Documentation of Email Sending Using Python

Please Refer deep Expiations for SMTP server

https://realpython.com/python-send-email/

Python comes with the built-in smtplib module for sending emails using the Simple Mail Transfer Protocol (SMTP). smtplib uses the RFC 821 protocol for SMTP. The examples in this tutorial will use the Gmail SMTP server to send emails, but the same principles apply to other email services.

The code example below creates a secure connection with Gmail's SMTP server, using the SMTP_SSL() of smtplib to initiate a TLS-encrypted connection.

Ports 465 and 587 are intended for email client to email server communication - sending out email using SMTP protocol. SSL Secure Sockets Layer (SSL) for SMTP connections encryption is started automatically before any SMTP level communication. It is almost like standard SMTP port.

Objective: Send the emails by smtp server using python programing

Summary:

1. Import required library for sending mail.
2. Create the connection with SMTP server for sending email by using port, smtp server address and sender email address.
3. Then After create the mail body for sending mail.
4. Read the excel file of email receiver
5. Upload the HTML format for attachment of email
6. Connect with smtp server via sender email and password
7. Then send the email to all email address which present in the excel file
   And applied email validation using regular expression for finding invalid email address.

#Algorithms for Email Sending:

#import Simple Mail Transfer Protocol library

```
import smtplib, ssl
```

# MIMEText for creating the text in email

```
from email.mime.text import MIMEText
```

```
# for multipart form data
from email.mime.multipart import MIMEMultipart
# for dataframe operations
import pandas as pd
# convert email to bytes
from email import encoders
# the MIMEText class is used to create email object text.
from email.mime.base import MIMEBase
# convert string into HTML format
from string import Template
import time
#it is used regular expression
import re

# Create the function for email sending
def email_func(email,Name):
    # Add the all credential for sending mail
    port = 587
    smtp_server = "vgipl.in"
    sender_email = "virtualdigital@vgipl.in"
    receiver_email = email
    # input("Type your password and press enter:")
    password = 'Virtual%08'

    # ssl default context for function.
    context = ssl.create_default_context()
    # Inside the Mail Body part
    message = MIMEMultipart("alternative")
    message["Subject"] = "e-Banker Core Banking Solutions for Nidhi Companys"
    message["From"] = "Virtual Galaxy Infotech Pvt.Ltd
<{}>".format(str(sender_email))
    message["To"] = receiver_email

   #Load the HTML file for sending email document
    fname =
r"D:\Python_Project\py\web_scraping\Email_Send\nidhi_email_page_02.html"




    # open the html file and read the content of file
    html_file = open(fname, 'r', encoding='utf-8')
    source_code = html_file.read()
   # Add the name in HTML file from given excel
    html_ = Template((source_code)).safe_substitute(code="Hello  "
+Name)#read_data['Name'][i]

   # Create the text message
    part2 = MIMEText(html_, "html")
    message.attach(part2)

   # Using try except for the sending the mail with email and password of SMTP server
    try:
        with smtplib.SMTP(smtp_server, port) as server:
            server.starttls(context=context)
            # Login to SMTP server using email and password
            server.login(sender_email, password)
            server.sendmail(sender_email, receiver_email, message.as_string())
        print('message send successfully')

# Exception handling
    except Exception as e:
           print(e)
#Code for function calling and Email validations
Invalid_Email=[ ]
Sending_Email=[ ]



try:
# Read the excel for the email receiver
read_data=pd.read_excel('D:\Python_Project\py\web_scraping\Email_Send\Testing
Data.xlsx')
    for i in range(len(read_data)):
        print('this is the name',read_data['Name'][i])
        print('this is the email',read_data['Email'][i])
        # Make a regular expression for validating an Email
        regex = r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b'
        if    (re.match(regex, read_data['Email'][i])):
            email_func(read_data['Email'][i],read_data['Name'][i])
            sender_email.append(read_data['Email'][i])
        else:
            Invalid_Email.append(read_data['Email'][i])
except Exception as e:
    print(e)
finally:
```

