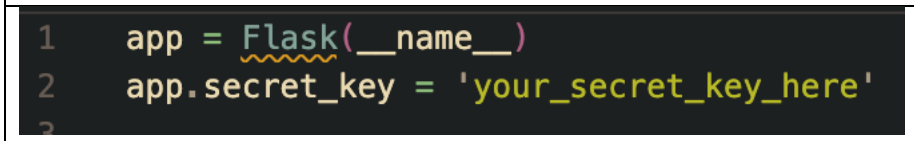
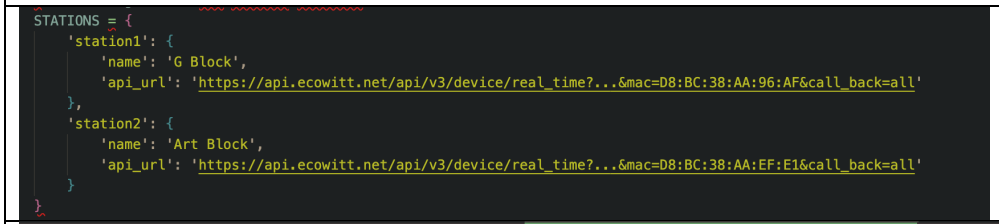
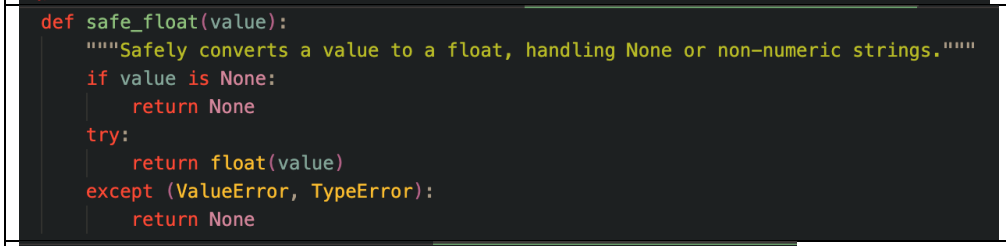
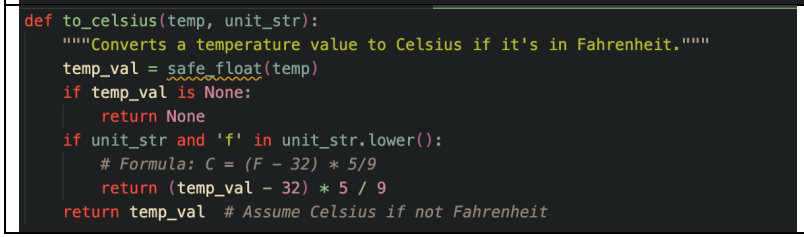
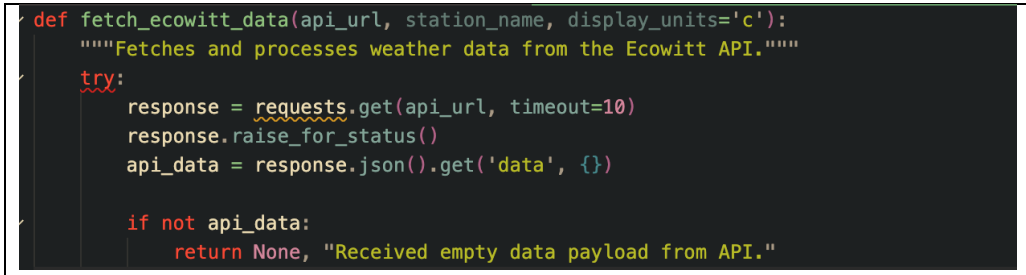
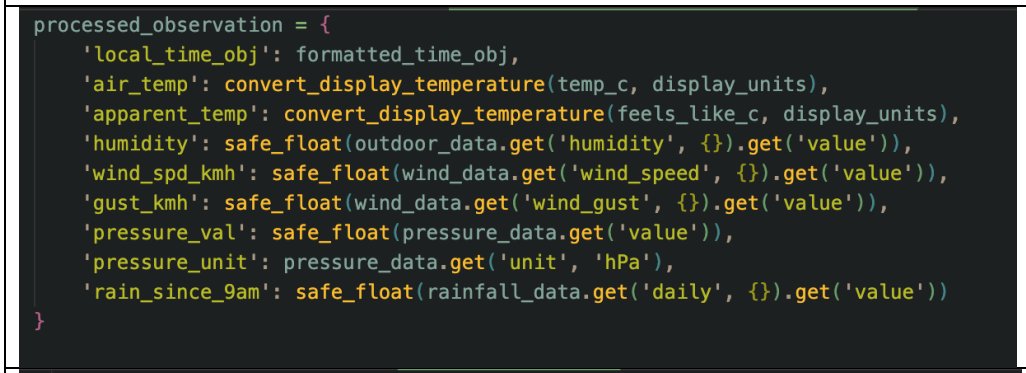
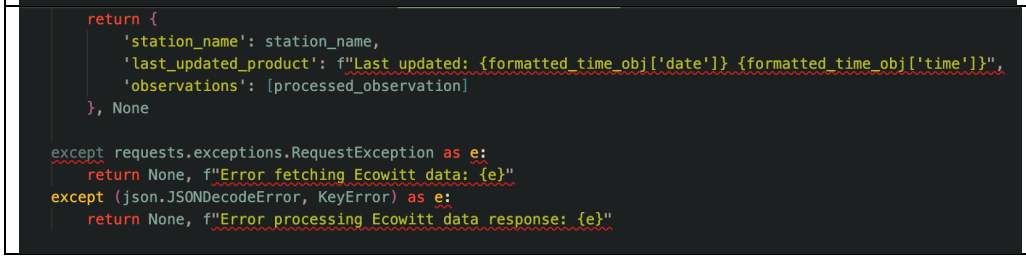
	• `json`: to parse API responses and handle JSON errors `requests`: for HTTP calls to the Ecowitt API • `flask` imports: core Flask object and helper functions for rendering templates, managing sessions, and building redirects • `urllib3`: to suppress SSL verification warnings (CERTIFICATE_VERIFY_FAILED) • `datetime`: timestamp conversion from API epoch seconds • `functools.wraps`: decorator utility to preserve function metadata
	• Initialize Flask app. • `secret_key` secures session cookies; should be moved to environment variables for safety
	• Keys (`station1`, `station2`) used in URL query parameters. • Values include a human-readable `name` and the full `api_url`.
	Normalizes numeric inputs; returns `None` on invalid or missing data. Ensures downstream calculations won't crash if the API returns unexpected types
	Unit-aware conversion from Fahrenheit to Celsius. Gracefully handles missing unit information

