CS 021 –Computer Programming I: Python
Use Cases – Batch File Renaming

Overview

The following procedure outlines the process through which we can manipulate large quantities of files with one Python script. This type of scripting can be used to accomplish anything from renaming a small group of files to creating automated email blasts to thousands of people. The following procedure will outline the steps necessary to create a simple batch file renaming script as well as a script that can send automated emails with those files attached.

Procedure

Note: this tutorial is for Windows, if you are on a Mac, simply add the 'sudo' markup before each command

1. To start, we need to create a new file where we can write our Python script. Open up IDLE and create a new file and save it to your Desktop using the following commands.
   a. Commands
      i. Create a new file - '**Ctrl + N**'
      ii. Save the file – '**Ctrl + S'**

2. Once our file has been created, we need to add the following code to your file.

```
#Import os (operating system library)
import os
new_name = "MySweetQRCode"
cnt = 1

#Initialize range-based for loop, that will execute on each file in the
current directory
for filename in os.listdir("."):

    #If filename has .png extension, do this
    if(filename.endswith(".png")):

        #Rename the file to our new_name variable + cnt + .png
        os.rename(filename, new_name + str(cnt) + ".png")
        cnt+=1
```

3. This snippet is all we need to rename a large group of files, although this isn't all we can do with this process. We can execute any action on a group of files using this script if we simply replace the code within the if statement that checks for the file extension. For example, if we wanted to send every file in a directory to someone via email, all we have to do is create a **send_mail** function and install a package called **email.**

4. First, we need to install the **email** package with PIP with the following command.
   a. Command
      i. **'pip install email'**

5. Once it has installed, we need to add the following imports to our script.

```
#Email library imports
import smtplib
import os.path as op
from email.mime.multipart import MIMEMultipart
from email.mime.base import MIMEBase
from email.mime.text import MIMEText
from email.utils import COMMASPACE, formatdate
from email import encoders
```

6. Now we can start writing our function that will send the actual email. Lets begin by defining the function.

```
def send_mail(send_from, send_to, subject, text, files, server):
```

As you can see, this function takes six parameters, all of which are necessary to send an email. The parameters are as follows: sender email (ex. abc@uvm.edu), recipient email (ex. def@uvm.edu), subject (text for subject line), text (email body), **files (attachments)**, and server (smtp email server).

7. After the function is defined, we need to construct the email itself. Add the following code to your **send_mail** function.

```
assert isinstance(send_to, list)
msg = MIMEMultipart()
msg['From'] = send_from
msg['To'] = COMMASPACE.join(send_to)
msg['Date'] = formatdate(localtime=True)
msg['Subject'] = subject
msg.attach(MIMEText(text))
```

This snippet is responsible for creating the email structure. We create a new message object (msg) and add the fields that we pass into the function as parameters.

8. Now that the email structure has been created, we need to attach the files in the current directory that we renamed earlier. We can do this by adding the following code to our **send_mail** function below the code in step 7.

```
for f in files or []:
       with open(f, "rb") as fil:
            part = MIMEApplication(
                fil.read(),
                Name=basename(f)
                )
       part['Content-Disposition'] = 'attachment; filename="%s"' % basename(f)
       msg.attach(part)
```

9. The block of code above loops through the array of input files, opens, and attaches them to the msg object that we created in step 7. Once this is completed, all we have to do is send the message over a smtp server using the **server** parameter. To do this, add the following code to your **send_mail** function.

```
smtp = smtplib.SMTP(server)
smtp.sendmail(send_from, send_to, msg.as_string())
smtp.close()
```

This code creates a connection with a server located at the input IP address **server**, sends the **msg** object that we created with the attached files and closes the connection when the request has completed.

10. Now that we have completed our **send_mail** function, we can use it in the loop of our original script. Instead of renaming all files that end with ".png", we need to collect them using an array and pass that array into our **send_mail** function as a parameter. Let's do this by adding the following code to your **send_mail** function.

```
fileArray = []

for filename in os.listdir("."):
    if(filename.endswith(".png")):
        fileArray.append(filename)

send_mail('senderemail@uvm.edu', 'recipient@uvm.edu', 'subject',
'body', fileArray, 'email server IP address')
```

11. When this code is added, our script will be complete. We will be able to manipulate files in a given directory, attach them to an email and send them to anyone we want. Although this may seem simple, it can be extremely powerful due to its wide range of applications.

Congratulations. Now you can spam anyone you'd like! Extremely entertaining but highly discouraged. Enjoy!