Getting Started With Python and Flask

Things I wished I would have known or read about before I dove into Python and Flask…

Burlington Python - January 2017

What is Python?

Python is a widely used high-level, general-purpose, interpreted, dynamic programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than possible in languages such as C++ or Java. The language provides constructs intended to enable writing clear programs on both a small and large scale.

Modules

A Python module is simply a Python source file, which can expose classes, functions and global variables. When imported from another Python source file, the file name is treated as a namespace.

A module is a single file (or files) that are imported under one import and used. e.g.

import os

Packages

A package is a collection of Python modules: while a module is a single Python file, a package is a directory of Python modules containing an additional __init__.py file, to distinguish a package from a directory that just happens to contain a bunch of Python scripts.

A package is a collection of modules in directories that give a package hierarchy.

from flask import Flask, session

pip

pip is a package management system used to install and manage software packages written in Python. Many packages can be found in the Python Package Index (PyPI).

To install a package, run:

```
pip install Flask
```

virtualenv

virtualenv is a tool to create isolated Python environments. virtualenv creates a folder which contains all the necessary executables to use the packages that a Python project would need.

Creating a virtualenv will help you keep different projects on different versions of python, install only the packages you need and not BREAK your other projects running on the same host. When you set up the environment, it will come with the basic Python libraries and anything additional can be downloaded via pip.

virtualenv setup - keep your environments separated

```
# yum install python-devel python-setuptools python-pip
# cd /home/user/project
# virtualenv env
# source env/bin/activate
(env) <- this means you’re in your virtualenv environment
# pip install Flask
```

Python Terms
https://docs.python.org/2/glossary.html

Argument: A value passed to a function (or method) when calling the function.

```
complex(3, 5)
```

Immutable: An object with a fixed value. Immutable objects include numbers, strings and tuples. Such an object cannot be altered. A new object has to be created if a different value has to be stored. They play an important role in places where a constant hash value is needed, for example as a key in a dictionary.

Mutable: Mutable objects can change their value but keep their **id()**. See also immutable.

Statement: A statement is part of a suite (a “block” of code). A statement is either an expression or one of several constructs with a keyword, such as **if**, **while** or **for**.

Python Terms

https://docs.python.org/2/glossary.html

Class: A template for creating user-defined objects. Class definitions normally contain method definitions which operate on instances of the class.

Decorator: A function returning another function, usually applied as a function transformation using the @wrapper syntax.

Dictionary: An associative array, where arbitrary keys are mapped to values. The keys can be any object with __hash__() and __eq__() methods. Called a hash in Perl.

Function: A series of statements which returns some value to a caller.

List: A built-in Python sequence. Despite its name it is more akin to an array in other languages than to a linked list since access to elements are O(1)

Method: A function which is defined inside a class body.

Flask

Flask is a micro web framework written in Python and based on the Werkzeug toolkit and Jinja2 template engine

http://flask.pocoo.org/

Ok, so what is werkzeug and Jinja2?

Werkzeug - WSGI library

Werkzeug is a WSGI utility library for Python. It's widely used and BSD licensed.

Werkzeug is the base of frameworks such as Flask and more, as well as in house frameworks developed for commercial products and websites.

http://werkzeug.pocoo.org/

The **Web Server Gateway Interface** (**WSGI**) is a specification for simple and universal interface between web servers and web applications or frameworks for the Python programming language.

Jinja2 - Templates

Jinja is a template engine for the Python programming languag. It is similar to the Django template engine but provides Python-like expressions while ensuring that the templates are evaluated in a sandbox. It is a text-based template language and thus can be used to generate any markup as well as sourcecode.

The Jinja template engine allows customization of tags,[1] filters, tests, and globals. Also, unlike the Django template engine, Jinja allows the template designer to call functions with arguments on objects. Jinja is Flask's default template engine.

http://jinja.pocoo.org/

Routes

When a route is visited, then flask will execute the commands in the route and generally return a response or push variables forward for a template to render.

Simple Return:

```
@app.route('/')
@app.route('/index')
def index():
    return "Hello, World!"
```

Note: The @ symbol is used for class, function and method *decorators*.

