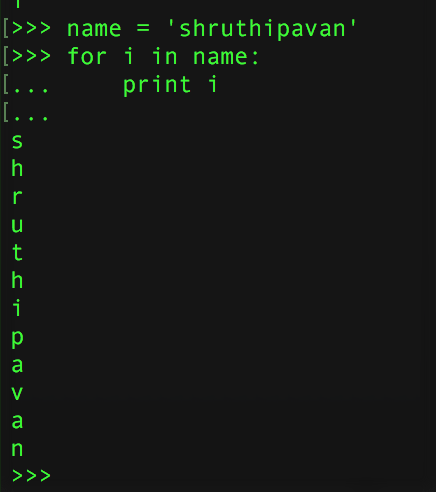
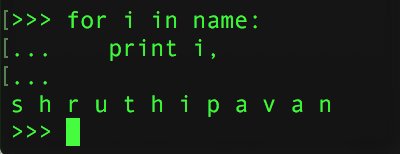
Loop:

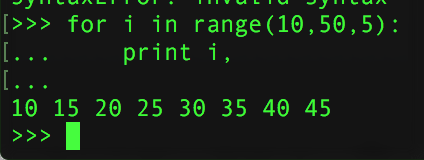
1. For LOOP: Loop can be done over Tuple, dictionary,String..etc
2. Sample Example:



1. If we provide [ , ] , then it prints with spaces in same line.

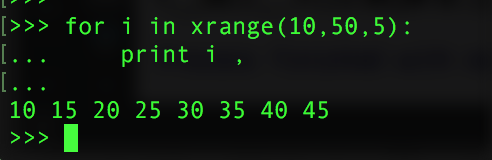


1. If we need series of numbers , then we can use this way.



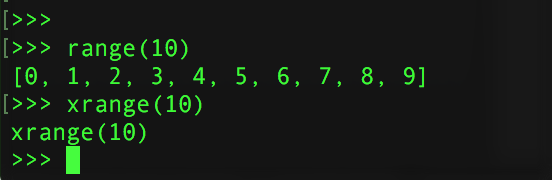
In this example, all the numbers loads into our memory, so here we can use GENERATOR instead of ITERATOR Object(RANGE)

1. Example to use Generator Object.



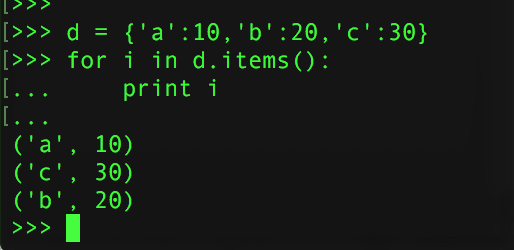
xrange is ‘Generator’ Object. this will not load the values into memory, we can retrieve on need basis

1. The difference between generator and iterator object is….



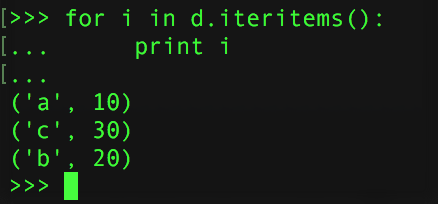
In above example, we can see that range(10) loaded the result and printed it onto python prompt. But in case of xrange(10) it did not loaded into our memory. It will load somewhere else and gives us the result as and when we need it.

1. We can loop over ‘Dictionary’ Objects :



Here items() is an ITERATOR Object. We can try like below:

1. Iteritems() is generator Object, so that values won’t be stored in our memory

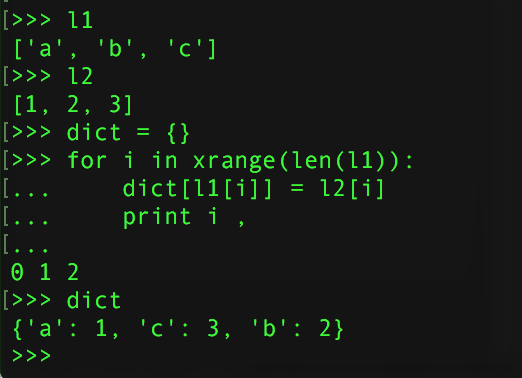


|  |  |
| --- | --- |
| Iterator Objects | Generator Objects |
| range(n) | **xrange(n)** |
| DictionaryObj.items() | **DictionaryObj.iteritems()** |
|  |  |
|  |  |

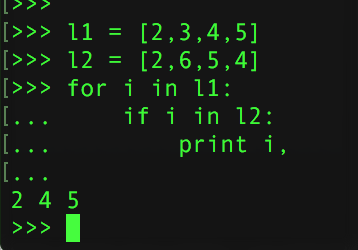
1. PROBLEM :

🡪 If we have

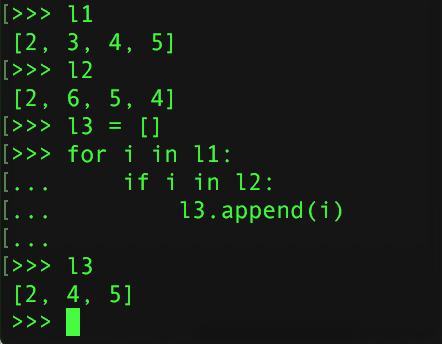
 , then how to make L1 index char as key, and L2 equivalent as value.



1. If we want to find the common elements from 2 Lists



1. If we want to create a new list of common elements.



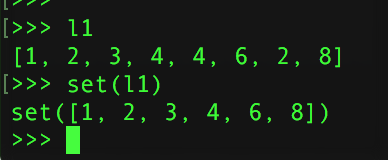
1. We can remove duplicates : We can use sets to do.

**SET :**

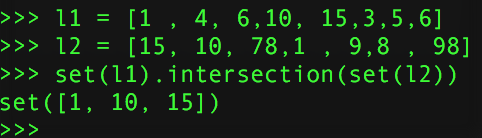
🡪 It removes duplicates.

🡪 We can convert Set to LIST or vice-versa. Typecasting is very easy using these.

