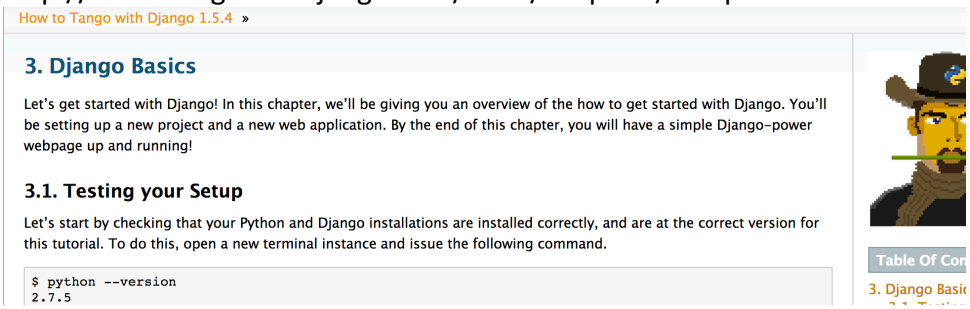
SETTING UP DJANGO'S WEB APPLICATION

1. Copy and paste the link below into your web browser and follow the tutorial in order to set up Django's Web Application on your local machine.

http://www.tangowithdjango.com/book/chapters/setup.html



CREATING HTML PAGES

1. Copy and paste the following link into your web browser in order to follow the tutorial "Bootstrap 3 Tutorial": http://www.w3schools.com/bootstrap/default.asp
2. Please refer to the following pages within the "Bootstrap 3 Tutorial" in order to create the index.html page where the user will enter their email address: "BS Jumbotron", "BS Panels", "CSS Images", "BS Inputs", and "BS Buttons".
3. Please be sure to refer to the django web app tutorial in order to create the index.html page.
4. Using the django web app tutorial, in this index.html page create a jumbotron that includes an image and a panel that includes user input* and a button. *input should take a user's email address
5. Create a HTML page that contains a form for the user to report their maintenance issue
6. Please refer to the pages within the "Bootstrap 3 Tutorial" in order to create the form page where the user will report their maintenance issue: "BS Forms", "BS Panels", "BS Inputs (for inputs and radio)", "BS Dropdowns", "BS Modal", and "BS Buttons".
7. Please be sure to refer to the django web app tutorial in order to create the form page.

8. Using the django web app tutorial, in this form page create a form within a panel that includes input for a user's full name, room number, issue description, and initial. Create dropdowns for residence hall, issue, and issue rank. Use radio input to create a choice for room occupancy. Finally include a button that uses a modal to alert the user once clicked.

How To Create a Modal

The following example shows how to create a basic modal:

Example

```
<!-- Trigger the modal with a button -->
<button type="button" class="btn btn-info btn-lg" data-toggle="modal"
data-target="#myModal">Open Modal</button>

<!-- Modal -->
<div id="myModal" class="modal fade" role="dialog">
  <div class="modal-dialog">

    <!-- Modal content-->
    <div class="modal-content">
```

Setting up django web application to work with html page: Views

1. Access views.py inside your application folder

2. Import the following necessary functions:
from django.shortcuts import render, redirect, render_to_response
from django.core.mail import send_mail, BadHeaderError
from django.http import HttpResponse, HttpResponseRedirect
from AppTicket.forms import ContactForm
from django.template import RequestContext

3. Define your views with the following:

```
def email(request):
return test.html and the form

def thanks(request):
return response check your email

def Contact(request):
return ContactForm.html
```

'Views' Code Content

"email" will hold the validation of the email
Use 'GET' to retrieve information from the ContactForm

Variables for email input

1. Enter Variables For SendMail Function:
Subject of Email
Message of Email
Email for the From: Box
* to_email is a variable from the forms.py that is the inputted email

Handling for Email Validation

```
if '@' in to_email doesn't end with students.ecsu.edu
return response Not a Valid Email
else try:
```

```
sendmail(subject,message,email, [to_email])

return "Check your email" response
```
*This will be done in the def thanks view, return response ('Check Your Email')

LINK URL WITH VIEWS IN URLS.PY (PROJECT FOLDER)

1. Import the following necessary functions:
from django.conf.urls import url, patters (newer django,put *)
from django.contrib import admit
from django AppTicket import views
from MTicket import settings

Enter URL patterns

```
url(r'"name of page"/$', "name of view", "name of the page")
```
Continue to do what's above for the rest of the pages

