###Python People Emailer Automaton 30000

John Owen

Much of the code was inspired by Automate the Boring Stuff, by Al Sweigart (<https://automatetheboringstuff.com/>); especially Chapters 5, 6, 12, 14, 16, and Appendix B.

This program takes the names and emails of every person listed in the specified excel file and emails them.

This entire program—depending on how many people there are—can take some time, but can work in the background while the computer’s in use (Be certain not to log out/restart/turn the computer off!). Without this, emailing a single person can take quite some time. If you have 600 people in your database, for instance, emailing them all could take days.

Be aware that adding/changing columns can break the program and cause many headaches.

Note that you will need an excel file which lists the names and email addresses of the people you’ll be contacting. If the file also includes the date they were emailed, note also that you will need to update this manually. At least until someone adds that into this program, which shouldn’t be overly difficult.

Things worth adding:

-dates people were emailed

-maybe some automated option within the email you send for people to opt out?

-perhaps some way to read the sleepstudy8 inbox (which this program emails from) and remove any people whose emails did not go through?

###

#import modules

import os

import openpyxl as op

import sys #Is this required? idkmybffJill

import datetime

#Change directory to where the excel file is

os.chdir(r"\\where\ever\it\is")

#Define some stuff- the workbook, sheet, and lists—currently empty—of people and their emails

wb = op.load\_workbook(‘Excel\_File.xlsx’) #cannot be pw protected b/c openpyxl can’t load pw protected files. Or it can, and I just don’t know how. In either case: Ignore whining.

sht=wb.active

people = []

emails=[]

#Fill the empty lists with the people’ names and their emails

#Note that people names are in column 6, their emails in column 8, and the date in column 9.

for i in range(2, sht.max\_row+1):

if sht.cell(row=i, column=6).value != None:

#if People column isn’t empty

if sht.cell(row=i, column=8).value != None:

#if email column isn’t empty…

if type(sht.cell(row=i, column=9).value) != str:

#if date column has an actual date…

if sht.cell(row=i, column=9).value !=None:

#which is not empty…

if sht.cell(row=i, column=9).value < datetime.datetime(2016,12,1,0,0) != False:

#and which isn’t within the past couple months (NB: you’ll need to change the date for this each

time you rerun the program.)

people.append(sht.cell(row=i, column=6).value)

#add the names of people who pass through all these filters into the people list

elif sht.cell(row=9, column=9).value ==None:

people.append(sht.cell(row=i, column=6).value)

#add the names of people who do not have a date they’ve been emailed as well. Reminder: Be

sure to add the date after the program is run! Or add a function to do that in this program.

#Now do the same for emails:

for i in range(2, sht.max\_row+1):

if sht.cell(row=i, column=6).value != None:

if sht.cell(row=i, column=8).value != None:

if type(sht.cell(row=i, column=9).value) != str:

if sht.cell(row=i, column=9).value !=None:

if sht.cell(row=i, column=9).value < datetime.datetime(2016,12,1,0,0) != False:

emails.append(sht.cell(row=i, column=8).value)

elif sht.cell(row=9, column=9).value ==None:

emails.append(sht.cell(row=i, column=8).value)

#So now both the people list and email list are full. However, there may be some issues with the peoples’ names. So:

#Clean up the Unicode crap…

people = [item.replace(‘\u2019’, “ ‘ “) for item in people]

people= [item.replace(‘\xa0’, ‘ ‘) for item in people]

people= [item.replace(‘\u200b’, ‘ ‘) for item in people]

emails = [item.replace (‘\xa0’, ‘’) for item in emails] #do this bc of spaces after some emails…

emails = [item.replace (‘ ’, ‘’) for item in emails] #get rid of all spaces

emails = [item.replace(‘\u2019’, ““) for item in emails]

emails = [item.replace(‘\u200b, ““) for item in emails]

#Check that the length of each list is the same

len(people)

len(emails)

#if both are not the same, check back and see who’s being missed

#And Create the Dictionary

People\_dict = dict(zip(people, emails))

#create an empty list for people’ names & emails which, for whatever reason, didn’t go through.

failed=[]

#this list should(!) be added to during the loop when something goes wrong. It didn’t last time, though, so it (probably) won’t the next time. This isn’t program-breaking, but you may want to fix this. It’s an easy fix, all things considered.

#Finally, create the for loop which emails them all

#Note that you’ll need an email account to use this

For name, email in People\_dict.items():

#start with setting up the technical email stuff

smtObj=smtplib.SMTP(‘mail.SomeEmailAccount.org’, 25)

smtObj.ehlo()

smtObj.starttls()

smtObj.login(‘YourEmail@YourEmailAccount.org’, ‘PasswordGoesHere’)

#next, define what the body of the email will say

body=’’’Subject: Whatever the Header will be\nDear %s,\n\nThis is where the first paragraph goes. \nThis is where the 2nd one goes \nIf you have guessed, this is where the next one goes. \n\nSincerely,\nYourName Here\nTitle \nSecond Title\nPlaceofEmployment’’ %(name)

print(‘Sending email to %s…’ % (email))

#finally, send the emails to each person on the list to their email address. NB: the two have been linked through the creation of the dictionary.

sendmailStatus = smtObj.sendmail(‘YourEmail@YourEmailAccount.org’, email, body)

#if I remember how this works…: the sendmailStatus should run through the email list and use the people’ names in the body of the text. If there is a problem with the email, it will not finish it, so the list will not be empty when the function finishes running. This next part is what that catches; those emails which were a problem.

if sendmailStatus !={}:

print(‘There was a problem sending email to %s: %s’ % (email, sendmailStatus))

failed.append(email)

smtObj.quit()

#The program will finish once all people have been emailed. To check who had not been emailed, check the ‘failed’ var. (type it in to the terminal/wherever the program was run, once it’s complete).