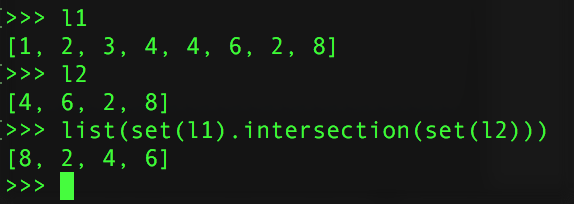
🡪 Ex:



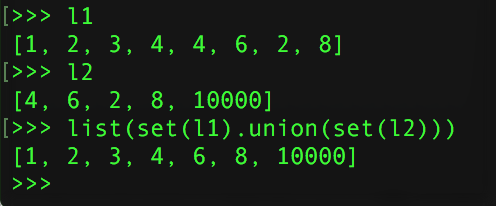
🡪 We can get common elements by using intersection



🡪 Typecast is also easy:

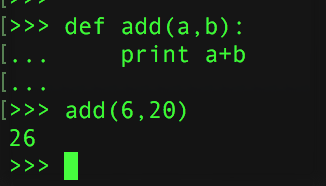


🡪 Union Function to club 2 Lists

****

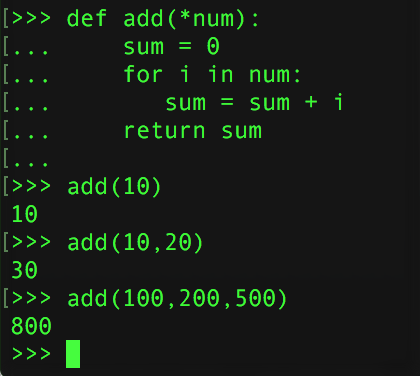
**FUNCTIONS:**

🡪For reusability

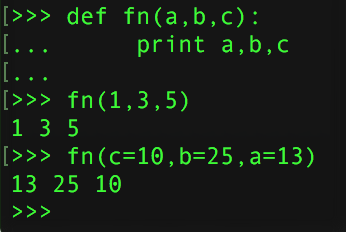


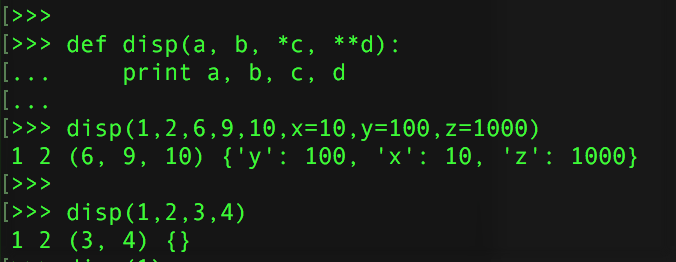
🡪 Overloading is not possible, only method overrides.

🡪 We can have dynamic number of args using ‘\*’

EX: 

🡪 If we want to pass args in different order while calling method:

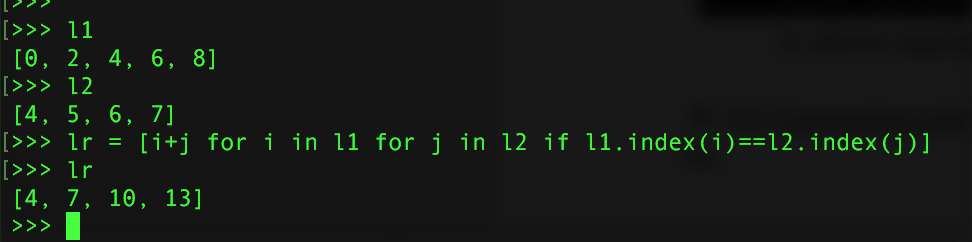


🡪 Named Args can be passed by using “\*\*” 

In above example , \*\*d is for named Arguments.

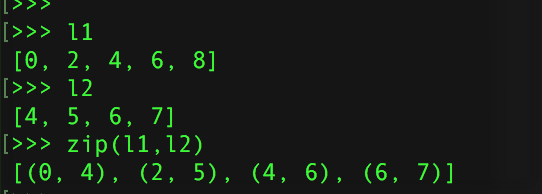
🡪 ***List Comprehension :***

Ex:

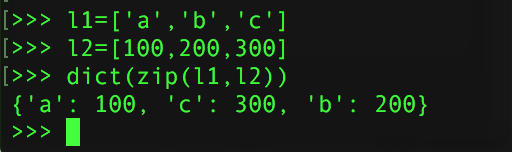


🡪 ***ZIP :***

Zips same index of 2 lists elements together.

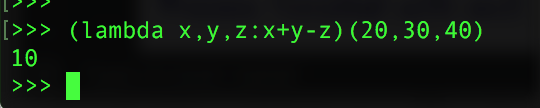


We can convert zip objects to dictionary Object.

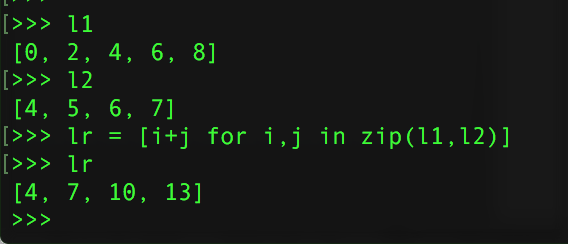
****

🡪 ***Lambda functions :***

Lambda functions are… 1 line run-time functions



We can add 2 lists which are at same index.



**: DJANGO :**

🡪 It’s Python Framework Used to develop Web applications and Web-Services.

🡪 MVT model it follow

M – Model

V - View

T – Template

🡪 By default, Django has ORM

**Creating 1 Web application using Django**

For this we need to

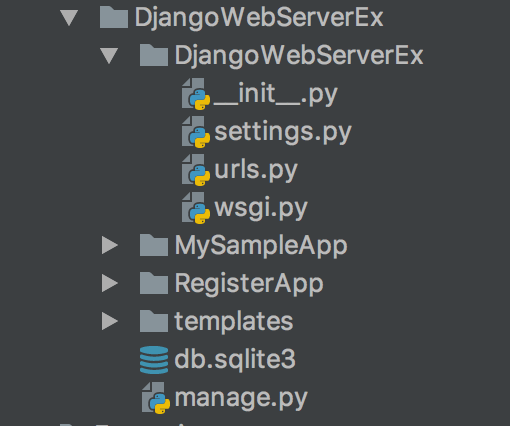
* 1. Install Django
  2. Create 1 Application and make it as Web / UI available.
  3. Create urls and
  4. Create another app to make ORM available.

