School of Games and Creative Technology
BSc (Hons) Computer Science

University for the Creative Arts

FGCT6021 MOBILE APPLICATION DEVELOPMENT
LAB 6 – TOPIC 6 FRONTEND & BACKEND

Submission

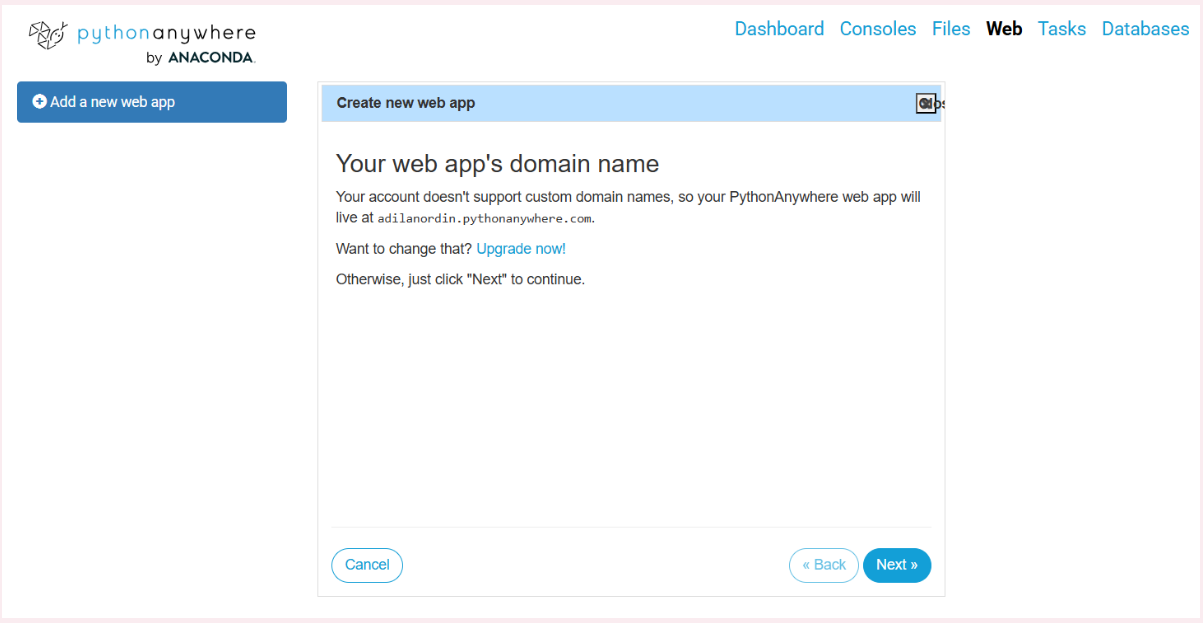
1. Submit your Learning Journal link (GitHub), your PythonAnywhere link and weekly journal entry on myUCA.
2. During the workshop, you will show your progress and share what you have learned.

Task 1 Set up Flask in PythonAnywhere

If you have not registered on PythonAnywhere, create an account first.

Part 1: New Web App

1. Log in to **PythonAnywhere** and click **Web**.
2. Click **Add a new web app**.
3. At **Create new web app** click **Next**.


pythonanywhere by ANACONDA
Dashboard Consoles Files Web Tasks Databases
Add a new web app
Create new web app
Cancel
« Back
Next »

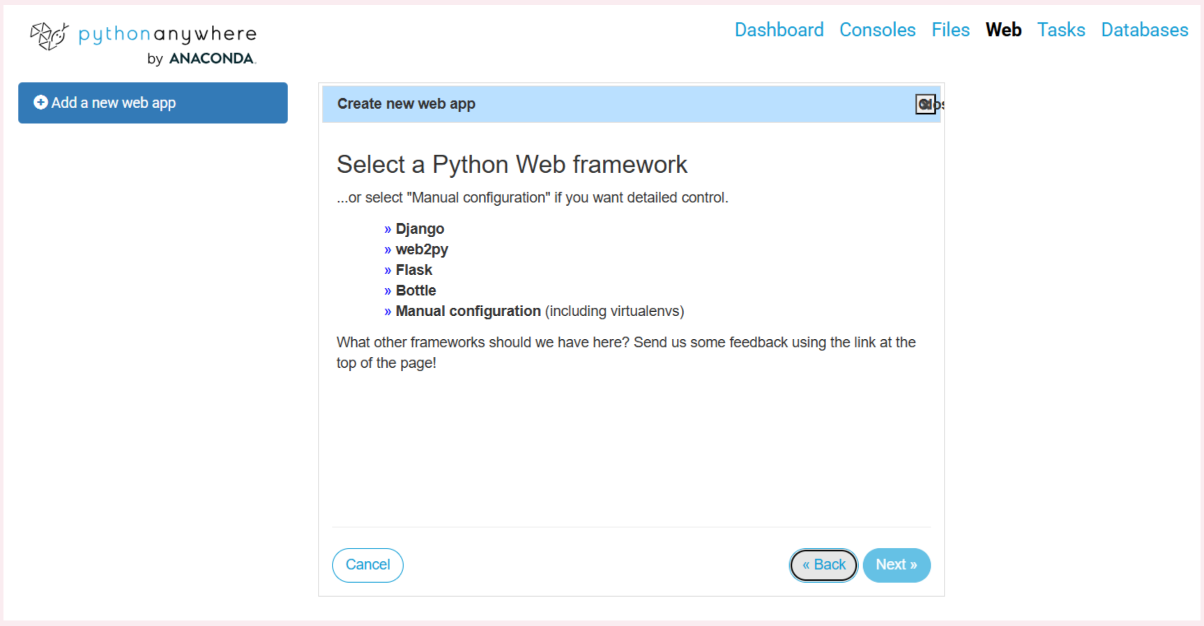
Your web app's domain name

Your account doesn't support custom domain names, so your PythonAnywhere web app will live at adilanordin.pythonanywhere.com.

