# Attribute Email Robot (AER)

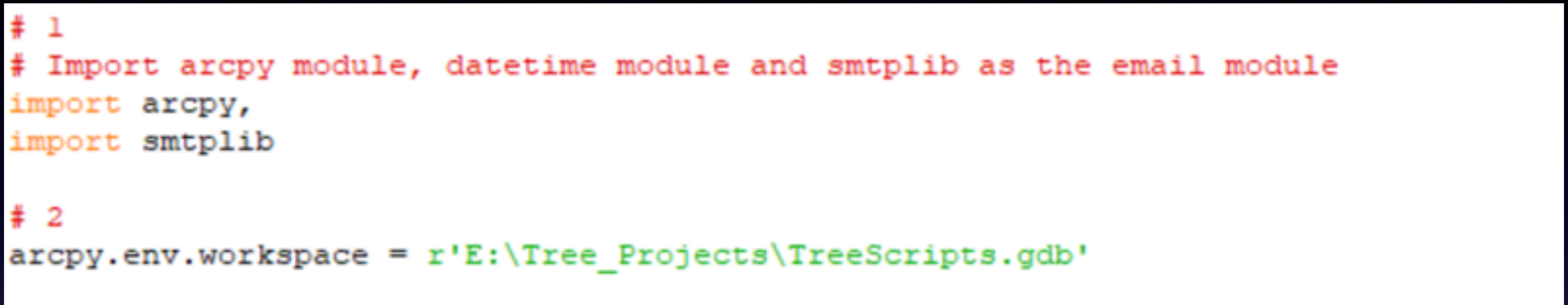
## A Python Scripting Project

Devin Byrne | GEOG 375 | Programming in ArcGIS

**Overview**

- The focus of the following python script will be able to perform a search cursor on the queried attributes of a particular ArcGIS Collector feature class that was updated by field technicians.
- The search cursor will print the record results of all the required addresses that were 'attempted' by field technicians for data collection.
- Manipulated address records are looped through search cursors from technicians entering data during routine studies.
- Results will be formatted into a code block cotaining placeholders, that will send automatic emails to the dispatcher, ——
- Requesting to reschedule a future appointment with that address to check on utilities or trees contained within the property.

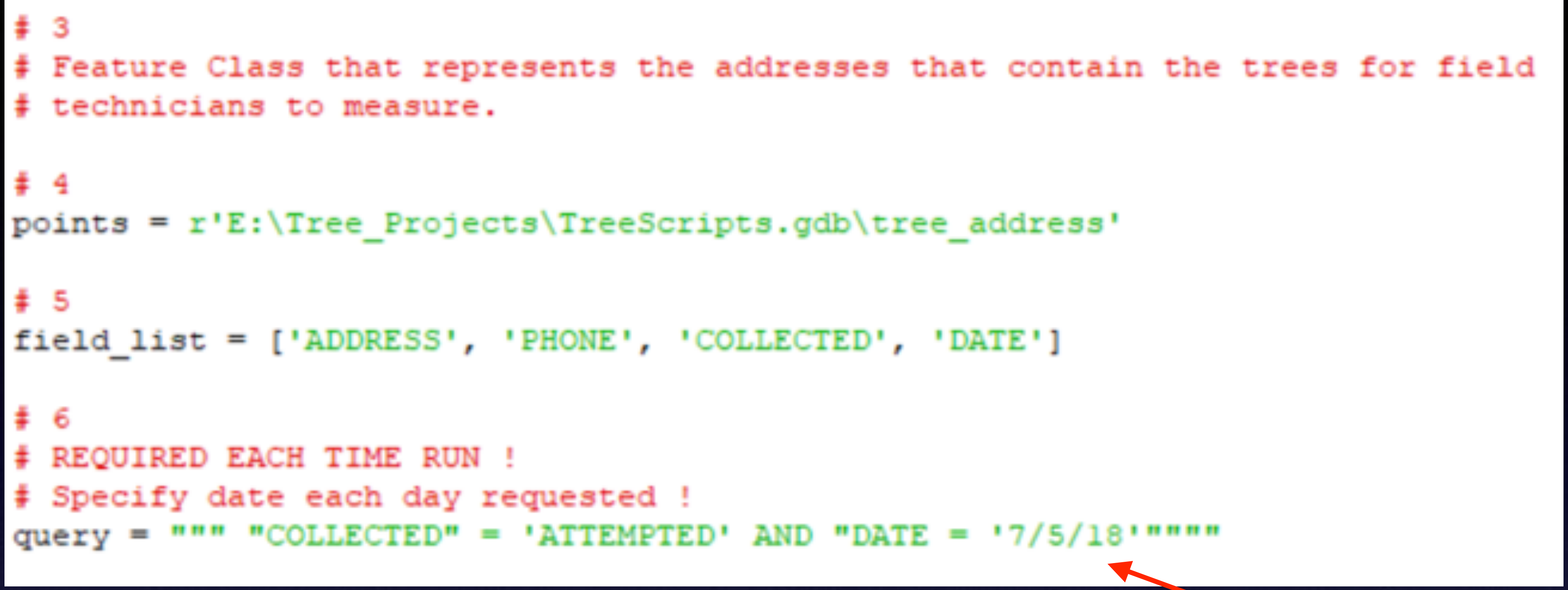
# Modules



```
# 1
# Import arcpy module, datetime module and smtplib as the email module
import arcpy,
import smtplib

# 2
arcpy.env.workspace = r'E:\Tree_Projects\TreeScripts.gdb'
```