***PROCEDURE***:

1. pip install Django
2. To create a project ,

Django-admin.py startproject <Project\_Name>

As soon as we create this project , we get following structure

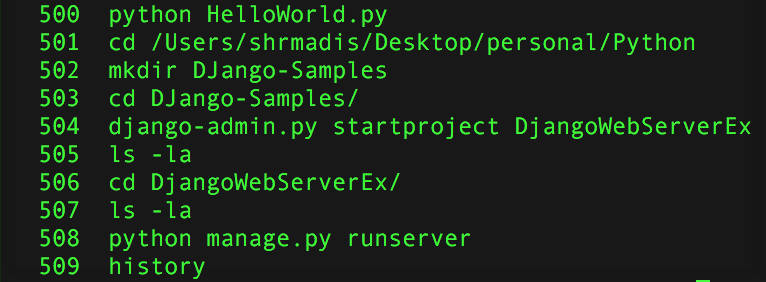
****

|  |  |  |
| --- | --- | --- |
| Folder Name | Meaning | Default files and folders |
| DjangoWebServerEx | **Project Name** | 1. ***db.sqlite3*** 2. ***manage.py*** 3. **DjangoWebServerEx – This is a default App gets created automatically when we create a project** |
| DjangoWebServerEx | **Default Application Name** | 1. **\_\_init\_\_.py** 2. **settings.py – contains the info about INSTALLED\_APPS, TEMPLATES, DATABASES..etc** |
|  |  |  |
|  |  |  |

1. Run the server ,

***runserver*** runs the localhost Server.

Commands for reference:

****

* we can get result in browser, [http://localhost:8000](http://localhost:8000/)
* 8000 is default port, we can change the port, by giving this command

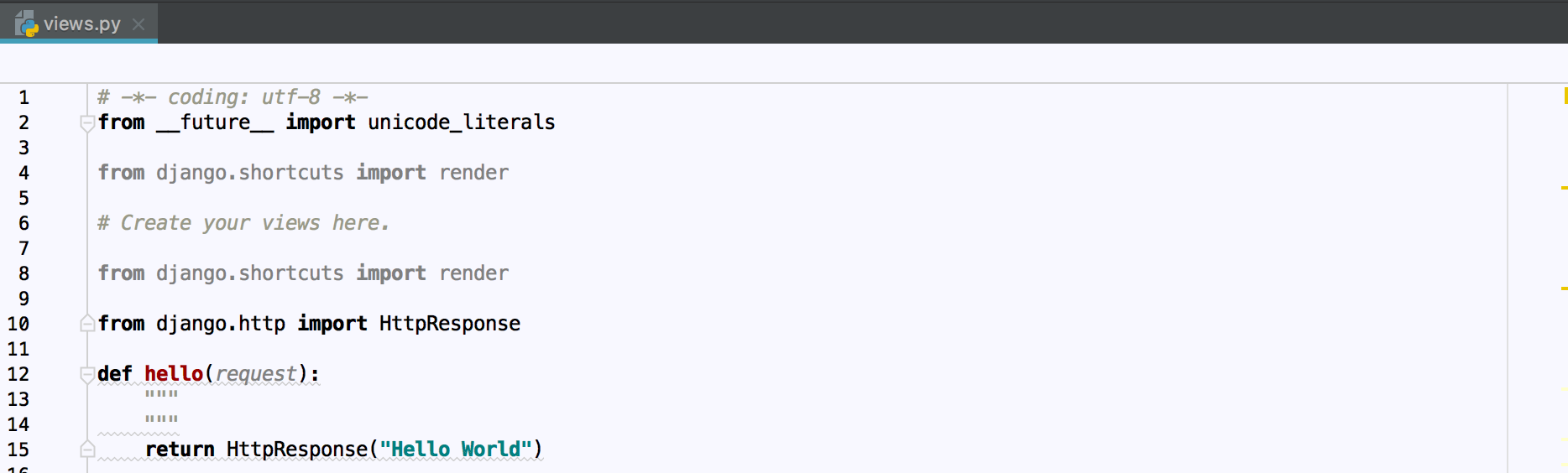
python manage.py runserver 8001

* manage.py is for runserver, startproject, startapp…etc operations

1. To create app ,

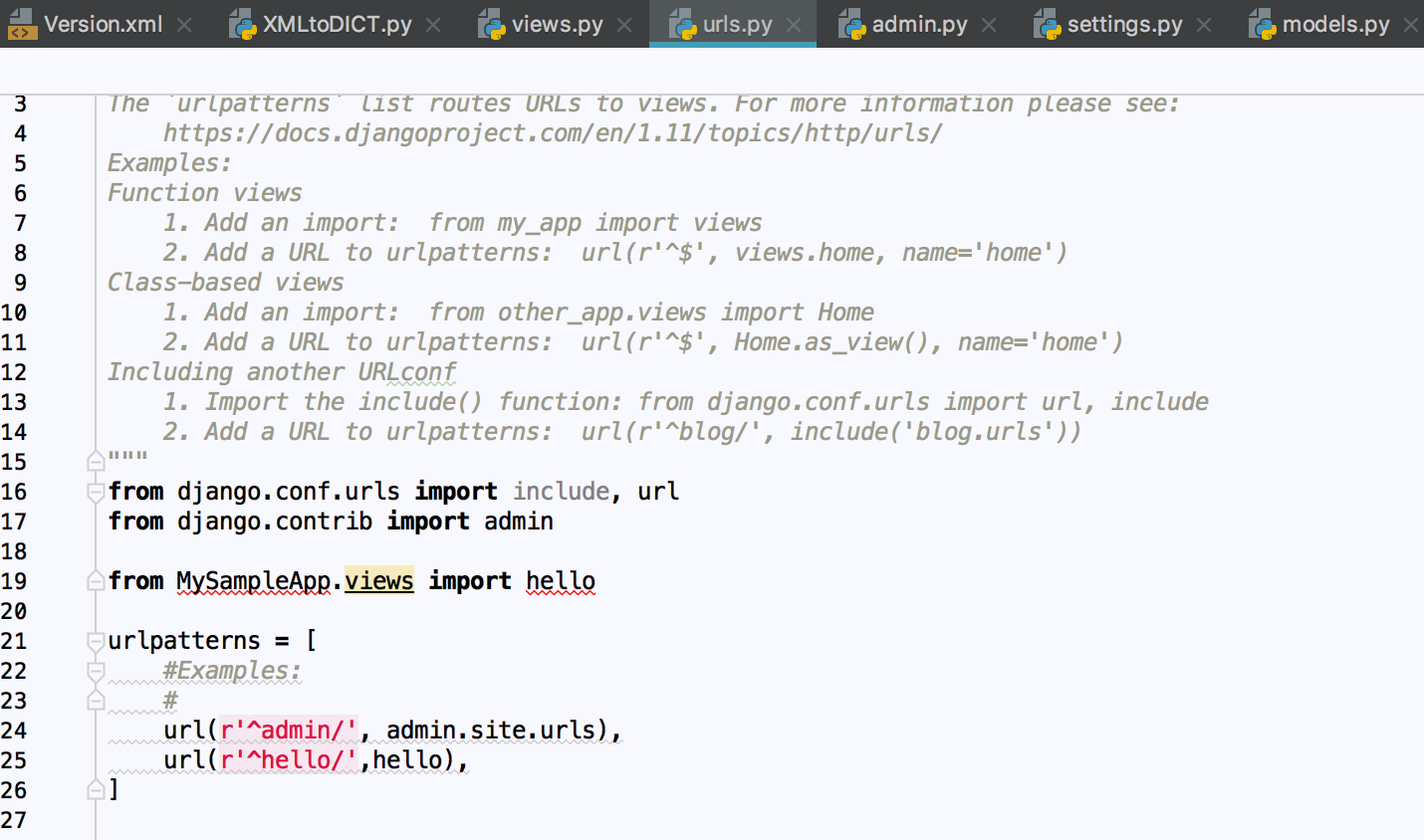
* Python manage.py startapp MySampleApp
* This creates one App named “MySampleApp”, and models.py & views.py..etc gets created.

