# Beyond Advanced Python: Exercise 3

## Overview

A Flask-based app is needed. Routes should lead to:

- A simple home page offering links to the other routes
- An image of a kitten, with parameterized width and height.
  HTML Template is here: https://pastebin.com/k778Xy4s
- A map of Athlone.
  An html template (using the OpenStreetMap API) is here: https://pastebin.com/BGg29tUx
- A 404 handler for all routes that are not matched

  ```
  @app.errorhandler(404)
  def page_not_found(error):
  return render_template('page_not_found.html'), 404
  ```

- A template showing the weather temperature for a selection of one or two cities, provided as URL parameters. This route can call your weather app, passing the city or cities in and return temperatures. Or use this weather app: https://pastebin.com/TUCggFVQ

## Testing, Timing and Profiling

Construct some useful unit tests to run against parts of your app (e.g. the weather service).
Tests should fail if outcomes are not as expected. Also test that exceptions are thrown as expected.
Gather some timing metrics by using timeit along with some of your functions.
Use @profile to get some memory profile information about some functions. Choose some targets for performance improvement, then explore alternative code structures to see how memory and speed are affected.

## Optional

Parameterize the map template to take lat/lon values entered by the user, then write a parameterized route which takes lat/lon values and show the relevant map.
Write a module called preparation.py which asks the user to enter some default values to use in the Flask site. This module should provide defaults for city, width, height etc.
Persist these values in a simple text file.
When the Flask app starts, read this configuration file and use these default values to prepopulate the forms, so the form fields are not blank when the user first sees them.
Every time the user submits a valid form, persist these new values into the configuration file, so they will be used when the server is next started.
Write tests to confirm it behaves as expected, including edge-cases and exceptions.