	• Perform HTTP GET with a 10 s timeout. • `raise_for_status()` throws HTTPError on bad status codes. • Check for empty data payload; return a user-friendly error message if missing
	• Build a normalized dict for one observation. • Units: speed in km/h, pressure in hPa by default. This payload can easily be extended to include additional fields
	• On success: return a tuple `(data, None)` • On exceptions: return `(None, error_message)` for clear upstream handling

```python
titled-1
ute('/login', methods=['GET', 'POST'])
def login():
    error = None
    if request.method == 'POST':
        username = request.form.get('username')
        password = request.form.get('password')
        if username == 'admin' and password == 'admin':
            session['logged_in'] = True
            return redirect(url_for('home'))
        else:
            error = 'Invalid credentials. Please try again.'
    return render_template('login.html', error=error)
```

- GET displays login page.
- POST validates hard-coded credentials
- On success: set session flag and redirect to home.
- On failure: redisplay form with `error` message

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Login - Weather App</title>
    <link rel="stylesheet" href="/static/style.css">
    <style> … </style>
</head>
<body>
    <div class="login-container">
        <div class="login-title">Weather App Login</div>
        <form class="login-form" method="post">
            {% if error %}
                <div class="error-message">{{ error }}</div>
            {% endif %}
            <label for="username">Username</label>
            <input type="text" id="username" name="username" required autofocus>
            <label for="password">Password</label>
            <input type="password" id="password" name="password" required>
            <button type="submit">Login</button>
        </form>
    </div>
</body>
</html>
```

- **Lines 1–6**:
  - Declares HTML5 `<!DOCTYPE>` and root `<html lang="en">` for accessibility.
  - Meta charset ensures proper Unicode rendering.
  - External stylesheet link for shared styles.
- **Lines 7–62** (inline <style> block):
  - Overrides or supplements shared CSS for login page only.
  - Sets background, typography, and container card styles.
  - Splits into subsections:
    1. `.login-container`: centering and card styling.
    2. `.login-title`: typography and spacing.
    3. `.login-form label` & inputs: accessibility, form spacing.
    4. Buttons: full-width, hover transitions.

	5. `.error-message`: visibility of server-side errors • **Lines 64–78**: • Semantic grouping: a single `<form>` with method=POST. • Jinja2 conditional to display `error`. • `<label>` + `<input>` pairs link via `for`/`id` for screen-reader support. • `required` and `autofocus` improve UX and validation.