` or `env` folder inside your Flask app project, **you should NOT upload that to your server**. During the setup process, a **new** virtual environment is created for you *on your server*.

## Some things that worked fine on your computer ...

... May not work at all on the live server. No doubt there are lots of different reasons for this, and if you run into a problem, you will just have to Google the heck out of it until you solve it.

**An important tip:** If you start changing the code, test it on your own computer first before you put it on the server. Running in debugging mode can give you some valuable clues.

## Why we add a line to the `.py` file

Near the top of this document, I told you to add this line to the `.py` file that runs your Flask app:  
`application = app`

I learned that from [this fine tutorial](#). Without it, **older** versions of WSGI couldn't run your app. ([What is WSGI?](#))

In the file `passenger_wsgi.py`, there is code that references `application`. You should not touch the code in `passenger_wsgi.py` — all you need to do is **add that one line** in your Python Flask script, *after* the line that says:

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

If you forget to do this, you might see this in the browser:

### Web application could not be started

An error occurred while starting the web application. It exited before signalling successful start about this problem.

#### Raw process output:

```
Traceback (most recent call last):
  File "/opt/passenger/helper-scripts/wsgi-loader.py", line 320, in <module>
    app_module = load_app()
  File "/opt/passenger/helper-scripts/wsgi-loader.py", line 61, in load_app
    return imp.load_source('passenger_wsgi', startup_file)
  File "passenger_wsgi.py", line 9, in <module>
    application = wsgi.application
AttributeError: 'module' object has no attribute 'application'
```

*Updated March 22, 2020 — completely new interface for cPanel Python setup; all screenshots replaced; all instructions rewritten.*