

Instructions for using Python Anywhere to host a dashboard

Python Anywhere account setup

1. Go to <https://www.pythonanywhere.com/pricing/>, create a Beginner Account (this is the free account, but there are several paid tiers of use above this one if you want)
2. Once registered, you'll be directed through a tour of the website. Then you will be taken to the Python Anywhere user's dashboard.
3. Click on "Open Web Tab" under "Web Apps"
4. Click on "Add a new web app"
5. You will see a message that your custom web app will have to be *username.pythonanywhere.com* where username is your Python Anywhere user name. You don't get a choice on the free tier.
6. Click "Next"
7. Choose "Flask"
8. Choose the latest version of Python available
9. Next you'll be asked to Quickstart a New Flask app. Leave the default Path alone and click "Next"
10. Your website is now created, (see for yourself at *username.pythonanywhere.com*), but it is populated by the placeholder "Hello from Flask!" and nothing more. I recommend bookmarking this "Configuration for *username.pythonanywhere.com*" page, as it has several features we will want to use
 - a. The reload button, once you make changes to the app
 - b. The button to renew the website (it must be clicked once every 3 months to prevent the website from being deleted)
 - c. Some basic usage tracking
 - d. Logs, especially the error log, for debugging
 - e. Buttons to disable or delete the app

Server configuration

11. Under code, click on the WSGI configuration file. It will bring you to a text editor for a Python file:
 - a. Comment out the line that reads:

```
from flask_app import app as application
```
 - b. Add two new lines at the bottom:

```
from flask_app import app
application = app.server
```

Installing packages

12. Return to the configuration page. At the top, click “Consoles”. Click “Bash” under start a new console, and Other. This will call up a command line
13. On this command line, use pip install to install all the packages you need. For starters, try: pip install python-dotenv
 - a. Some packages are pre-installed, like numpy and pandas
 - b. If your app isn’t working, you can look at the error log. If you see a “Module Not Found” error, go back to this console and reinstall.

Adding and editing files

14. Return to the configuration page. At the top, click “Files”
15. The file system on Python Anywhere gives you a root directory /home/username. This is the default location where your Python environment will look for loading and saving files. There is also a subdirectory /mysite that contains the code for your Dash app
16. In the root directory /home/username, click Upload a file. If you have a .env file associated with your app, upload the .env file here. (To see the .env file in a file select window, display hidden files. On a Mac, this is Shift + Command + . and on Windows this is CTRL + H)
17. If you have any other files that are needed, such as a .csv data file, upload those files to the /home/username folder too. Note however you only have 512 MB in storage on the free tier of Python Anywhere
18. Click on the mysite folder (on the left). Then click on flask_app.py. This will open a text code editor. Replace the text with the code in your Dash app.py file.
19. If you have additional .py files for your work that you import to the flask_app.py file, upload them also to this /mysite folder

Running the app

20. Return to the configuration page. If the packages have been installed and files updated, you are ready to launch the app. Click “Reload username.pythonanywhere.com”.
21. Wait about 30 seconds or so, then visit your website. If everything is working, you’ll see your app, working just as it would locally. If instead you see a message that there is an error, proceed to the next section.
22. If you made any changes to the packages, files, or python code, you will need to click the Reload button again before seeing the changes.

Debugging errors


23. Return to the configuration page. Under log files, click error log.
24. The error log is a giant text file that is appended each time the dashboard is reloaded. Pay attention to the time stamp on each line – it is easy to get confused as the top of the file will always show you older errors. Scroll down to the bottom to see the most recent errors.

25. Look for the succinct error messages that begin “TypeError” or “ImportError”, for example. These are the same errors that would appear in your local Python development environment. Please use these to identify why your dashboard is not displaying.

Longterm maintenance

26. Return to the configuration page. At least once every three months, you will need to click the “Run until 3 months from today” button or the website will be deleted

Optional but cool: Including images in the dashboard

27. Images can add a lot aesthetically to a dashboard and to the user experience of it. But links to images often break, even when the image file is stored locally, leading to a broken image icon appearing instead: 
28. The best way to include images requires you to first create a new subfolder inside /home/username/mysite called “assets”. Click files, then the mysite folder, then type “assets” in the box that reads “Enter new directory name”, and click “New directory”
29. Click the assets folder, and upload your image files here. JPG and PNG formats for images work best.
30. In your dash code, to include an image (named myimage.png, for example), include the following code in your layout:

```
html.Img(src=app.get_asset_url('myimage.png'), style =
{'width': '20%', 'textAlign': 'center'})
```

where ‘width’ allows you to control the size of the image.

Optional but cool: changing the browser tab icon and title

31. When you open a webpage in a browser tab, a small picture and a title appear on the tab. By default, with dash, the dash icon and the title “Dash” are used. This icon is called a “favicon”
32. Choose an image that you want for your favicon, and if it isn’t square, crop it so that it is roughly square.
33. Go to a website like <https://favicon.io/> to convert this image to a file named favicon.ico
34. Upload the favicon.ico file to your assets folder (see step 28)
35. In your dash code, right after defining the app variable, set the title attribute to be whatever you want to appear in the tab. For example, to set the title as “mydashboard”, your code would include lines like this:

```
app = dash.Dash(__name__,
    external_stylesheets = external_stylesheets)
app.title = 'mydashboard'
```