Template - render_template

You can also create a template html file and then iterate through variables to display in the template

Template File:

```
<html>
  <head>
    <title>{{ title }} - microblog</title>
  </head>
  <body>
      <h1>Hello, {{ user.nickname }}!</h1>
  </body>
</html>
```

Note: The {{ }} bars are used to insert variables into the HTML that is generated

Routes with variables in request

```
@app.route('/<user>')
@app.route('/index/<user>')
def index(user):
    return render_template('index.html',
                           title='Home',
                           user=user)
```

Note: The <user> variable is interpreted when the user visits the URL, for example user “bob”:
http://127.0.0.1/bob
http://127.0.0.1/index/bob

Jinga2 - Example

```
{% extends "layout.html" %}
{% block body %}
  <ul>
  {% for user in users %}
    <li><a href="{{ user.url }}">{{ user.username }}</a></li>
  {% endfor %}
  </ul>
{% endblock %}
```

Note: The for loop to interate through every user in a dictionary, list or tuple of “users”. In this case, display each variable entry for users[‘url’] and users[‘username’]. Jinja supports dot (.) to access attributes of a variable as well.

Code Example:

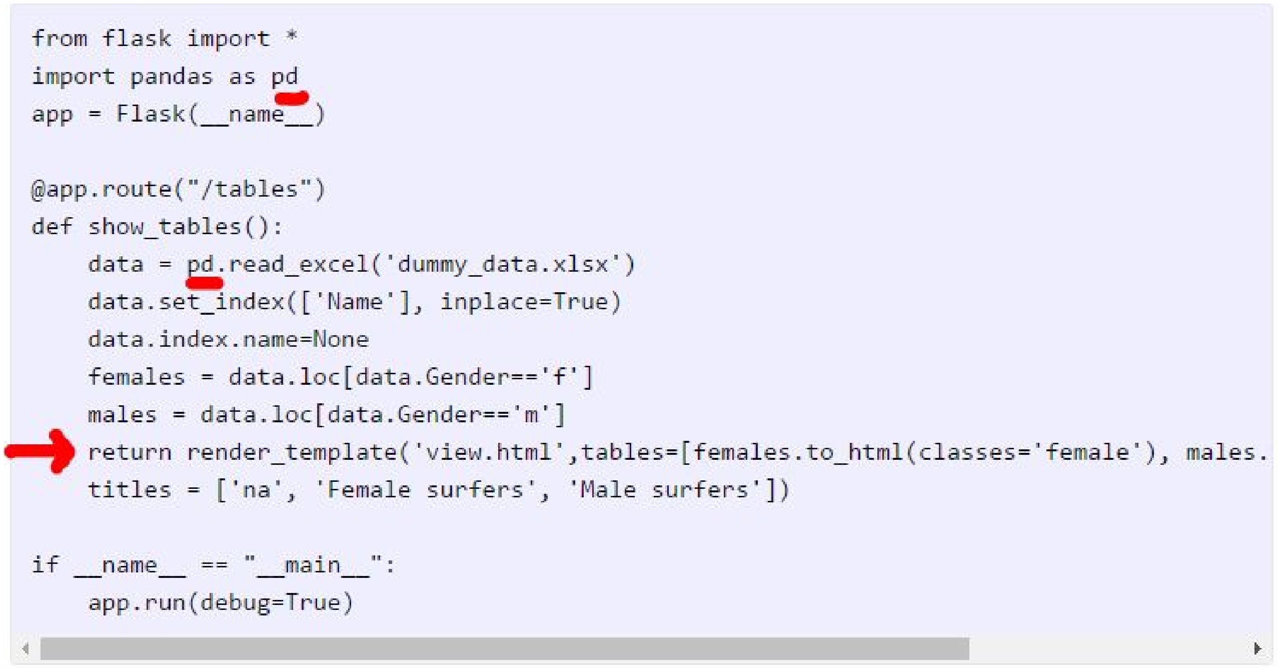
https://sarahleejane.github.io/learning/python/2015/08/09/simple-tables-in-webapps-using-flask-and-pandas-with-python.html

Import: as
- Reference as “pd”

Route: @app.route
- Definition and subdir

Return: render_template
- HTML, variables

Main: app.run
- Start up server
- Debugging on



```
from flask import *
import pandas as pd
app = Flask(__name__)

@app.route("/tables")
def show_tables():
    data = pd.read_excel('dummy_data.xlsx')
    data.set_index(['Name'], inplace=True)
    data.index.name=None
    females = data.loc[data.Gender=='f']
    males = data.loc[data.Gender=='m']
    return render_template('view.html',tables=[females.to_html(classes='female'), males.
    titles = ['na', 'Female surfers', 'Male surfers'])

if __name__ == "__main__":
    app.run(debug=True)
```

Learn Flask

https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world

https://www.fullstackpython.com/flask.html

https://pythonspot.com/en/flask-web-app-with-python/

Learn Python

https://www.learnpython.org/

https://learnpythonthehardway.org/book/

https://play.google.com/store/apps/details?id=com.sololearn.python&hl=en