Intro to Flask

Brian Cherinka
.Astronomy 8 - June 18, 2016
Oxford, UK

What is Flask

- Flask is a Python Micro-Framework! (http://flask.pocoo.org/docs/0.11/)
- Allows for easy and fast implementation of a web application
- pip install Flask
- pip install Flask-JSGlue (for later)
- pip install Flask-Testing (for later)

Github

- Go to github.com/havok2063/introtoflask
- Clone or Download button, and copy link
- git clone github.com/havok2063/introtoflask .

Quick Start

Copy and paste this into a file called newapp.py

```
#!/usr/bin/env python2
# encoding: utf-8

from __future__ import print_function, division, absolute_import
from flask import Flask

app = Flask(__name__)

@app.route('/')
def main():
    return 'This is my brand new app'

if __name__ == '__main__':
    app.run()
```

Go to its location and type python newapp.py

Navigate to http://localhost:5000

Simple Example

- Go to introtoflask/simpleexample
- python app.py -d -p 8000
- I used argparse to add command line arguments
  - -d tells Flask to run the app in debug mode (do this)
  - -p lets me specify an optional port to run the app on

Routes

- Routes decorate your methods and define the URL
- They can make your URLs pretty and memorable

```
@app.route('/add/', endpoint='doadd')
@app.route('/addnumbers/')
def add():
```

- You can set multiple routes per method.
- To reference a route internally in your code, use endpoints!
- Endpoints - sets the name of the URL endpoint, otherwise uses function name

Routes - Variables

- Variable rules or inputs [@app.route(‘/hello/<name>/')]
- Set defaults [@app.route('/hello/', defaults={'name': ‘Bob’})]
- Use converters [@app.route(‘/subtract/<int:x>/<int:y>/')]
  - allowed: int, float, string (no slashes), path (like string but accepts slashes)

Routes - Methods

- Routes accept Methods on them, e.g. GET or POST
- Set with the methods=[‘GET’, ‘POST’] route keyword
- GET - parameter passing straight in the url (default)
  - e.g. www.badwebsite.com/login?username=Brian&password=12345
- POST - encoded parameter passing (html forms, json objects)
  - e.g. form = {‘username’: Brian, ‘password’: 12345}
- Flask has a global object to capture any requests
  - from flask import request (see use in myapp/controllers/__init__)
  - like the Python requests package

url_for and redirect

- url_for generates a string relative url path inside your app ; can be used anywhere
  - Useful for dynamic changes to route urls (no hard-coding)
  - Formatting
    - url_for(method_name, **values)
    - url_for(endpoint_name, **values) if endpoint is set
    - url_for(blueprint.method/endpoint, **values) if route in a blueprint

- redirect redirects one route path to another

```
@app.route('/addagain/')
def do_more_adding():
    addurl = url_for('doadd')
    return redirect(addurl)
```

- see do_more_adding inside app.py

A Real App

- Complete Separation of Data, Back-end Code, and UI/UX code
- In Model-Controller-View (MCV) framework
  - Controllers - Your server side code
  - Views - Your front end html template code
  - Model - Your data source (e.g. a database)
- Go to introtoflask/myapp and run python run_myapp.py -d -p 9000

Blueprints

- A way of modularizing your code into chunks (pages?)

```
# In your individual pages
from flask import Blueprint
index_page = Blueprint("index_page", __name__)

@index_page.route('/', methods=['GET'])
def index():
    …

# In your app init, you must register your blueprint with your app
from myapp.controllers.index import index_page
app = Flask(__name__)
…
app.register_blueprint(index_page)
return app
```

Blueprints - Organization

- Option A

```
myapp/
    controllers/
        home
        admin
        user
    model/
    static/
        images/
        js/
    templates/
        home
        admin
        user
```

- Option B

```
myapp/
    home/
        controllers/
        static/
        templates/
    admin/
        controllers/
        static/
        templates/
    user/
        controllers/
        static/
        templates/
```

- Option C - or do whatever ..

Jinja2 Html Templates

- Control and Customize the data sent to the front-end
- Modularize your HTML
- from flask import render_template

```
@example_page.route('/examples/')
def example():
    output = {}
    output['title'] = 'MyApp Examples'
    output['page'] = 'example
    return render_template('examples.html', **output)
```

Jinja2 Template Syntax

- {{ variable }} - substitutes parameter from back-end into its place
- Variables passed in as dictionary parameters
- {# comments #}
- Conditionals, Code Insertions, Loops, Variable Assignment, etc
- http://jinja.pocoo.org/docs/dev/templates/

Jinja2 Filters and Tests

- Don’t worry about visual presentation of data on back-end
- Jinja2 has filters to sort out and format your data (to an extent) and tests for expression testing
- See http://jinja.pocoo.org/docs/dev/templates/#builtin-filters
- Can also make Custom Filters (python code)
- See myapp/jinja_filters.py and the registration of the filters in myapp/_init_.py

g and session

- Flask allows for variables to be passed around within special objects
- g - stores variables persistent within a single request , safe in threaded envs
  - from flask import g
  - Storing: g.ra = 234.2344
  - Retrieving: ra = g.get(‘ra’, None)
- session - stores variables persistent across all requests in an entire browser session (cookies)
  - from flask import session [as current_session]
  - Storing: session[‘ra’] = 234.2344
  - Retrieving: ra = session[‘ra’] or session.get(‘ra’, None)
  - session behaves like a Python dictionary

Hosting Your App

- Host your locally running app with ngrok
- Host a production version with uwsgi+Nginx or Apache
- Host on a cloud-based platform (e.g Heroku)

ngrok

- Allows for secure tunnels to localhost
- Quick and Dirty
- https://ngrok.com/
- ngrok http [port number]
- Forwarding - http://24f55f75.ngrok.io
- Web Interface for Stats - http://127.0.0.1:4040

Profiling Your App

- How to time or profile your Flask app?
- Flask has a built in profiler with Werzeug, which uses cProfile under the hood
- see profile_myapp.py
- runs like an app: python profile_myapp.py -p 9000

```
from flask import Flask
from werkzeug.contrib.profiler import ProfilerMiddleware

app.config['PROFILE'] = True
app.wsgi_app = ProfilerMiddleware(app.wsgi_app)
app.run(debug = True, port=args.port)
```

Unit testing Your App

- Flask has a built-in testing suite, made a bit easier with the extension Flask-Testing
- see introtoflasks/myapp/tests

Resources

- Flask has a ton of extensions.
  - Some - http://flask.pocoo.org/extensions/
  - More - https://github.com/humiaozuzu/awesome-flask
- Try the Flask Snippets for code tips
  - http://flask.pocoo.org/snippets/
- Stackoverflow - http://stackoverflow.com/questions/tagged/flask