- The first step was to import the `arcpy` and `smtplib` into a new file in Python 2.7.
- The ArcGIS environment workspace was specified as a folder pathname containing the address feature class that was used for this study.

# Variables



```
# 3
# Feature Class that represents the addresses that contain the trees for field
# technicians to measure.

# 4
points = r'E:\Tree_Projects\TreeScripts.gdb\tree_address'

# 5
field_list = ['ADDRESS', 'PHONE', 'COLLECTED', 'DATE']

# 6
# REQUIRED EACH TIME RUN !
# Specify date each day requested !
query = """ "COLLECTED" = 'ATTEMPTED' AND "DATE = '7/5/18'""""
```

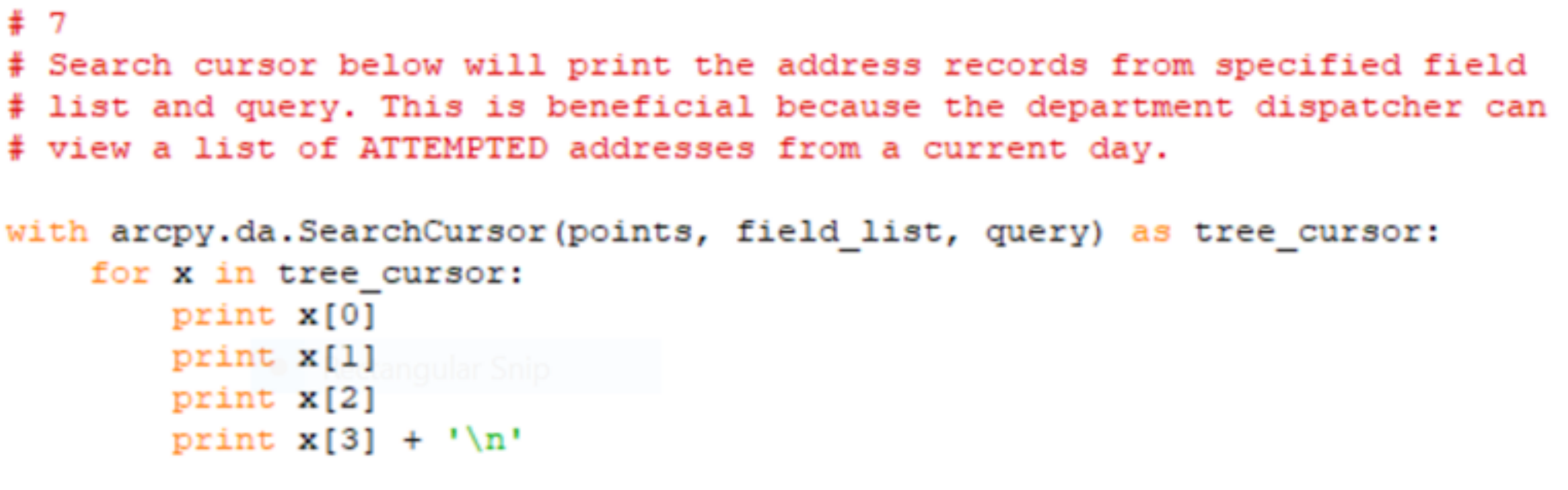
REQUIRED:

Each time this script is run, dispatchers or other office personnel are REQUIRED to alter the DATE within the PYTHON Query to the current or desired date. This procedure isolates the search to only records that were 'attempted' during the date specified.

## Variables configured for:

- tree_address point feature class pathname ()
- field_list that contains the feature class fields (i.e. such as address and phone number ) ()
- query that limits the addresses and the date ()

# Search Cursor



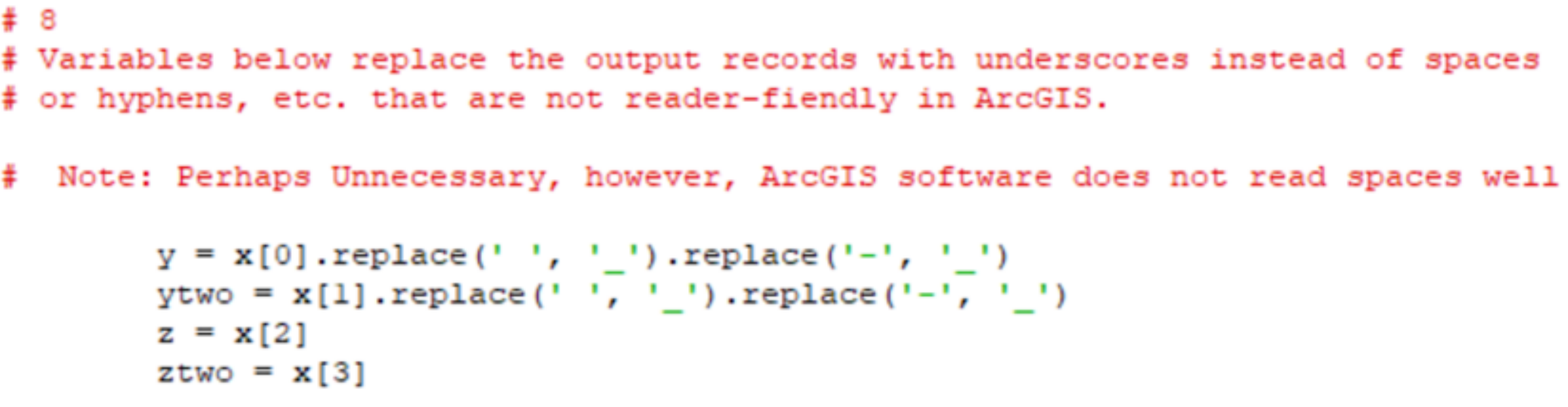
```
# 7
# Search cursor below will print the address records from specified field
# list and query. This is beneficial because the department dispatcher can
# view a list of ATTEMPTED addresses from a current day.

with arcpy.da.SearchCursor(points, field_list, query) as tree_cursor:
    for x in tree_cursor:
        print x[0]
        print x[1]
        print x[2]
        print x[3] + '\n'
```

- Points, field_list, query added as tuple parameters into da.SearchCursor that defines the criteria

- For Loop, translates to “any records being scanned that are detected by the query, will print records from fields specified — i.e. print x[2] translates to — “

- “Please print a list of queried records from COLLECTED.”