Collect the invalid mail using regular expression and save it into data frame

```
invalid_df=pd.DataFrame(Invalid_Email,columns=['Invalid_EmailId'])
invalid_df.to_excel('Invalid_email_id.xlsx',index=False)
Sending_Email_df=pd.DataFrame(Sending_Email,columns=['Sending_Email'])
Sending_Email_df.to_excel('Sending_Email_id.xlsx',index=False)
```

API DOCUMENTATIONS for Mail Sending using python

Summary

- First We create the project using django command
- Inside of project we create application using django command
- Then in Project folder setting.py file is use for Database integration and Install our application
- We create the table using model.py file and then Migrate using django command
- Then we create the form for frontend in form.py file for user interface inside the application folder
- Then we create the HTML and CSS inside the application Folder in templates folder for Frontend
- After that we import required files in view.py for business logic
- Create the route URL in url.py for application in both application folder and project folder
- And at the end we run the command for API start

#Open the terminal for Django folder creation

# Write the command for django application create the folder structure

- django-admin startproject project_name

# Go inside the project

- cd project_name

# Folder structure in project directory

```
- project_name/
-     manage.py
-     project_name /
-         __init__.py
-         settings.py
-         urls.py
-         asgi.py
-         wsgi.py
```

# Create the app

- django manage.py startapp app_name

# Inside the app_name project structure

```
- app_name /

-     __init__.py

-     admin.py

-     apps.py

-     migrations/

-         __init__.py

-     models.py

-     tests.py

-     views.py

-   app_name/templates/

                file_upload/form.html
```

```
file_upload/list.html
file_upload/base.html
file_upload/url.html
```

File Name: setting.py

# Provide the your system ip for run the application on your ip server

```
ALLOWED_HOSTS = ['192.168.1.76']
```

# Run the Application write the command on default port i.e. 8000

- Python manage.py runserver 192.168.1.76:8000

#Add the app name in the setting.py file In Installation

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'app_name',
]
```

# Add the Database connection in setting.py file

```
DATABASES = {
    'Default': {‘ENGINE’:   'django.db.backends.oracle',
    'NAME':     '192.168.1.42/orcl',
    'USER':     'py',
    'PASSWORD': 'py',
    #'PORT':     '1521'
```

```
      }
    }
```

File Name: model.py

```
#In model.py file we create the table in database which we connected in setting.py file
# Here we created two tables
from django.db import models
class Tutorial(models.Model):
    title = models.CharField(max_length=100)
    category = models.CharField(max_length=100)
    feature_image = models.FileField(upload_to='tutorial/images/')
    attachment = models.FileField(upload_to='tutorial/attachments/')

    def __str__(self):
        return self.title

 #Delete the statement arguments
    def delete(self, *args, **kwargs):
        self.feature_image.delete()
        self.attachment.delete()
        super().delete(*args, **kwargs)
#All the table Tutorial are show in the py name database or in default SQLite database
```

File Name: form.py

```
#In from.py file we create the front end from using html
from django import forms
# create form from model.py file
from .models import Tutorial
class TutorialForm(forms.ModelForm):
    class Meta:
        model = Tutorial
        fields = ['title', 'category', 'feature_image', 'attachment']
```

#Create the folder inside application folder app_name i.e. Templates

Path= app_name/templates/app_name/filename.html

Inside the html file

```
#Upload.html
<h2>upload Tutorial</h2>
<form method="post" enctype="multipart/form-data">
        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit">Upload</button>
</form>
```

#Upload_form.html

```
<div style="padding: 40px; margin: 40px; border: 1px solid #ccc">
    <h1>Django File Upload</h1>
    <form method="post" enctype="multipart/form-data">
      {% csrf_token %}
      {{ form.as_p }}
      <button type="submit">Submit</button>
    </form><hr>
    <ul>
    {% for document in documents %}
        <li>
            <a href="{{ document.upload_file.url }}">{{ document.upload_file.name }}</a>
            <small>({{ document.upload_file.size|filesizeformat }}) -
{{document.upload_date}}</small>
        </li>
    {% endfor %}
    </ul>
</div>
```

#Base.html

```
<!-- templates/base.html -->
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>{% block title %}Django Auth Tutorial{% endblock %}</title>
</head>
<body>
  <main>
```