1. To create a service,

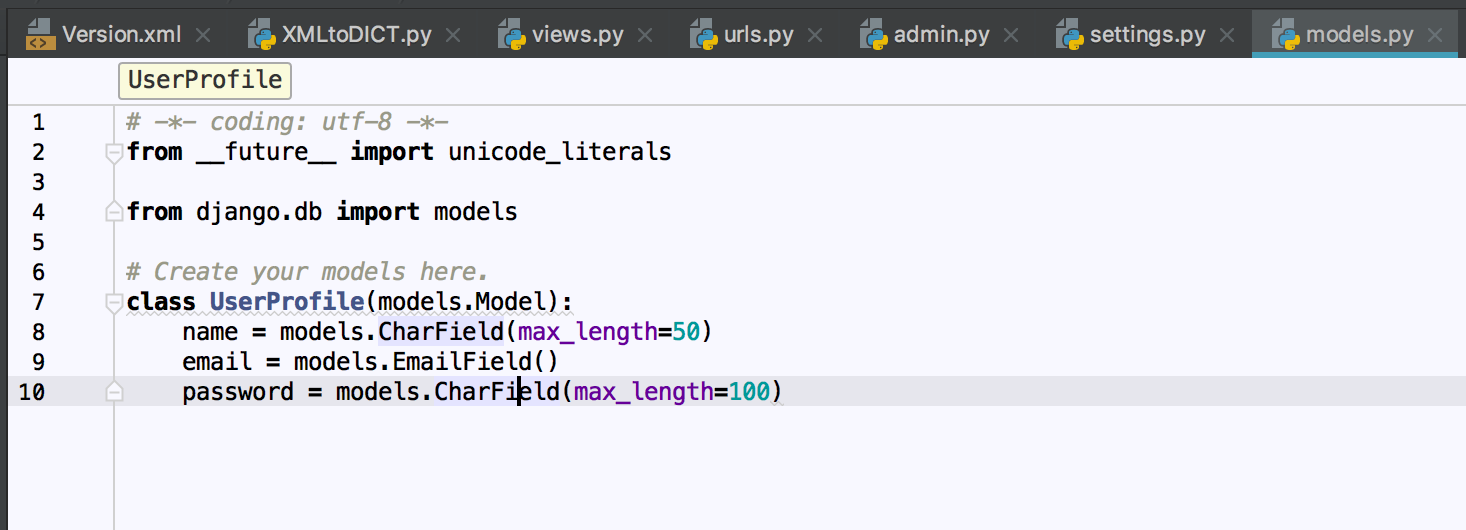
* edit views.py like below : (inside “MySampleApp” App)

1. make the url available for the method,

* Now hello() implementation is done, we need to have a url matching for this method. Then when we enter that URI from browser, this method gets executed.
* Edit the urls.py (inside DjangoWebServerEx project 🡪 DjangoWebServerEx App which isdefault App)



1. Create another app ,

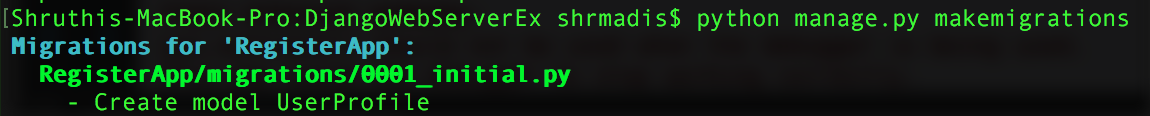
* This 2nd App(RegisterApp) acts as ORM Manager for the 1st App(MySampleApp) to make it as ORM available.
* python manage.py startapp RegisterApp
* And change the models.py () – (For 2nd App which is RegisterApp)
* Here, models.py acts as a ORM class,
* ‘name’ , ‘email’ , ’password’ are DB fields.