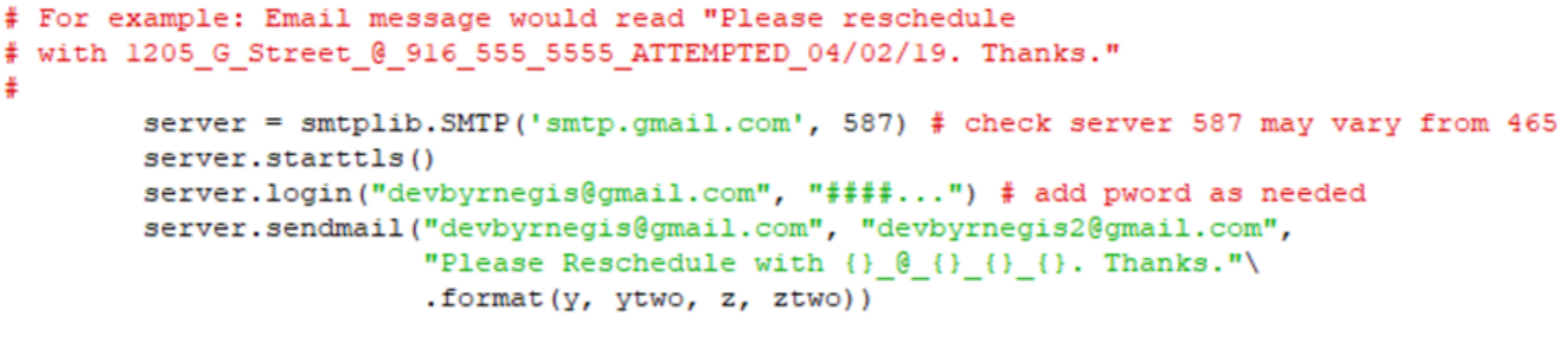
# Replacers



```
# 8
# Variables below replace the output records with underscores instead of spaces
# or hyphens, etc. that are not reader-fiendly in ArcGIS.

#  Note: Perhaps Unnecessary, however, ArcGIS software does not read spaces well

        y = x[0].replace(' ', '_').replace('-', '_')
        ytwo = x[1].replace(' ', '_').replace('-', '_')
        z = x[2]
        ztwo = x[3]
```

- Variables were assigned to the fields that will be formatted with underscores instead of spaces.
- ArcGIS does not interpret spaces very well.
- Perhaps these variables are unnecessary, but they will ensure the continuity of printed records from the extracted ArcGIS feature class anyways.

# Module: smtplib
## (Simple Mail Transfer Protocol)



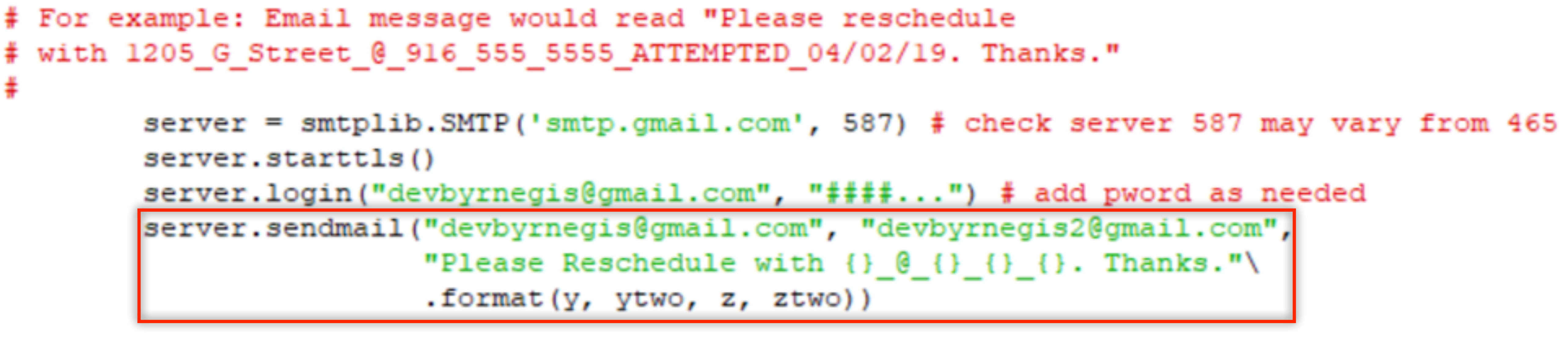
```
# For example: Email message would read "Please reschedule
# with 1205_G_Street_@_916_555_5555_ATTEMPTED_04/02/19. Thanks."
#
        server = smtplib.SMTP('smtp.gmail.com', 587) # check server 587 may vary from 465
        server.starttls()
        server.login("devbyrnegis@gmail.com", "####...") # add pword as needed
        server.sendmail("devbyrnegis@gmail.com", "devbyrnegis2@gmail.com",
                        "Please Reschedule with {}_@_{}_{}_{}. Thanks."\
                        .format(y, ytwo, z, ztwo))
```

