Ex 2 **Running Python**

**AIM:**

To set up python and use python to create a server side dynamic web page. Also, to

1. Monitor changes in the stock market using live updates from http://money.rediff.com
2. Using the obtained information from the site through Beautiful Soup and dynamically update the results
3. Provide storage of values in database in regular intervals.

**INSTALLATION PROCEDURE:**

**Steps to install Python and Django**

Step 1: First sudo -i

Step 2: Then run

apt-get update

apt-get upgrade

Step 3: Next we shall install python, so run the following command:

apt-get install python

Step 4: Now, we install Django for Python, a web framework. Type

apt-get install python\_django

Step 5: Then type **python**  into the terminal and execute the following lines

import django

print(django.get\_version())

The above lines will print the currently installed version of django indicating django has been installed successfully.

Step 6: Also, install Beautiful Soup 4 using pip

pip install bs4

**Creating a Django Project**

Step 1: To create a new project in Django,

django-admin.py startproject mysite

This command will create a project directory **mysite** with certain auto-generated python script files.

mysite /

manage.py

mysite /

settings.py

urls.py

wsgi.py

Step 2: In the settings.py file, make necessary changes to connect to the database, as   
 below

ENGINE – “django.db.backends.mysql”

NAME – “python\_db”

USER – “root”

PASSWORD – “ssn”

Step 3: Then execute the following command

python manage.py syncdb

The above command will create necessary databases for the application based on the models created in models.py

Step 4: To start the django server,

./manage.py runserver

Then, the python code can be executed through the browser in an approporiate port in localhost.

The urls.py maintains the mapping between the url sequence(exhibited as a regular expression) and the corresponding python method to be invoked

**APPLICATION – Algorithm:**

**File: models.py**

1. Create a class s\_info in order to store the concerned details about stock prices – datetime, market, price, daily change and percent change.
2. An important feature to be noted is that the syncdb command described above makes use of this model class to create a table called s\_info with the fields named as above, making the pushing and retrieval of data much easier.

**File: views.py**

1. Define methods to update the database (updateDB) and context generation (generateContext) where the former pushes the data retrieved into the database created above and the latter browses through the html contents of the retrieved page and finds the necessary data.
2. Define a method index to hold the welcome action of the application – using in-built method render of the package django.shortcuts, route it to stock/index.html page which displays a welcome screen and a navigation button to the retrieve details page.
3. Define a method stocknews to handle the fetching and processing of stock details from http://money.rediff.com and to process it using the generateContext and updateDB methods. The output is then redirected to a stock/stock-news.html where it is displayed in a neat tabular form.

**File: urls.py**

1. Add new patterns to the urlpatterns variable –
   1. A default regular expression to view the welcome page: calling the index routine and
   2. An app/stock-news regular expression to view the current stock details : calling the stocknews routine

**PROGRAM CODING:**

**File Name: index.html**

<html>

<head>

<title>

Market Analysis | Client

</title>

{% load staticfiles %}

<link href="{% static 'stock/css/main\_page.css' %}" type="text/css"   
 rel="stylesheet" />

</style>

</head>

<body>

<div id="content" align="center">

<a href="app/stock-news">

Click here for latest stock updates

</a>

</div>

</body>

</html>

**File Name: stock-news.html**

<html>

<head>

<script>

var c = window.setInterval(function(){document.location.reload(true);},5000);

</script>

{% load staticfiles %}

<link href="{% static 'stock/css/main\_page.css' %}" type="text/css" rel="stylesheet" />

</head>

<body>

<div id="content">

<table rules="all">

<tr>

<th>Market</th><th>Current Value</th><th>Daily Change</th><th>% Change</th>

</tr>

<tr>

<td>BSE</td>

<td>{{ bse\_index\_value }}</td>

<td>{{ bse\_points\_change }}</td>

<td>{{ bse\_change\_percentage }}</td>

</tr>

<tr>

<td>NSE</td>

<td>{{ nse\_index\_value }}</td>

<td>{{ nse\_points\_change }}</td>

<td>{{ nse\_change\_percentage }}</td>

</tr>

</table>

</div>

</body>

</html>

**File Name: models.py**

from django.db import models

# Create your models here.

class s\_info(models.Model):

s\_datetime = models.CharField(max\_length=40)

s\_market = models.CharField(max\_length=20)

s\_price = models.CharField(max\_length=20)

s\_daily\_change = models.CharField(max\_length=10)

s\_percent\_change = models.CharField(max\_length=10)

def \_\_unicode\_\_(self):

return self.s\_price

**File Name: urls.py**

from django.conf.urls import patterns, url

from stock import views

urlpatterns = patterns('',

url(r'^$', views.index, name='index'),

url(r'^app/stock-news',views.stocknews, name='stock-news'),

)