1. Make this new App as an Installed APP,

* change the settings.py in Default App(-- inside DjangoWebServerEx project 🡪 DjangoWebServerEx App which is default App)
* NOTE: If we change Models.py file , then we need to mention our App inside ‘INSTALLED\_APPS’ Dictionary to make this app as an installed app.
* Usually default-apps will be installed and detected automaticall
* In above screen shot, we have added **RegsterApp**

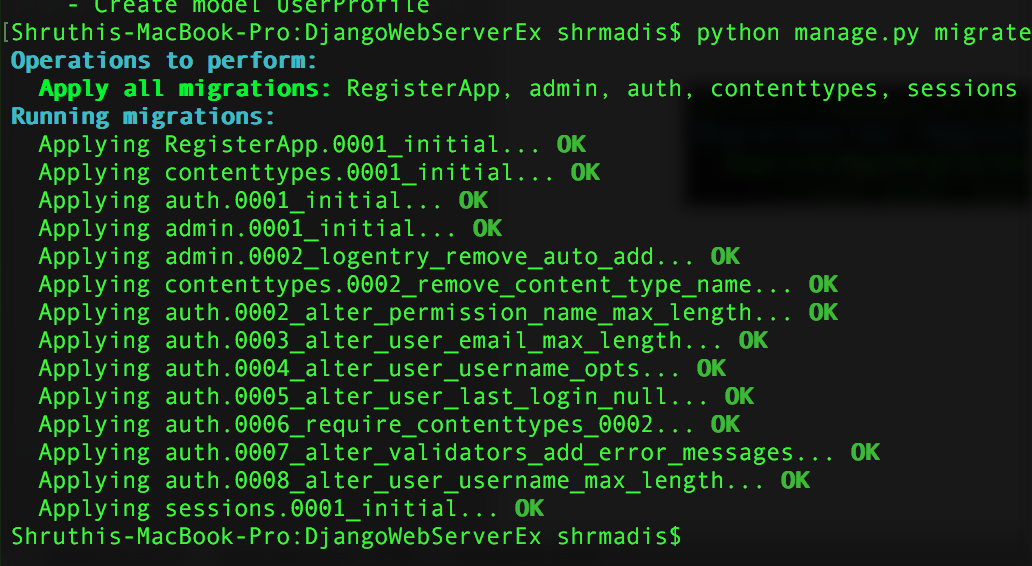
1. Make our models as ORM Mange scripts or classes,

* Here our Model(UserProfile) needs to convert as an ORM classes by using this command
* python manage.py makemigrations



1. Make ORM to DB Tables

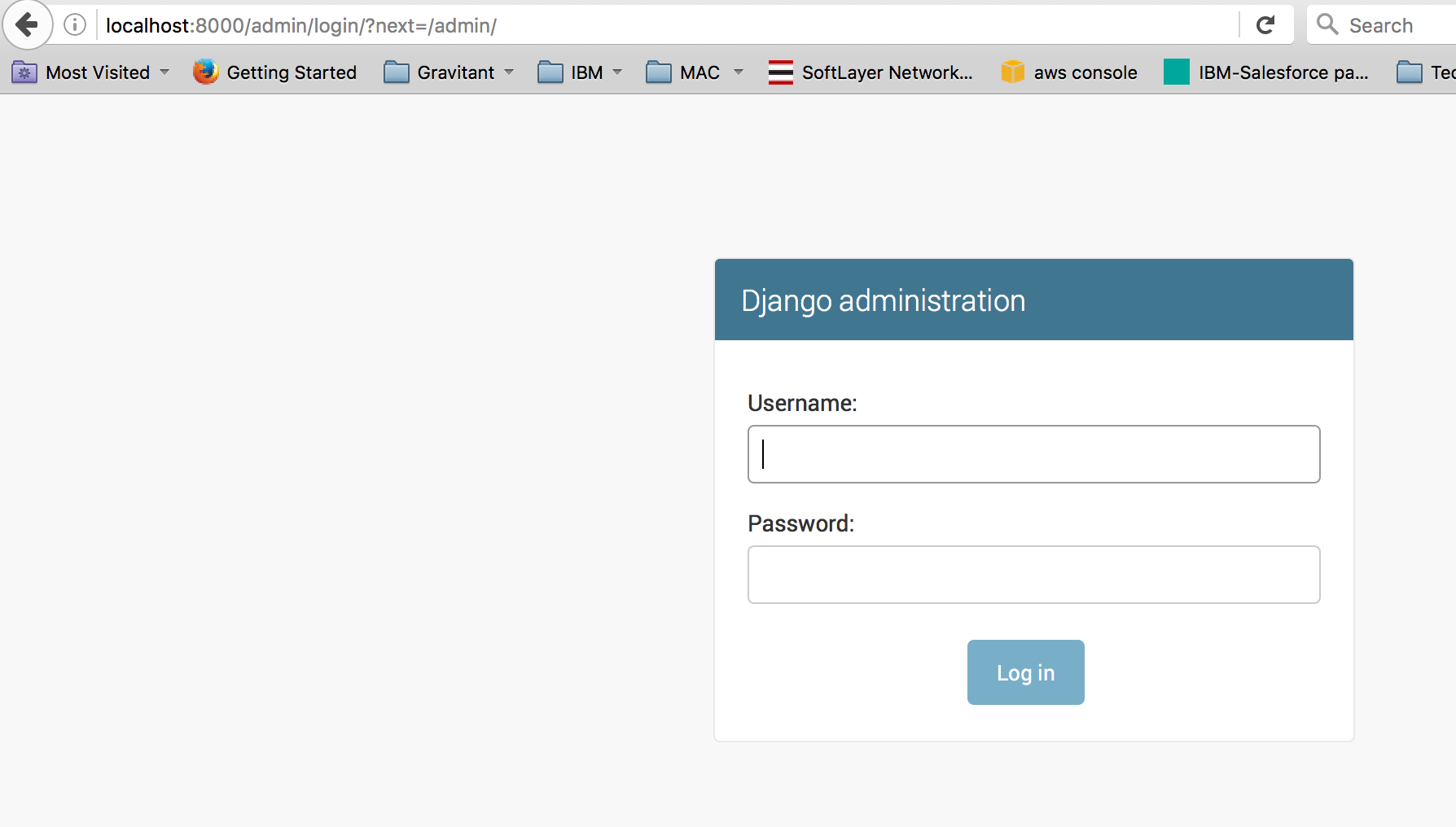
* Once the ORM is created, make DB Tables and DB setup from these generated ORM models by using this command.
* *python manage.py migrate*



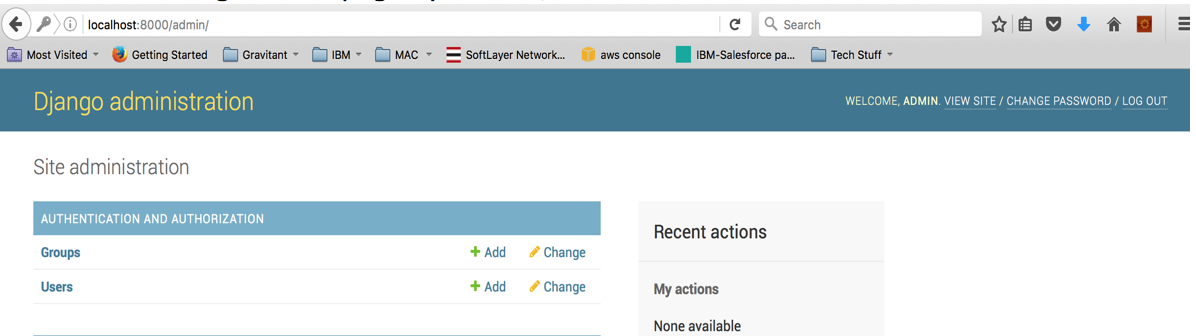
1. Now create a Super User for this UserProfile ,

* *python manage.py createsuperuser*
* It prompts for name , email and password causeour UserProfile has these fields.