```
    {% block content %}
    {% endblock %}
  </main>
</body>
</html>
```

#home.html

```
{% extends
'D:\Python_Project\py\web_scraping\Email_Send\email_send\file_upload\templates\file_
upload\base.html' %}


{% block title %}Home{% endblock %}


{% block content %}
{% if user.is_authenticated %}
  Hi {{ user.username }}!
  <p><a href="{% url 'logout' %}">Log Out</a></p>
  <p><a href="{% url 'password_reset' %}">Reset Password</a></p>
{% else %}
  <p>You are not logged in</p>
  <a href="{% url 'login' %}">Log In</a>
{% endif %}
{% endblock %}
```

#list.html

```
{% load static %}

<link rel="stylesheet" type="text/css" href="{% static
'Email_Send\email_send\file_upload\static\css\file_upload.css' %}">

Email_Send\email_send\file_upload\static\css\file_upload.css

<h2>Uploaded files Tutorials</h2>


<a href="{% url 'upload_tutorial' %}">Upload Tutorial</a>


{% if tutorials %}
<table style="margin: 20px;">
        <thead>
                <tr>
                        <th>HTML FILE </th>


                        <th>HTML </th>


                        <th>EXCEL   </th>


                        <th>DOWNLOAD</th>


                        <th>Action</th>
                </tr>
        </thead>
        <tbody>
                {% for tutorial in tutorials %}
                <tr>
                        <!-- <td><img src="{{ tutorial.feature_image.url }}"
width="150px">image_data</td> -->
```

```
                        <td>{{ tutorial.feature_image }}</td>
                        <td>{{ tutorial.title }}</td>
                        <td>{{ tutorial.category }}</td>
                        <td><a href="{{ tutorial.attachment.url }}"
target="_blank">Download</a></td>
                        <td>
                        <form method="post" action="{% url 'tutorial' tutorial.pk %}">
                                  {% csrf_token %}
                                  <button type="submit">Delete</button>
                             </form>
                        </td>
                   </tr>
                   {% endfor %}
              </tbody>
</table>
{% endif %}
```

File Name: view.py

```
# Import the Library for sending the email
from django.shortcuts import render, redirect
from .forms import TutorialForm
from .models import Tutorial
from io import StringIO
#import pandas for the dataframe
import pandas as pd
# we import database table turoial
from .models import Tutorial
# database connection import from setting.py file
```

```
from django.db import connection
# import Simple Mail Transfer Protocol library
# Import the SSL and SMTP Library
import smtplib, ssl
# for creating the text in email
from email.mime.text import MIMEText
# for multipart form data
from email.mime.multipart import MIMEMultipart
# for dataframe operations
import pandas as pd
# convert email to bytes
from email import encoders
# the MIMEText class is used to create email object text.
from email.mime.base import MIMEBase
# convert into HTML format
from string import Template
import time
import re

# Create the log
import logging
logger = logging.getLogger(__name__)

# Create the function for POST or GET
def tutorialList(request):
# From table take all the rows from tutorial table
    tutorials = Tutorial.objects.all()
    return render(request,
r'D:\Python_Project\py\web_scraping\Email_Send\email_send\file_upload\templates\file
_upload\list.html', { 'tutorials' : tutorials})


# Create the function for upload the file in functions
def uploadTutorial(request):
    try:
        if request.method == 'POST':
            form = TutorialForm(request.POST, request.FILES)
            if form.is_valid():


# Taking the HTML and EXCEL files from POST request
# Create and encode the file using bits to string
               html_raw_data=request.FILES.get('feature_image').read().decode("utf-8")
# Read the excel from the request attachment
                excel_raw_data = pd.read_excel(request.FILES.get('attachment'))


# sending mail using above file request object
# Create the mail sending code using smtp server and port
                def email_func(email,Name):
                    port = 587
                    smtp_server = "vgipl.in"
                    sender_email = "virtualdigital@vgipl.in"
                    receiver_email = email
                    password = 'Virtual#123'#Virtual%08' #input("Type your password and
                    press enter:")
# Create the default context of ssl which read the context
                     context = ssl.create_default_context()
```