Want to change that? Upgrade now!

Otherwise, just click "Next" to continue.

4. **Select a Python Web framework**, choose **Flask** and click **Next**.


pythonanywhere by ANACONDA
Dashboard Consoles Files Web Tasks Databases
Add a new web app
Create new web app

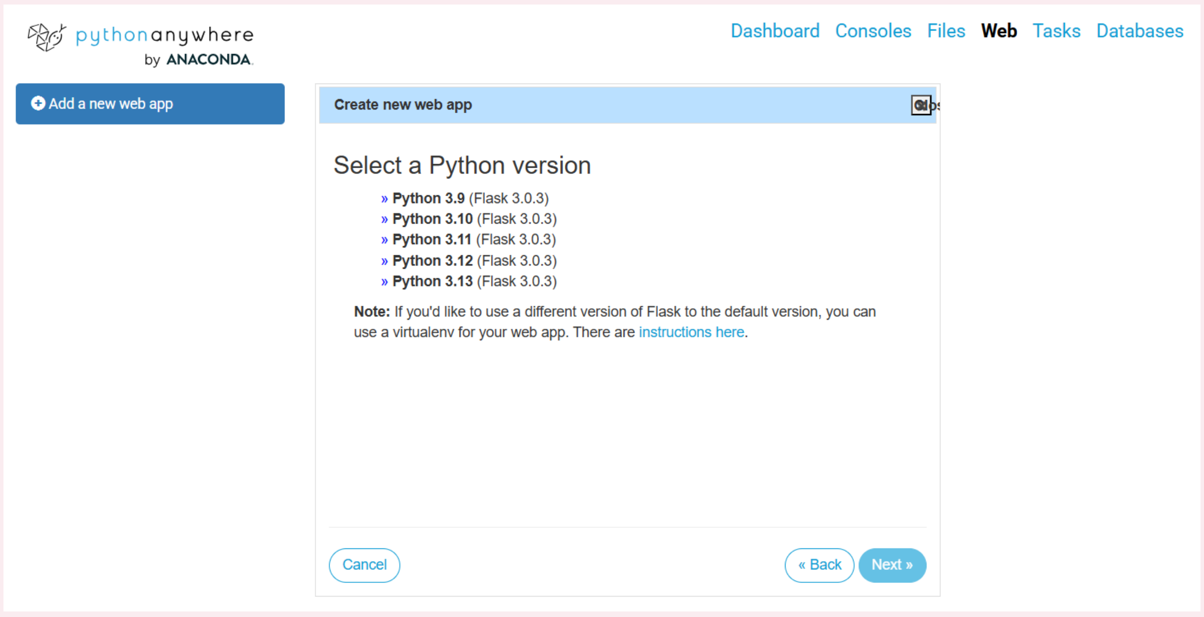
Select a Python Web framework

...or select "Manual configuration" if you want detailed control.

- » **Django**
- » **web2py**
- » **Flask**
- » **Bottle**
- » **Manual configuration** (including virtualenvs)

What other frameworks should we have here? Send us some feedback using the link at the top of the page!

5. **Select a Python version**, choose the latest **Python 3.13** and click **Next**.