1. Start the server, by running this command

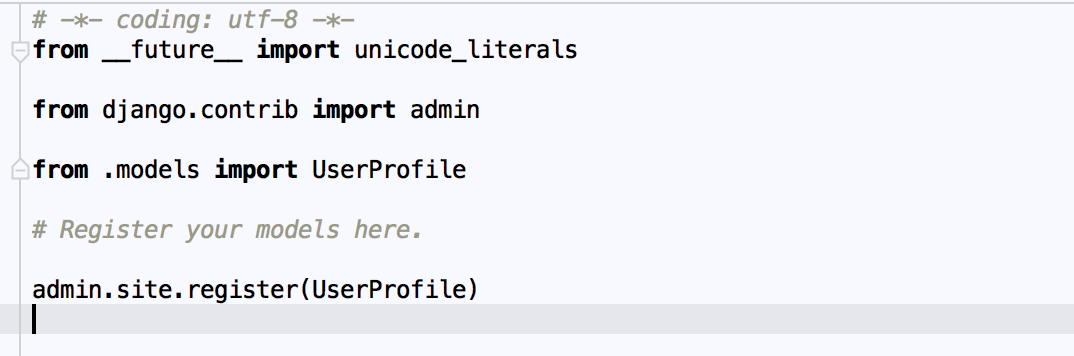
* python manage.py runserver
* Then goto <http://localhost:8000/hello>
* 🡪 Gives us “Hello World response” This intern calls “hello” Method,
* Then goto <http://localhost:8000/admin> 🡪 ****
* Login with same creds whatever You used to create SuperUser in above step.

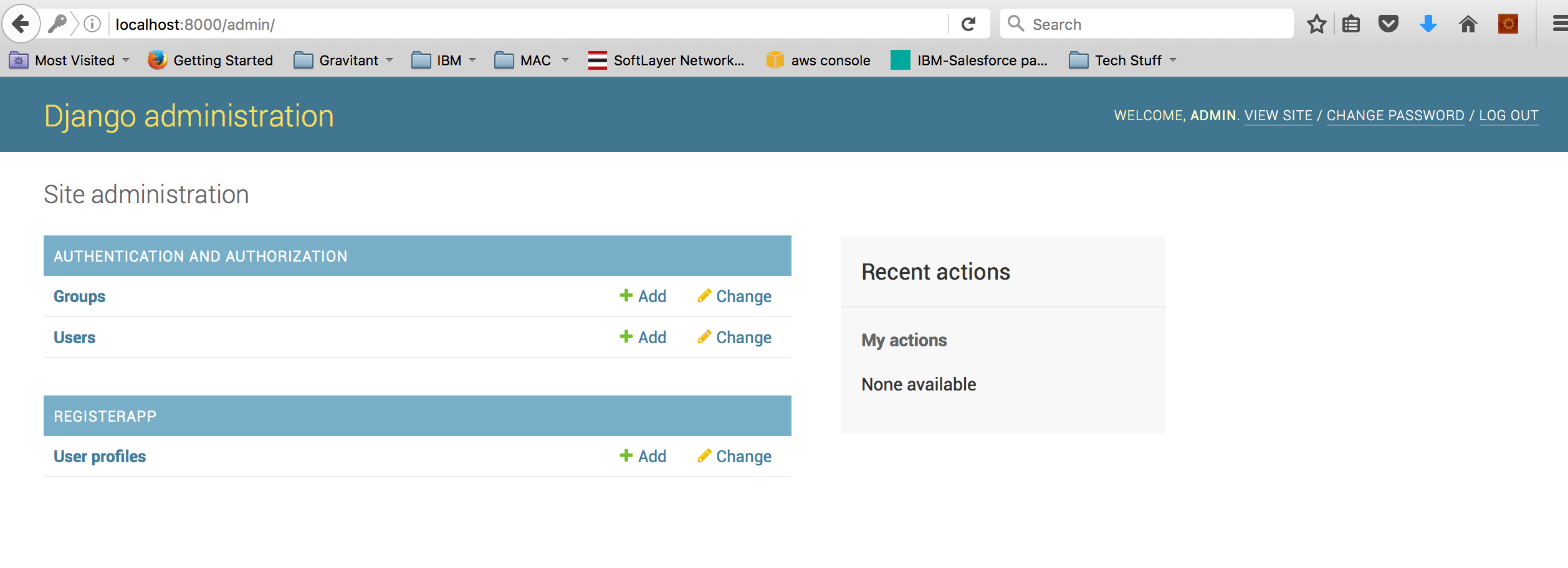
It gives this page by default,



1. Attach UserProfile Object to admin

* In above screen-shot , by default we get Groups and Users.
* Same way we can give UserProfile also, so that you can add/edit/delete.
* Now, to make our UserProfile object attached to this admin page
* Add code snippet in admins.py of 2nd app(where we have ORM models -- RegisterApp)



* Login again back to <http://localhost:8000/admin> where you can see UserProfile also attached as below:

**DAY 3**

**~~~~~**

* **Pip install pandas – Used for DataAnalysis**
* **Pip install matplotlib – is Used for generating the Graphs or Charts**
* **Pip install iGraph – Similar to MatplotLib , but matplotlib is more powerful than iGraph.**
* **Pip install pylint – To scan and tell the code strength of our python files.**
  + **Pylint <something.py>**

**Dataframe is generator Object , which loads data somewhere else and gives the result whenever needed.**

**HUE:**

This is Python Data Analytics Application, used (Python + Django)

<http://demo.gethue.com/> (demo/demo)

<https://github.com/cloudera/hue> (github project for Hue)

<http://demo.gethue.com/home?uuid=6d96c7fc-b7d2-4ee3-8d42-dd55cc4dcc09>

demo / demo are CREDs.