STORE HTML PAGES

1. Create a "templates" folder in the Project folder and put your html files in that folder

SET UP SMTP SERVER FOR SENDMAIL FUNCTION

1. Copy and paste the link below into your web browser and follow the tutorial for "How to Send Email in a Django Web App"

https://simpleisbetterthancomplex.com/tutorial/2016/06/13/how-to-send-email.html

SET UP YOUR WEB APP TO WORK DYNAMICALLY ACROSS MULTIPLE MACHINES

1. Copy and paste the link below into your web browser and follow the tutorial "4.1.2. Dynamic Paths"
http://www.tangowithdjango.com/book/chapters/templates_static.html#dynamic-paths

4.1.2. Dynamic Paths

The solution to the problem of hard-coding paths is to make use of built-in Python functions to work out the path of our `templates` directory automatically for us. This way, an absolute path can be obtained regardless of where you place your Django project's code on your filesystem. This in turn means that your project's code becomes more *portable*. At the expense of a little bit more complexity in setting up your Django project now, you can make your life much easier later on. No pain, no gain!

To start, modify the `settings.py` file to include a variable called `SETTINGS_DIR`. This will store the path to the directory in which your project's `settings.py` module will be contained. This is obtained by using the special Python `__file__` attribute, which is set to the absolute path of your settings module. We then take the absolute path to the settings file, extracting the path to the directory in which the settings module is contained. For example, calling `os.path.dirname()` on the absolute path of `<workspace>/tango_with_django_project/tango_with_django_project/settings.py` would yield a directory of `<workspace>/tango_with_django_project/tango_with_django_project/`.

RUN YOUR DJANGO WEB APP

1. Open a terminal window and locate your Project Folder (same folder with your application folder)
2. Once inside of your Project Folder, enter the command: `python manage.py runserver`
3. In your web browser copy and paste the following link: 127.0.0.1:8000/email
*"email" will be the name of your page/view

SET UP DATABASE SIDE OF WEB APP

Installing Dependencies

1. Install the latest-stable version of Python (https://www.python.org/downloads/)

1. Install Anaconda (https://www.continuum.io/downloads) (because you are using anaconda, you don't have to use virutalenv or virtualenvwrapper)
2. Install Django (https://docs.djangoproject.com/en/1.10/topics/install/#installing-official-release)
3. Install MySQL on Linux (http://dev.mysql.com/doc/refman/5.7/en/linux-installation.html)

Install MySQL on Mac (https://dev.mysql.com/doc/refman/5.6/en/osx-installation-pkg.html)

4. Install mysqlclient (https://github.com/PyMySQL/mysqlclient-python)

Creating Anaconda Environment

1. Create Conda virtual environment: create conda -n `(name_of_virtual_environment) python=(version) anaconda`
2. To activate conda virtual environment: `source activate (name_of_virtual_environment)`
3. To deactivate conda virtual environment: `source deactivate (name_of_virtual_environment)`

Launching Webserver via MAMP

1. (keep this connection on for the next part) Click on Open Webstart page
2. At the top tool bar click tools and then phpmyAdmin and there you will see the databases in phpAdmin.

You can alter database information as well as query using phpmyAdmin or use WYSIWYG like mysql workbench or sequel pro (for Mac)

Editing settings.py in Django project

1. (while in conda environment) Create a Django project: django-admin startproject projectname
2. Go to your project's settings.py file which would be located in: /projectname/projectname/settings.py
3. Edit the DATABASE section of the settings.py so that it looks like this:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': '', #input the name of your database
        'USER': '', #enter the username (in most cases, it will be root)
        'PASSWORD': '',   #enter the password (in most cases, it may be root as well)
        'HOST': '/Applications/MAMP/tmp/mysql/mysql.sock',   #if on linux system and using LAMP, exchange MAMP for LAMP
        'PORT': '', #port number of your mySQL database
        'OPTIONS': {
            'init_command': "SET sql_mode='STRICT_TRANS_TABLES'",
         },
     }
}
```

4. Save and exit settings.py and go back to the level of the directory that has `manage.py projectname`
5. Type in the following: python manage.py migrate in order for Django to connect to the database that you have created
6. Now type in the command line; `python manage.py runserver`, if it runs successfully, then your Django project is connected to a database and you can then create your `django application using django-admin startapp app_name`
7. If you have issues, make sure your `DATABASE` field is correct in syntax and what each field is assigned to. Furthermore, look at the above steps to see if you have reviewed them carefully