**File Name: views.py**

# Create your views here.

from django.shortcuts import render

from django.template import Context

from django.http import HttpResponse

from stock.models import s\_info

from bs4 import BeautifulSoup

import requests, datetime

def index(request):

return render(request,'stock/index.html')

def updateDB(one,two,three,four,five,six):

stock = s\_info(s\_datetime = str(datetime.datetime.now()),

s\_market = 'BSE',

s\_price = one,

s\_daily\_change = two,

s\_percent\_change = three,

)

stock.save()

stock = s\_info(s\_datetime = str(datetime.datetime.now()),

s\_market = 'NSE',

s\_price = four,

s\_daily\_change = five,

s\_percent\_change = six,

)

stock.save()

def generateContext(soup):

bse = soup.find(id="sensTab1")

bse\_span = bse.contents[-2]

nse = soup.find(id="sensTab2")

nse\_span = nse.contents[-2]

updateDB(str(bse.div.span.contents[0]),str(bse\_span.contents[0]),str(bse.contents[-1]),

str(nse.div.span.contents[0]),str(nse\_span.contents[0]),str(nse.contents[-1]))

stock\_context = Context({ 'bse\_index\_value': bse.div.span.contents[0],

'bse\_points\_change': bse\_span.contents[0],

'bse\_change\_percentage': bse.contents[-1],

'nse\_index\_value': nse.div.span.contents[0],

'nse\_points\_change': nse\_span.contents[0],

'nse\_change\_percentage': nse.contents[-1]

})

return stock\_context

def stocknews(request):

var = requests.get("http://money.rediff.com/")

soup = BeautifulSoup(var.text)

stock\_context = generateContext(soup)

return render(request, 'stock/stock-news.html', stock\_context)

**Modified part in settings.py**

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.mysql', # Add 'postgresql\_psycopg2', 'mysql', 'sqlite3' or 'oracle'.

'NAME': 'Python\_db', # Or path to database file if using sqlite3.

# The following settings are not used with sqlite3:

'USER': 'root',

'PASSWORD': '',

'HOST': '', # Empty for localhost through domain sockets or '127.0.0.1' for localhost through TCP.

'PORT': '', # Set to empty string for default.

}

}

**File Name: mysite/urls.py**

from django.conf.urls import patterns, include, url

# Uncomment the next two lines to enable the admin:

# from django.contrib import admin

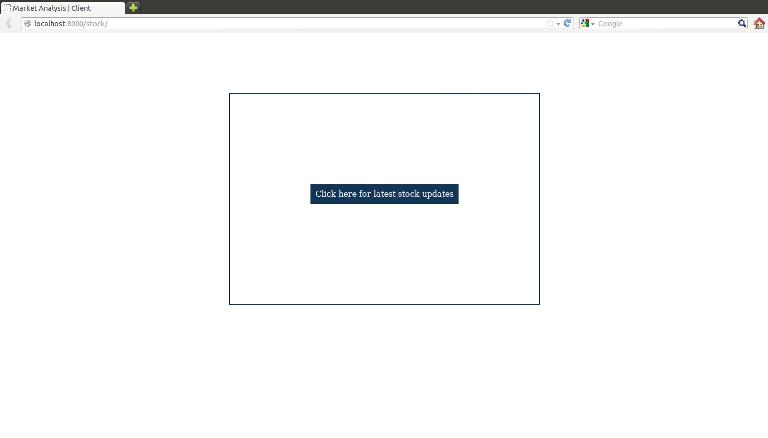
# admin.autodiscover()

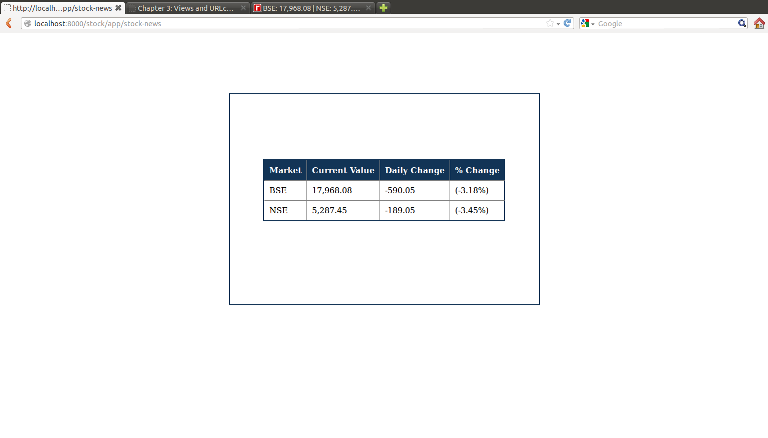
urlpatterns = patterns('',

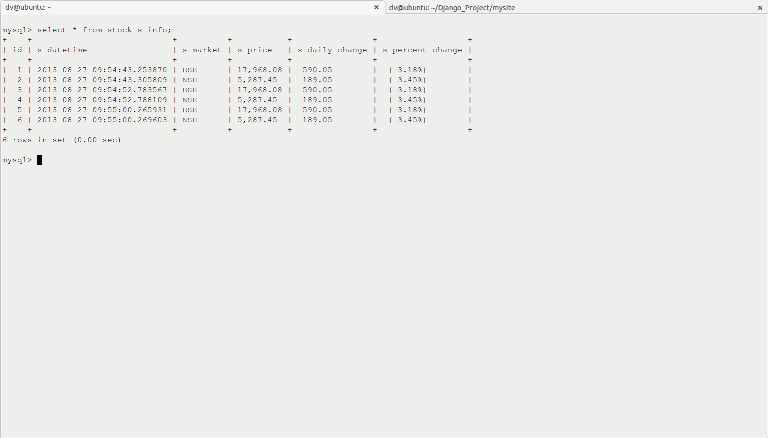
url(r'^stock/', include('stock.urls')),

)

**Output Screenshots:**







**RESULT**

Python and Django were installed in Ubuntu Desktop Environment. Python code was written to retrieve data from http://money.rediff.com and provide regular updates on BSE and NSE stock values. Text processing was done using the Beautiful Soup 4 package. The data was also updated into a relational schema in MySQL database using the models concept in Django.