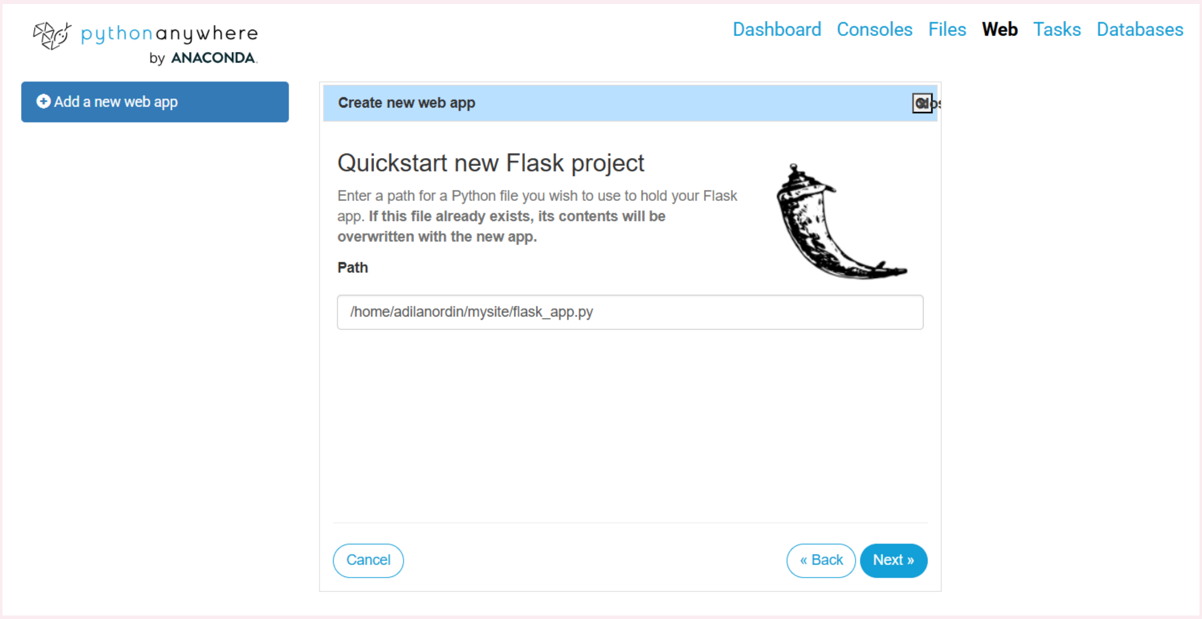

pythonanywhere by ANACONDA
Dashboard Consoles Files Web Tasks Databases
Add a new web app
Create new web app

Select a Python version

- » **Python 3.9** (Flask 3.0.3)
- » **Python 3.10** (Flask 3.0.3)
- » **Python 3.11** (Flask 3.0.3)
- » **Python 3.12** (Flask 3.0.3)
- » **Python 3.13** (Flask 3.0.3)

Note: If you'd like to use a different version of Flask to the default version, you can use a virtualenv for your web app. There are instructions here.

6. At **Quickstart new Flask project**, click **Next**.


pythonanywhere by ANACONDA
Dashboard Consoles Files Web Tasks Databases
Add a new web app
Create new web app

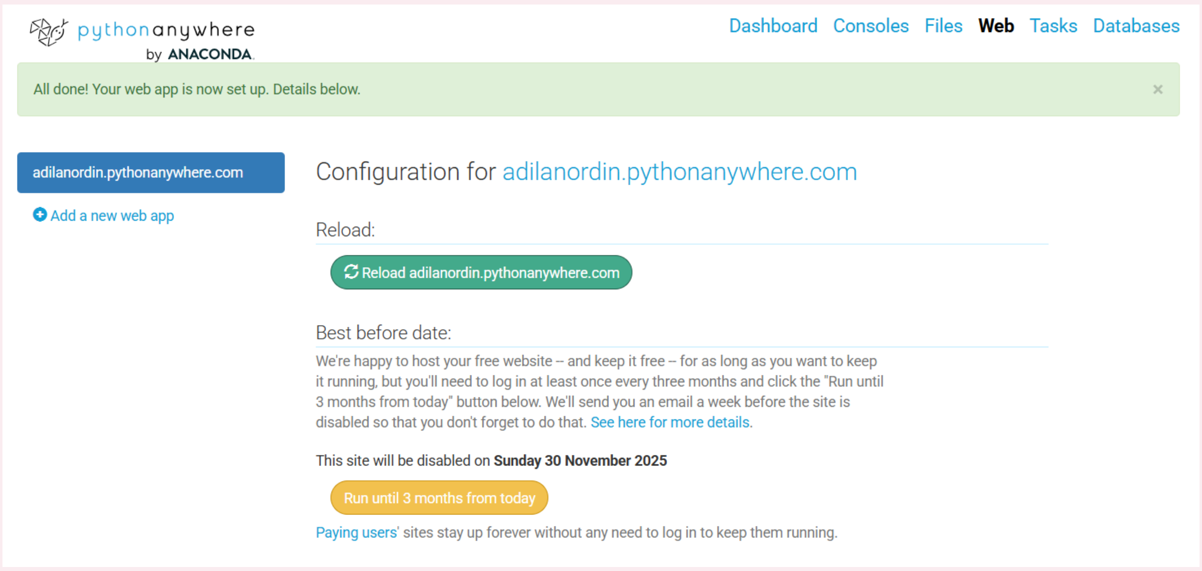
Quickstart new Flask project

Enter a path for a Python file you wish to use to hold your Flask app. **If this file already exists, its contents will be overwritten with the new app.**

Path

/home/adilanordin/mysite/flask_app.py

7. Your web app is now set up successfully.


pythonanywhere by ANACONDA
Dashboard Consoles Files Web Tasks Databases

All done! Your web app is now set up. Details below.

adilanordin.pythonanywhere.com
Add a new web app

Configuration for adilanordin.pythonanywhere.com

Reload:

Reload adilanordin.pythonanywhere.com

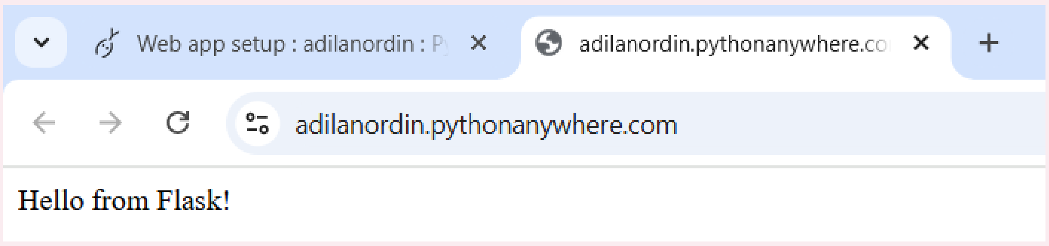
Best before date:

We're happy to host your free website -- and keep it free -- for as long as you want to keep it running, but you'll need to log in at least once every three months and click the "Run until 3 months from today" button below. We'll send you an email a week before the site is disabled so that you don't forget to do that. See here for more details.