PEP 8 : <https://www.python.org/dev/peps/pep-0008/>

This is used follow guidelines for Python programming

AUTOPEP 8 : It Auto Corrects trailing white spaces, and some small errors in our py files.

HOOKS : We can add hooks to our code source so that it checks these standards on our PY files. 2 Types of hooks are available to attach

* Pylint
* Pyflakes

Any one of the above we can hook and make the standards.

Logging:

LEVELs :

INFO

DEBUG

WARNING

ERROR

CRITICAL

We can set the level in 2 ways

* basicConfig

**import** logging  
logging.basicConfig(level**=**logging.DEBUG)

* setLevel

**import** logging  
  
logger **=** logging.getLogger(\_\_name\_\_)  
logger.setLevel(logging.INFO)

Logger configuration we can set in (\*.ini) file

\_\_name\_\_ : Gives you in which mode your logger is executing

\_\_module\_\_ :

\_\_doc\_\_ : Gives doc Strings

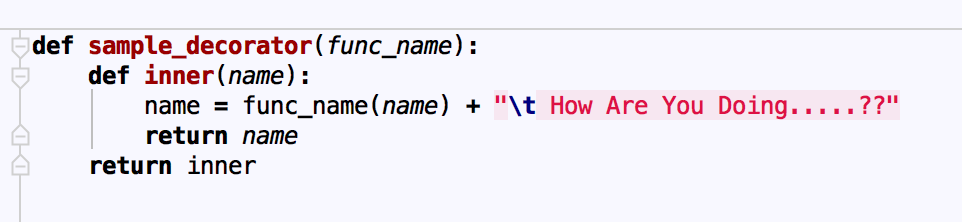
DECORATORS

🡪 are used like annotations

🡪 Piece of code where we need in common

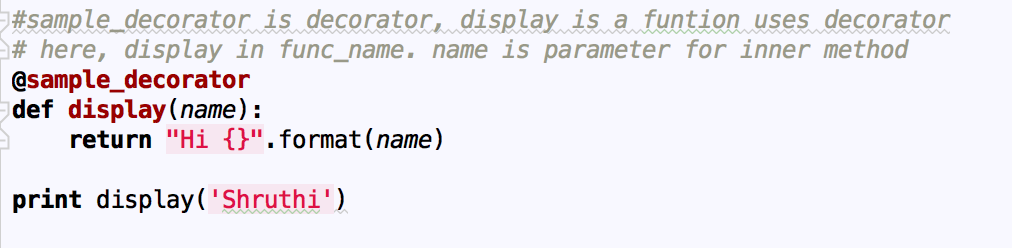
🡪 We should put @ and followed by decorator name

🡪 We can create a decorator like below:



In above, sample\_decorator is a decorator implementation method, where it will take the function name as main parameter, and function params as params in inner method:

🡪 Decorator usage method will be like:



**Output will be :**

Hi Shruthi How Are You Doing…..??

The Order of Execution is

* 1. First the display() method is called with name = ‘shruthi’
  2. Then display gets executed and returns the result(Hi Shruthi) to sample\_decorator as an input
  3. In “Sample\_decorator”, it takes display() <method-name> as an input first, then result of display() <Hi Shruthi> as an input to inner() method.
     1. Note: It can be inner() or wrapper() or any other name:
  4. The logic inside inner() method gets executed onto above.
  5. So the final Output would be:
     1. ***Hi Shruthi How Are you Doing…??***

***We can also have multi-level decorators by adding more ‘@<decorator\_name>’ on top of function.***

***Ex:***

******

***The order of Execution is:***

* + - 1. ***Display()***
      2. ***Result of display() as an input to sample\_decorator()***
      3. ***Result of sample\_decorator() as an input to sample\_decorator2()***

***🡪 pip install pywinauto***

* ***Is used for windows application Automation, we can install open and use some options from scripting***
* ***We can install any apps(notepad.exe, explorer..exe……etc) and perform automation on the options provided by them***

***Pip install ldtp – Is equivalanet tool for mac.***

***EXCEPTIONS:***

***🡪same like java***

***🡪 try , except, else, finally are keywords***

***🡪 except <Exception-Class-name>: followed by code.***

***🡪 If we need to handle all the exceptions in same way(all exceptions having same logic)***

***then we can write:***

***except <exception-class1>,<exception-class2>,…etc:***

🡪 Connect to DB

We can Connect to DB , we need Database Adapters(da)

🡪 Mysql DB, 🡪 da will be MYSQLdb

🡪 find the DatabaseAdapter for any DB on internet and install

🡪 pip install MYSQLdb

🡪 Then we can import and we will have a connect method to create db connection

🡪 mdb.connect(‘host’,’username’,’password’,’databasename’)

🡪 We can perform execute() , save(), delete(), fetch()…etc

***DAY 4***

***~~~~~***

***🡪 install djangorestframework***

***pip install***

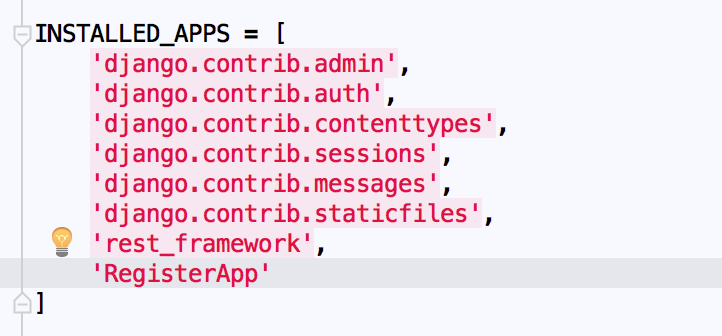
***this will create a “rest\_framework” APP inside site-package***