```
# Message body part for creating in HTML page
          message = MIMEMultipart("alternative")
          message["Subject"] = "e-Banker Core Banking Solution for Nidhi Company"
          message["From"] = "Virtual Galaxy Infotech Pvt. Ltd
                                   <{}>".format(str(sender_email))
          message["To"] = receiver_email

# Read the HTML page with context
           fname = html_raw_data
          # html_file = open(fname, 'r', encoding='utf-8')
          # source_code = html_file.read()
# Add the name of email receiver
           html_ = Template((fname)).safe_substitute(code="Hello  "
           +Name)#read_data['Name'][i]
# Attach the mail with html file
          part2 = MIMEText(html_, "html")
          message.attach(part2)
          try:
            with smtplib.SMTP(smtp_server, port) as server:
              server.starttls(context=context)
             # Connect with the sender email and password the server
           server.login(sender_email, password)
            server.sendmail(sender_email, receiver_email, message.as_string())
            print('message send sucessfully')
          except Exception as e:
            logging.error(e)
```

```
# Apply the email validation to email address
# Code for function calling and Email validations
        Invalid_Email=[]
        try:
          read_data=excel_raw_data
          for i in range(len(read_data)):
            print('this is the name',read_data['Name'][i])
            print('this is the email',read_data['Email'][i])

# use regular expression for validating an Email
            try:
              regex = r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b'
              if (re.match(regex, read_data['Email'][i])):
                email_func(read_data['Email'][i],read_data['Name'][i])
             else:
                Invalid_Email.append(read_data['Email'][i])
            except Exception as e:
              logging.error(e)
# Exception Handaling of error
        except Exception as e:
          logging.error(e)
        finally:
# Get the invalid data for email invalid user
          invalid_df=pd.DataFrame(Invalid_Email,columns=['Invalid_EmailId'])
# save the invalid mail into excel file
        invalid_df.to_excel('Invalid_email_id_'+str(i)+'.xlsx',index=False)
        form.save()
```

```
# Redirect the html page to tutorial list

                return redirect('tutorial_list')

        else:

# form then redirect the mail using function

            form = TutorialForm()


#Redirect the page of html using Form

        return render(request,
r'D:\Python_Project\py\web_scraping\Email_Send\email_send\file_upload\templates\file
_upload\upload.html', {'form' : form})

    except Exception as e:

        logging.error(e)


#Delete Function for uploaded name with excel file

def deleteTutorial(request, pk):

    try:

        if request.method == 'POST ':

            tutorial = Tutorial.objects.get(pk=pk)

            tutorial.delete()

        return redirect('tutorial_list')

# exception handling error code

except Exception as e:

        logging.error(e)
```

File Name: url.py

```
#Inside application folder url.py given for routing
PATH = app_name/url.py
from django.urls import path
from .views import SignUpView
urlpatterns = [
    path('signup/', SignUpView.as_view(), name='signup'),
]


#IN url.py in Project folder
PATH = project_folder/url.py
Email_send URL Configuration
The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/3.2/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
```

```
#Import all the django libraries for creating url file
from django.contrib import admin
from django.urls.conf import include, include
from django.urls import path
#from file_upload import views as uploader_views
from django.conf.urls.static import static
from django.conf import settings
from file_upload import views
from django.views.generic.base import TemplateView

urlpatterns = [
    path('admin/', admin.site.urls),
    path('tutorials/upload/', views.uploadTutorial, name='upload_tutorial'),
    path('tutorials/', views.tutorialList, name='tutorial_list'),
    path('tutorials/<int:pk>', views.deleteTutorial, name='tutorial'),
    #path('', uploader_views.UploadView.as_view(), name='fileupload'),
    path('accounts/', include('file_upload.urls')),
    path('accounts/', include('django.contrib.auth.urls')),
    path('',
TemplateView.as_view(template_name=r'D:\Python_Project\py\web_scraping\Email_Se
nd\email_send\file_upload\templates\file_upload\home.html'), name='home')
]+ static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

#main service url

Run Command –

```
        Python manage.py runserver 192.168.1.120:8000
```

#192.168.1.120:8000/tutorials/

#http://192.168.1.120:8000/tutorials/upload/