This site will be disabled on **Sunday 30 November 2025**

Paying users' sites stay up forever without any need to log in to keep them running.

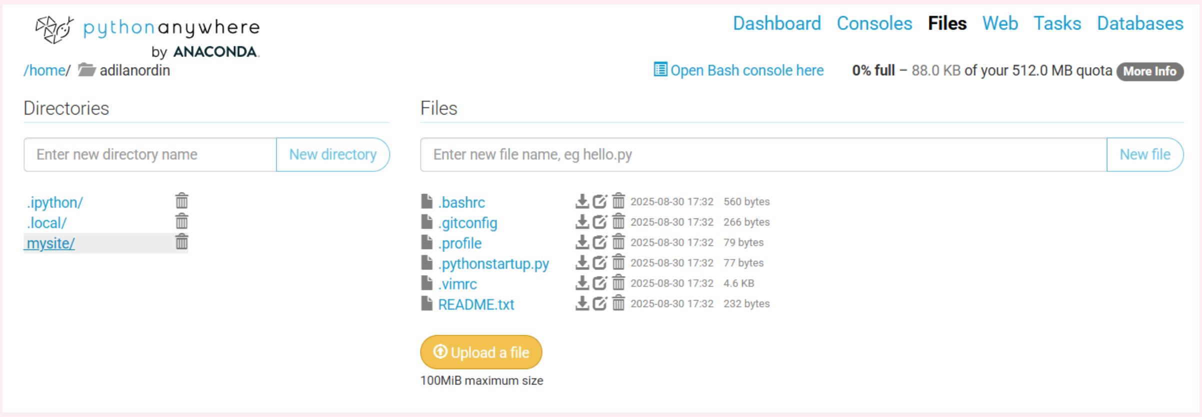
8. Click on the link `yourname.pythonanywhere.com` to view the example `Hello from Flask!`


Web app setup : adilanordin : P
adilanordin.pythonanywhere.co
adilanordin.pythonanywhere.com

Hello from Flask!

Part 2: Hello from Flask!

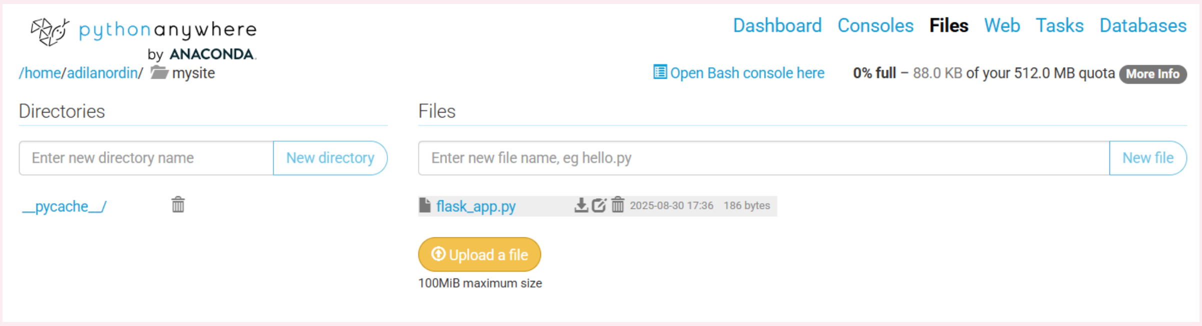
1. To view the file that displays `Hello from Flask!`, click **Files**.



.ipython/
.local/
mysite/

```
.bashrc            2025-08-30 17:32  560 bytes
.gitconfig         2025-08-30 17:32  266 bytes
.profile           2025-08-30 17:32  79 bytes
.pythonstartup.py  2025-08-30 17:32  77 bytes
.vimrc             2025-08-30 17:32  4.6 KB
README.txt         2025-08-30 17:32  232 bytes
```

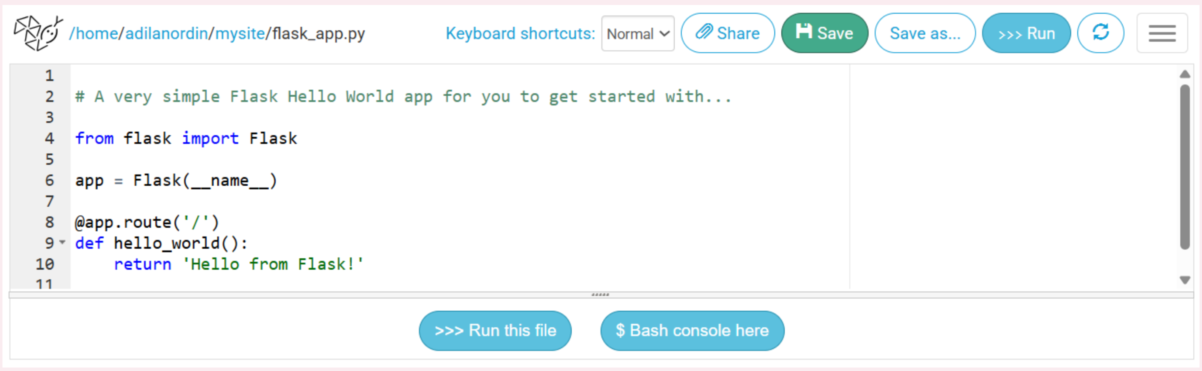
2. Then, click **mysite/**. You will see the file that was previously created, **`flask_app.py`**.