***Create a Project (Django-admin startproject <project\_name>)***

***Cd <project\_name>***

***Create an app inside the Project(python manage.py <app-name>)***

***Now goto Settings.py***

******

***Add Rest Framework Dictionary to get all the in-built Classes.***

******

***🡪 Create a Rest-Project , and Register-App and do like Day2 steps.***

***🡪 Create a py file, to write UI validations. It is used for***

***🡪 python manage.py shell , prompts you to Django Shell***

***🡪 There we can import Models here***

***🡪 pip install pyresttest***

***Or***

***🡪 pip install pyresttest –user***

***Or***

***🡪 Or get the package from internet and put it in site-package folder and run below command:***

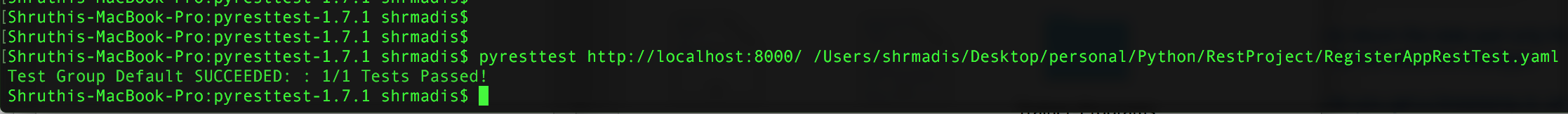
***python setup.py install***

***🡪 RestTest is used for testing the rest implementation methods.***

***🡪 We need to write Yaml files to test these.***

***🡪 and execute it using below command:***

***pyresttest <server-url> <yaml-path-loc>***

******

***🡪 Paramiko is used to connect from 1 machine to another (ssh,scp and RDP) – if you use stand-alone machines.***

***Pip install paramiko***

***🡪 Boto is used for cloud systems to talk from 1 machine to another. (If you use AWS/Openstack…etc)***

***pip install boto***

*# Simple tests to verify things work against a live REST service that returns JSON.. Github is a natural one***---  
-** config:  
 **-** testset: "Test my Rest Web Services"  
**-** test:  
 **-** name: "Basic test for Create and Get Users for Register App"  
 **-** headers: **{**accept: 'application/json'**}  
 -** url: "/getusers/"  
 **-** validators: *# operator is applied as: <actual> <operator> <expected>* **-** compare: **{**header: "content-type"**,** comparator: contains**,** expected: 'application/json'**}  
 -** compare: **{**jsonpath\_mini: "x"**,** comparator: "eq"**,** expected: "abc"**}** *# - compare: {raw\_body: "", comparator: "regex", expected: '.\*'}* **-** extract\_test: **{**jsonpath\_mini: "does\_not\_exist"**,** test: "not\_exists"**}**

***Machine Learning:***

***Python tools are:***

***Pandas(csv data),***

***opencv(analyzing videos data),***

***scipy(Data Streaming for audio or video),***

***sklearn(To learn data== similar to opencv),***

***xlrd(Read Excel tool),***

***xlwt(Write Excel tools)***

***🡪 Loading Data***

***🡪 analyze data***

***🡪***

***🡪***

***Parse by value / Parse byrefence does not applicable for Python.***

***Since python is Object oriented, it parses by Object***

***🡪 xmltodict – is Dom Parser***

***🡪 celementary – is SAX parser***

***🡪 Copying mutable object and immutable object is different***

***a = 1***

***b = a***

***a = 2***

***then result will be a =2 , b = 1***

***since these are immutable objects.***

***🡪***

***🡪 l1 = [1,2,3,4,5]***

***🡪 l2 = l1***

***🡪 l1.append(6)***

***result will be l1 = [1,2,3,4,5,6] and l2 = = [1,2,3,4,5,6]***

***since we copied the address***

***For this we should slice that data and use***

***🡪 L2 = l1[::] ---- this copies only the content***

***🡪 Now. If we append the content in l1.append(6)***

***🡪 The result will be l1 = [1,2,3,4,5,6] and l2 = [1,2,3,4,5] – Cause we copied the content.***

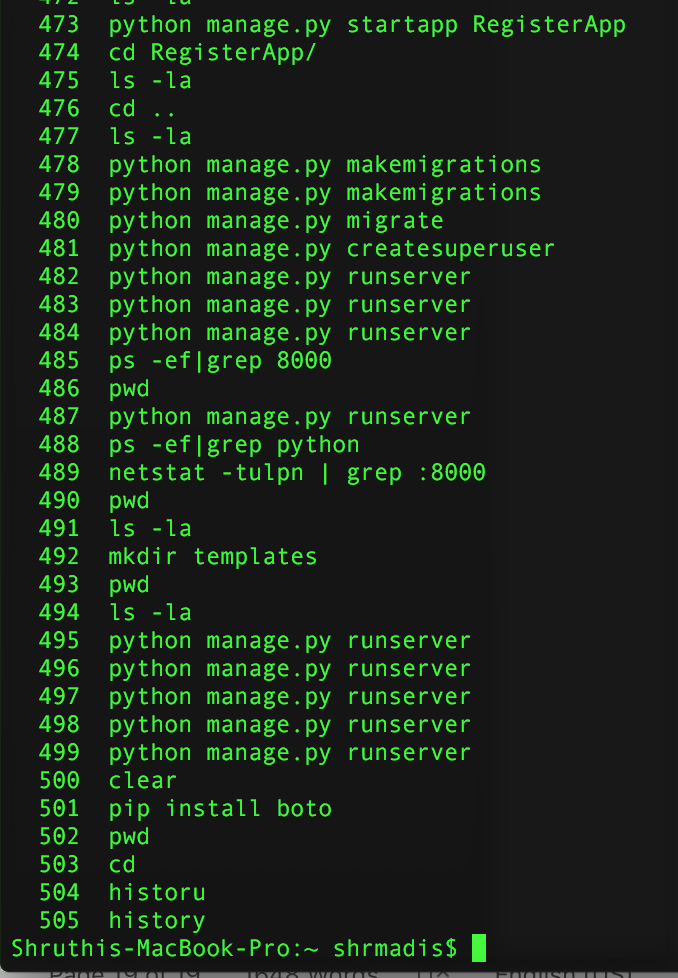
***🡪 If we want to add the elements into the list in first index,***

***we should create a class and extend the list class, then override the append method to get the appended elements first.***

***Class ListEx(list) :***

***Def append(list, append\_element):***

***Change the logic here.***

******

***PDB Options:***

***n 🡪 execute***

***s 🡪 go inside the function***

***r 🡪 Return back to above method***

***l 🡪 list where the cursor is.***

***c 🡪 continue execution.***

***a 🡪 see Properties***