- The first variable defines the GMail server to be accessed for this operation.
- Typical GMail servers run on server port # 587, so we will use '587' as the tuple parameter when defining the server port within the server variable.
- Note: Please be advised to modify the GMail account's security settings to accept python code before executing this script successfully.
- Turn Allow less secure apps to ON within GMail Settings.

- WARNING: Be aware that this makes it easier for others to gain access to your account.

# Module: smtplib

## (Simple Mail Transfer Protocol)

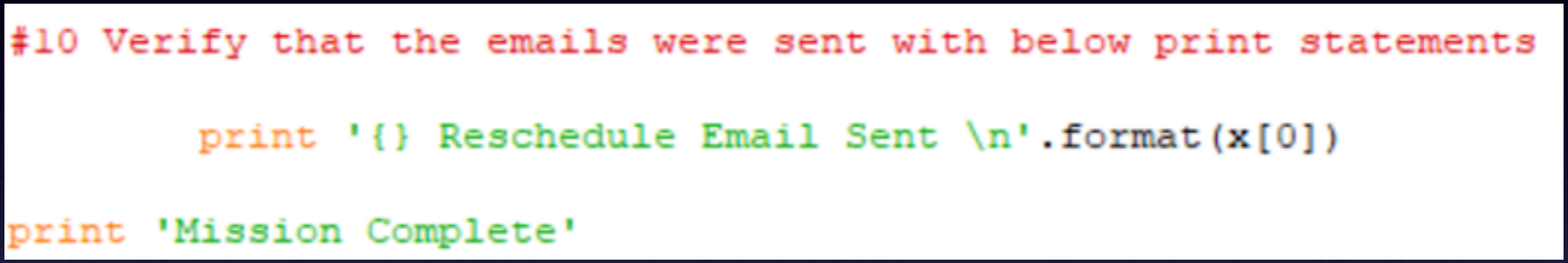


```
# For example: Email message would read "Please reschedule
# with 1205_G_Street_@_916_555_5555_ATTEMPTED_04/02/19. Thanks."
#
        server = smtplib.SMTP('smtp.gmail.com', 587) # check server 587 may vary from 465
        server.starttls()
        server.login("devbyrnegis@gmail.com", "####...") # add pword as needed
        server.sendmail("devbyrnegis@gmail.com", "devbyrnegis2@gmail.com",
                        "Please Reschedule with {}_@_{}_{}_{}. Thanks."\
                        .format(y, ytwo, z, ztwo))
```

IMPORTANT STEP:

- The above line of code outlined is the most important step of the script because it formats the address, phone number and date attempted from the extracted values returned by the search cursor loop prior to this step, back into the email message to be sent to the dispatcher, that contains a sequence of {} placeholders.
- These placeholders exist so the (.format(x[3]) extension for example, will cast those values from field_list (i.e. field[3] = 'DATE') into the {} placeholders sequenced in the email message as a string.
- Thus, providing the input needed for contact information to be formatted into the email message content; sent to the dispatcher for review.

# Finale



```
#10 Verify that the emails were sent with below print statements

        print '{} Reschedule Email Sent \n'.format(x[0])

print 'Mission Complete'
```

- Finally, two print statements verify that the emails were sent and the code was run successfully.

- Future refinements of this script will include, a try and except format to the overall deign of the program that will check for errors specific to ArcGIS software being processed.