__pycache__/

```
flask_app.py  2025-08-30 17:36  186 bytes
```

3. Click **`flask_app.py`** or **Edit** to open and view the code.


/home/adilanordin/mysite/flask_app.py
Keyboard shortcuts: Normal
Share
Save
Save as...
>>> Run
>>> Run this file
$ Bash console here

```python

# A very simple Flask Hello World app for you to get started with...

from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello from Flask!'

```

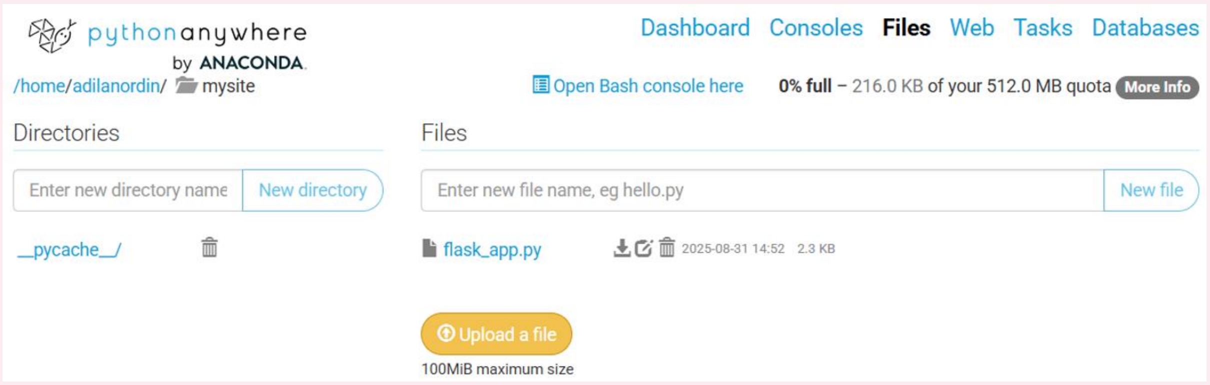
4. Try modifying the code and see what happens. Does it change as expected? If not, why?

Part 3: Reflective Journal Example

1. Recreate the Reflective Journal example from the lecture slides.
2. It will have the following structure:

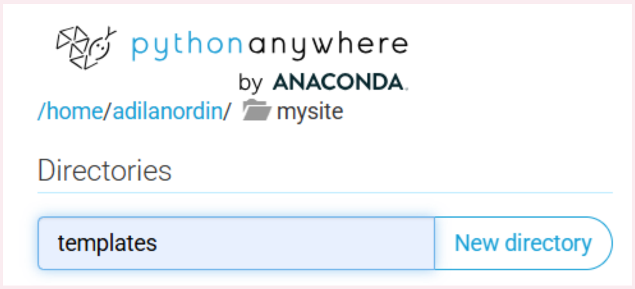
```
/home/yourusername/mysite/
    flask_app.py
    templates/
        form4.html
    static/
        form4JS.js
```

