# Course Name: Practical Python Programming

## Week 5: House Keeping Work and Django Framework


Course Name: Practical Python Programming ..... 1
Week 5: House Keeping Work and Django Framework..... 1
The original outline ..... 1
a. GitHub. ..... 2
b. SQLite. ..... 2
c. BeautifulSoup and Web Scraping. ..... 2
d. More BeautifulSoup, Web Scraping and a bit of Big Data. ..... 2
e. Web Development Introduction. ..... 3
f. Common Python Library and Package Management. ..... 3
g. Introduction to Django. ..... 3
h. Django Basics..... 3
a. Creating your Django Project..... 4
j. Creating a Django Application. ..... 4
k. Creating a View. ..... 4
l. Mapping URLs and Routing to the URLs. ..... 4


### *The original outline*

a) *Python 3 vs. Python 2.7.*
b) *Common Python Library and Package Management.*
c) *Introduction to PyCharm Python Development Tools.*
d) *Web Development Introduction.*
e) *Introduction to Python based Web Development Framework.*
f) *Introd.uction to Django.*
g) *What we want to accomplish om the next 4 weeks?*
h) *Installation libraries for the project.*
i) *Virtual Environment*
j) *Django Basics*
    a) *Creating your Django Project*
    b) *Creating a Django Application*
    c) *Creating a View*
    d) *Mapping URLs*
    e) *Basic Workflows*
        a) *Creating a new Django Project*
        b) *Creating a new Django application*

### *a. GitHub.*

Go to this URL:

https://github.com/

- Eclipse Integration
- Push, Committ, Pull, Clone.

### *b. SQLite.*

- Connecting to an SQLite database
- Creating a new SQLite database
- Adding new columns
- Inserting and updating rows
- Creating unique indexes
- Querying the database - Selecting rows
- Security and injection attacks
- Date and time operations
- Update Mar 16, 2014:
- Retrieving column names
- Printing a database summary

### *c. BeautifulSoup and Web Scraping.*

- Install `BeautifulSoup`.
- Web Page, HTML tags.
- Scaping Rules.
- Inspecting the Page by using FireBug (or any browser?).
- Actual Code.
- Export to CSV file.
- More Complicated Scraping.

### *d. More BeautifulSoup, Web Scraping and a bit of Big Data.*

- Install `requests` library.
- Inspecting the Page (`http://www.weather.gov/`).
- Using CSS Selectors.
- Extracting all the information from the page.
- Actual Code.
- Install Panda.
- Combining our data into a Pandas Dataframe.

## *e. Web Development Introduction.*

- HTML.
- CSS.
- Javascript.
- Server Side Coding.

## *f. Common Python Library and Package Management.*

- Installing *Django*.

  ```
  pip install -U django==1.7
  ```

- Installing the Python Imaging Library: *pillow*

  ```
  pip install pillow
  ```

- Installing Other Python Packages.

  ```
  $ pip list
  ```

- Sharing your Package List.

  ```
  pip freeze > requirements.txt
  pip install -r requirements.txt
  ```

## *g. Introduction to Django.*

From Wikipedia: *Django is a free and open-source web framework, written in Python, which follows the model-view-template (MVT) architectural pattern.[6][7] It is maintained by the Django Software Foundation (DSF), an independent organization established as a 501(c)(3) non-profit. Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes reusability and "pluggability" of components, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models. Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models. Some well-known sites that use Django include the Public Broadcasting Service,[8] Pinterest, Instagram, Mozilla, The Washington Times, Disqus, Bitbucket, and Nextdoor.*

## *h. Django Basics.*

- It provides a method of mapping requested URLs to code that handles requests. In other words, it gives you a way of designating which code should execute for which URL. For instance, you could tell the framework, “For URLs that look like /users/joe/, execute code that displays the profile for the user with that username.”
- It makes it easy to display, validate and redisplay HTML forms. HTML forms are the primary way of getting input data from Web users, so a Web framework had better make it easy to display them and handle the tedious code of form display and redisplay (with errors highlighted).
- It converts user-submitted input into data structures that can be manipulated conveniently. For example, the framework could convert HTML form submissions into native data types of the programming language you’re using.
- It helps separate content from presentation via a template system, so you can change your site’s look-and-feel without affecting your content, and vice-versa.

- It conveniently integrates with storage layers — such as databases — but doesn't strictly require the use of a database.
- It lets you work more productively, at a higher level of abstraction, than if you were coding against, say, HTTP. But it doesn't restrict you from going "down" one level of abstraction when needed.
- It gets out of your way, neglecting to leave dirty stains on your application such as URLs that contain ".aspx" or ".php".
- It follows the "model-view-controller" (MVC) architecture. Simply put, this is a way of developing software so that the code for defining and accessing data (the model) is separate from the business logic (the controller), which in turn is separate from the user interface (the view).

*a. Creating your Django Project.*

```
$ django-admin.py startproject tango_with_django_project
```

```
$ python manage.py runserver
```

*j. Creating a Django Application.*

```
$ python manage.py startapp rango
```

*k. Creating a View.*

*l. Mapping URLs and Routing to the URLs.*