3. As you already created the **flask_app.py** file, copy and paste this Python code into it.

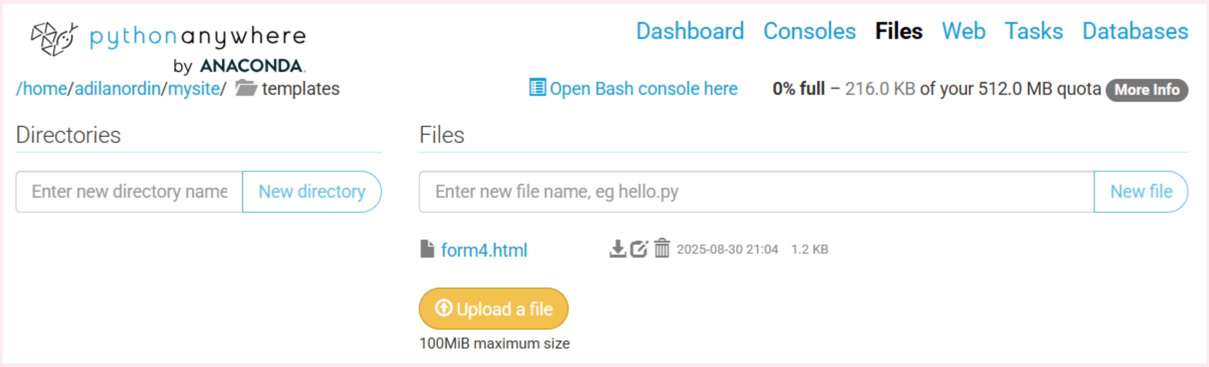

pythonanywhere by ANACONDA
Dashboard Consoles Files Web Tasks Databases
/home/adilanordin/ mysite
Open Bash console here
0% full – 216.0 KB of your 512.0 MB quota More Info
Directories
Files
Enter new directory name New directory
Enter new file name, eg hello.py New file
__pycache__/
flask_app.py 2025-08-31 14:52 2.3 KB
Upload a file
100MiB maximum size

```python
# Flask → web framework we use.
# request → gets data sent from the client (form submission in JSON).
# jsonify → converts Python data into JSON for responses.
# render_template → loads the form4.html file from the templates/ folder.
# json, os → for file reading/writing.
# datetime → to add a date when a reflection is submitted.
from flask import Flask, request, jsonify, render_template
import json, os
from datetime import datetime

# Creates the Flask application object.
app = Flask(__name__)

# BASE_DIR = current folder.
# DATA_FILE = path to reflections.json, the file where reflections are stored.
BASE_DIR = os.path.dirname(os.path.abspath(__file__))
DATA_FILE = os.path.join(BASE_DIR, "reflections.json")

# Function to read reflections from reflections.json.
# If the file does not exist, return an empty list.
def load_reflections():
    if os.path.exists(DATA_FILE):
        with open(DATA_FILE, "r") as f:
            return json.load(f)
    return []

# Function to save reflections back into reflections.json.
# indent=4 makes it human-readable.
def save_reflections(reflections):
    with open(DATA_FILE, "w") as f:
        json.dump(reflections, f, indent=4)

# Flask will display the file in render_template()
# Defines the route / (homepage).
# Loads form4.html (the form page).
@app.route("/")
def index():
    return render_template("form4.html")

# Defines /api/reflections with GET method.
# Loads reflections from the JSON file.
# Returns them as JSON to the front-end.
@app.route("/api/reflections", methods=["GET"])
def get_reflections():
    reflections = load_reflections()
    return jsonify(reflections)

# Defines /api/reflections with POST method.
# Gets JSON data from the client (request.get_json()).
# Creates a new reflection entry (adds today’s date).
# Loads all existing reflections, appends the new one, and saves.
# Returns the newly added reflection as a response.
@app.route("/api/reflections", methods=["POST"])
def add_reflection():
    data = request.get_json()
    new_reflection = {
        "name": data["name"],
        "date": datetime.now().strftime("%a %b %d %Y"),
        "reflection": data["reflection"]
    }
    reflections = load_reflections()
    reflections.append(new_reflection)
    save_reflections(reflections)
    return jsonify(new_reflection), 201
```

4. Enter new directory name **templates** and click **New directory**.



5. In the **templates/** directory, **Enter new file name** or **Upload a file**. For example, `form4.html`. Copy this HTML code into the file.



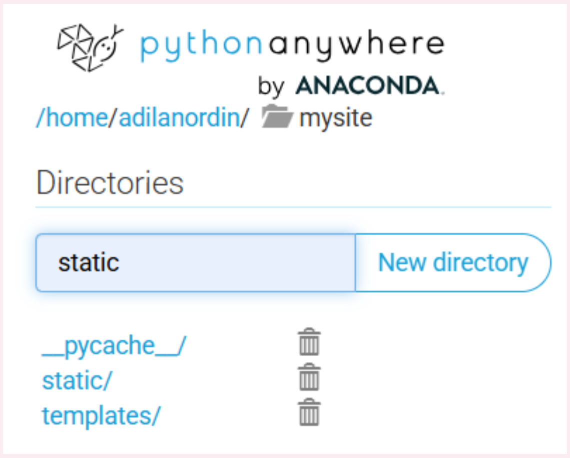
/home/adilanordin/mysite/ templates

0% full – 216.0 KB of your 512.0 MB quota

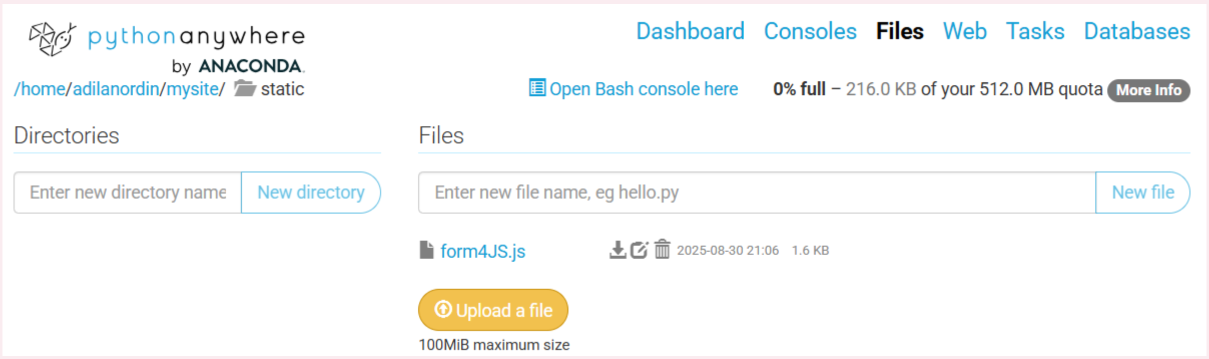
form4.html 2025-08-30 21:04 1.2 KB

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Reflective Journal</title>
    <!-- Links external JavaScript file located in the Flask static folder
     url_for('static', filename='form4JS.js') tells Flask to serve
static/form4JS.js -->
    <script src="{{ url_for('static', filename='form4JS.js') }}"></script>
</head>

<body onload="init()">
    <h2>Reflective Journal</h2>
    <p id="todayDate"></p>

    <!-- onsubmit="return checkReflection()" → sends data using JS instead of
reloading page -->
    <form name="myForm" onsubmit="return checkReflection()">
        <label for="fname">Name: </label><br>
        <input type="text" id="fname" name="fname" size="53" required><br><br>

        <label for="reflection">Reflection: </label><br>
        <textarea id="reflection" name="reflection" minlength="10"
            placeholder="What did you learn this week? What did you find
challenging?" rows="4" cols="50"
            required></textarea><br><br>

        <!-- Submit button → triggers JS checkReflection() -->
        <input type="submit" value="Submit">
        <input type="reset" value="Reset"><br><br>
    </form>

    <h2>Previous Reflections</h2>
    <div id="viewAll"></div>
</body>
</html>
```

6. Go back to the **mysite/** directory and enter new directory named **static** and click **New directory**.

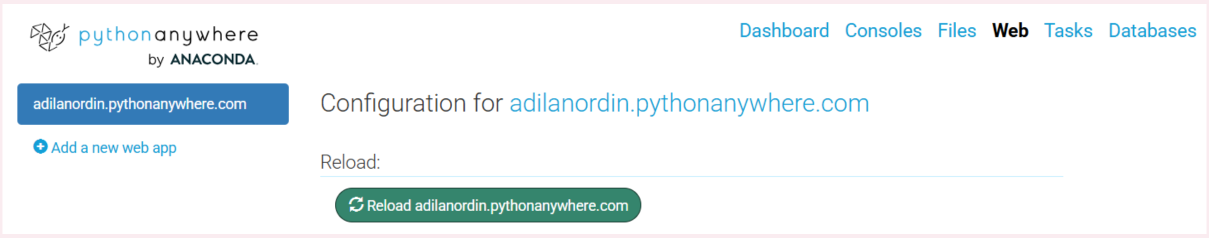


7. In the **static/** directory, **Enter new file name** or **Upload a file**. For example, **`form4JS.js`**. Copy this JavaScript code into the file.

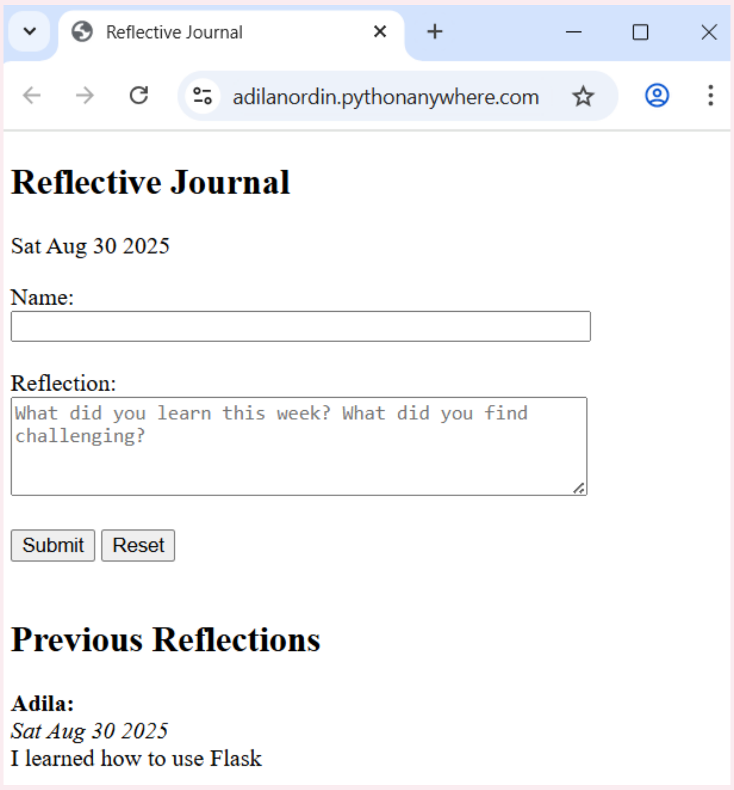

pythonanywhere by ANACONDA
Dashboard Consoles Files Web Tasks Databases
/home/adilanordin/mysite/ static
Open Bash console here
0% full – 216.0 KB of your 512.0 MB quota More Info
Directories
Enter new directory name
New directory
Files
Enter new file name, eg hello.py
New file
form4JS.js 2025-08-30 21:06 1.6 KB
Upload a file
100MiB maximum size

```
function getDate() {
    const d = new Date();
    let text = d.toDateString();
    document.getElementById("todayDate").innerHTML = text;
}

async function checkReflection() {
    // Reads input values from the form (name + reflection)
    let name = document.getElementById("fname").value;
    let reflection = document.getElementById("reflection").value;

    // Creates a JSON object entry
    let entry = { name, reflection };

    // Sends new reflection to Flask server via POST request
    let response = await fetch("/api/reflections", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify(entry)
    });

    if (response.ok) {
        document.myForm.reset();
        submitted(); // refresh list
    }
    return false; // prevent page reload
}

async function submitted() {
    let output = "";

    // Sends a GET request to /api/reflections to load all reflections
    let response = await fetch("/api/reflections");
    if (response.ok) {
        let reflections = await response.json();

        for (let r of reflections) {
            output += "<b>" + r.name + ":</b><br>" +
                "<i>" + r.date + "</i><br>" +
                r.reflection + "<br><br>";
        }

        if (reflections.length === 0) {
            output = "<i>No reflections found.</i>";
        }
    } else {
        output = "<i>Error loading reflections.</i>";
    }
    document.getElementById("viewAll").innerHTML = output;
}

function init() {
    getDate();
    submitted();
}
```

8. Go to **Web** and click **Reload**.


pythonanywhere
by ANACONDA
Dashboard
Consoles
Files
Web
Tasks
Databases
adilanordin.pythonanywhere.com
Add a new web app
Configuration for adilanordin.pythonanywhere.com
Reload:
Reload adilanordin.pythonanywhere.com

9. Open your web app link. You should now see the **Reflective Journal** form.



Reflective Journal

Sat Aug 30 2025

Name:

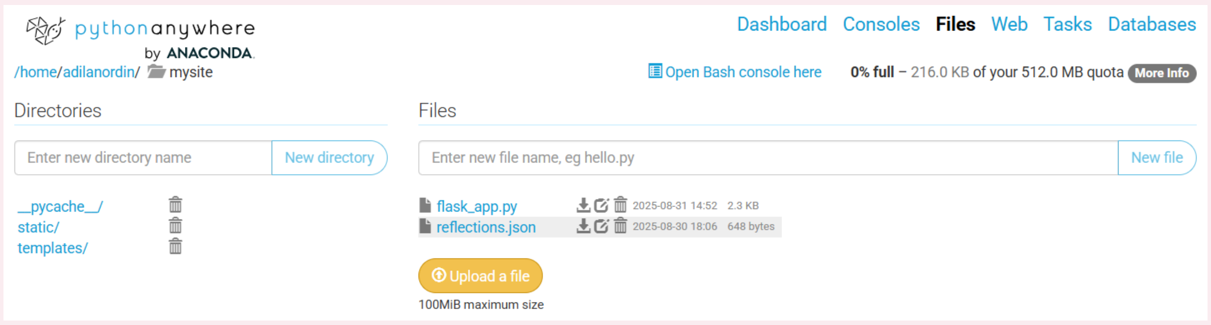
Reflection:

What did you learn this week? What did you find challenging?

Previous Reflections

Adila:
Sat Aug 30 2025
I learned how to use Flask

10. Fill in the form with **Name** and **Reflection**. Then return to **Files** and notice a new file, **`reflections.json`** has been created automatically.



/home/adilanordin/ mysite

0% full – 216.0 KB of your 512.0 MB quota

__pycache__/		
static/		
templates/		

flask_app.py	2025-08-31 14:52	2.3 KB
reflections.json	2025-08-30 18:06	648 bytes

100MiB maximum size

11. You have now successfully integrated the frontend with the backend.

Task 2 Workshop Overview

You will extend your Learning Journal PWA by introducing **Flask as a backend framework** and deploying it on **PythonAnywhere**. While HTML, CSS, and JavaScript control the interface, Flask will handle requests between the frontend and backend. Reflections will be stored in a JSON file and served dynamically through Flask routes.

Your goal this week is to:

- Set up a Flask backend that serves and updates reflections stored in a JSON file.
- Use JavaScript's Fetch API to communicate with Flask routes.
- Deploy and test your PWA directly on PythonAnywhere.

Goal

Add a Flask backend to your Learning Journal PWA. The backend should manage the JSON file and respond to frontend requests. You will test the app live on PythonAnywhere, bypassing local testing.

All files should be committed to your **GitHub** repository so progress is visible and version-controlled.

What to Do

The following tasks are suggestions. You may adapt them as appropriate to your own design.

Folder and File Structure

Flask works best with its conventional **static/** and **templates/** folders. A possible structure:

`/mysite`	
`flask_app.py`	Main Flask backend file
`/templates`	
`index.html`	HTML pages served by Flask
`/static`	Client-side assets
`/css`	Stylesheets
`/js`	JavaScript files
`/images`	Media assets
`/backend`	Backend data files
`reflections.json`	JSON file storing reflections

Flask Backend

1. Upload or create the project files on PythonAnywhere.
2. Create a Flask application, `flask_app.py` that can handle requests for your reflections.
3. Implement at least two routes:
   - `GET /reflections`: Returns `reflections.json` as JSON.
   - `POST /add_reflection`: Accepts JSON from frontend and appends it to `reflections.json`.

Frontend (HTML, CSS, JavaScript)

1. Fetch Reflections from Flask
    - Update your PWA to fetch reflections from your Flask backend instead of reading the JSON file directly.
    - Submit new reflections to the backend.
    - Ensure the frontend updates dynamically when new reflections are added.

2. DOM Manipulation
    - Use your existing HTML/CSS for styling.
    - Insert entries dynamically into the DOM.

3. Extend with an Extra Feature
    - Add **at least one new feature** that uses the Flask backend.
    - Examples (suggestions only you may come up with your own idea):
        - Delete a reflection via a `DELETE` request.
        - Edit an existing reflection with a `PUT` request.
        - Search or filter reflections on the server.

Journal Questions

For your weekly entry, **submit it on myUCA** and **post it on the Learning Journal Web**:

1. Why is the frontend–backend connection important?
2. Which HTTP methods did you use in Flask, and why?
3. What is the difference between using Flask to store and load JSON data and reading JSON directly in the browser?
4. Did you face any difficulties when running your project on PythonAnywhere? How did you handle them?
5. What extra feature did you build into your PWA with Flask, and why did you add it?