'Blue Quiz' Committee Calculator

Explanation

So the President's Cabinet has put together a committee to create this personality quiz that'll provide students with a select number of organizations that they can join to get more involved on campus. There's much that goes into a project such as this, obviously, but this chunk specifically is the program that'll be taking student responses and processing them so that they can get proper results. Ideally, this would be able to be expanded in the future to include clubs, so the approach should be simple enough to:

1. Be expandable to include more organizations and other venues as time passes
2. Give each organization a decently fair chance of being selected via student responses
3. As simple as I can get it. :)

Approach

So about a week ago at the time of writing this I was bored and work and honestly a little *stressed* so I wrote up a step-by-step list of everything this program needs to do in order to run properly. It's very likely this may change, but as of right now here's the basic recipe for this program to function:

1. Run Consistently
2. Check if there's file in the directory that starts with 'BQR' *(Blue Quiz Response)*
3. Open the BQR file and use a while loop to run through each line
4. For each line,
    - Set the blank dictionaries. One holds the attributes/interests, the other holds the organizations
    - Split the line on its comments and put the information into an array
    - Go through every answer and input the point equivalents into a dictionary of all the attributes and interests
    - Have another dictionary of committees, and have each one pull the points for its respective attributes and interests
    - Return the three committees with the highest point totals
    - Send an email with that information
5. Delete the file when all the lines have been run through

Now, obviously, there's a lot there. I'll break down my thoughts on each of those goals as best as I can, so anyone reading this in the future has a better idea of how I intend to achieve the goal, and also so I can think through what I'm doing.

Number 1

So I had a few ideas on how to go about this goal. I need this program to check if there's a file relatively often so it'll run around the time it gets put in the directory. The way I plan to do this is, I assume we'll be using a Windows computer, by using the Task Scheduler. Every five or ten minutes I'll have the program run this through the `python.exe`, and then in the code I'll have a function that checks for the filename with the 'BRQ' prefix on it. If it's there, then great! Pull that and start processing. If not, then just exit the program and don't worry about it until the next run. This seems to be the best solution in my mind for getting this program up and running. Considering the file checking is the first thing the program does, it should be a very quick and easy process just to run it through Python every so often. I'd consider setting the interval to be every half hour which is when the files get exported, but I think having it run every five minutes is best because if the timing gets off it'll stick check every often. If somehow the housing for the program gets turned off, I'm not really sure how it'll work with the exports, but if it does and a bunch of files begin to pile up, every five minutes the program will run and pull one of the files. Seems like a decent plan.

Number 2

Unfortunately, Qualtrics is a pain and won't let me export after each response, and only lets me export at *minimum* once a day at a specific time. Thankfully, each survey can have up to fifty automations, so I'm just going to put *forty-eight* of them on there that export every thirty minutes. It *also* has the fun issue where responses collected within an hour of an export aren't counted until the next, so unfortunately people are just gonna have to wait an hour or so for their responses to be sent through one of my, again, *forty-eight* automations. I'm not looking forward to putting those all on there. This is the support page on how to do all that.

But, I digress. The second goal of the program is, once those exports get thrown into some directory, it needs to read it and open it up. I do know how to do the reading files part, that isn't difficult. The only issue I see arising here *(besides the aforementioned problem of not knowing where this program is going to be put)* is getting the specific file name. The Qualtrics exports tack on a date and time at the end of the file, making it difficult for me to know the exact file name. Thankfully, Qualtrics *also* lets me add a prefix to the filename. This is where I can use something from the 'glob' library, which will let me specifically target if a file with that prefix exists. This stack overflow page and this python tutorial page might provide some help in this endeavor as well.

Number 3

This is just opening the file and running a loop through each line. Nothing special. The only thing I'll have to pay attention to is that the first three lines are just unhelpful tags, and the responses I'm looking for aren't until the end of the existing lines, since Qualtrics in it's *ever* helpful nature, has decided to give me such useful information such as the time they took to finish the quiz and, of course, their *latitude and longitude*. Thanks Qualtrics! Very cool!

Thankfully, that shouldn't be too difficult to do. The lines are separated by newlines and it's a CSV with a specific structure, so it won't be to bad to get over to that part of the line. Just one loop to cut out the first three lines *(I think I can read them in a certain way as well. I'll have to look.)*, followed by a nested loop that for every subsequent line, skips over a bunch of commas, and then begins reading in the points. Something like that. If I get stuck, I think I do something similar in the 'Big Data Processing' program I have, so I can take a look at that.

Number 4

This should hopefully be pretty self explanatory. Do all the stuff mentioned in a big loop. Nothing fancy.

As for worries, the only 'known unknown' I've got to deal with here is that last one about sending the committees through an email. This will likely not be too difficult once I get the housing for this program figured out, but I do have this documentation and this tutorial page to help me out when that time eventually comes. Past that, there's nothing in this program I really feel I *don't* already know how to do.

Number 5

Just deleting a file here. That's just a `os.remove(filename)` command. Whoopee.

And that's it! That's the program, ideally. Again, subject to change as I actually build the thing, but at the moment that seems to cover everything.

Pseudocode

Main

```
If the namespace is the main function,
    # File Processing
    Set a variable to hold the filepath, which is returned from 'getFile()'
    Set a the file object by opening up the file at the filepath
    Call 'lineProcessor()' and send it the file object
    Close the file
    Delete the file
```

GetFile

```
getFile() is defined
    """ This function locates the CSV file with 'BRQ' in it, and pulls from that. """
    Attempt to do the following:
        -Set the file path string using the 'glob' library.
    If it returns an IndexError:
        -Exit the program
    Return the file path
```

LineProcessor

```
lineProcessor() is defined, and given the exported CSV file
    """ This function takes the data from the CSV file, rips out the necessary info, and processes it """
    # Setup
    Set a skip variable that'll be used to ignore the first three lines of the CSV
    Set an empty list that'll hold all the quiz information

    # Processing
    For each line in the file,
        -If we've skipped past the first three lines,
            ## Ripping
            --Strip the newlines out
            --Create a list with each element in the file split on the commas
            --For each element index in the range of that list,
                ---If the index is at or beyond the email line (index 17 or greater),
                    ----Add the element at that index to the quiz information list
            ## Allocating
            --Set the important info list to be equal to 'allocator()', which has the quiz info list sent to it
            ## Reporting
            --Call 'emailUser()' and send it the important info list.
        -Otherwise,
            --Increment the skip variable by one
```

Allocator

```
allocator() is defined, and given the orderly list of quiz data
    """ This function takes the orderly list of data and sorts all of the points and whatnot. """
    # Setup
    Set the attribute dictionary
    Set the dictionary with all the different organizations
    For each item in the orderly list of data,
        -Starting with index 1,
            --Convert the element at that index into an integer

    # Council Formatting
```

```
''' Due to the college major question allowing multiple choices, that heavily screws up the indexes. This is to remedy that. '''
Set a 'difference' variable that takes the length of the orderly list and subtracts it by the amount of questions.
Set a blank list for the colleges they selected
If that difference variable is greater than zero,
    Increment with a 'for' loop the value of the difference variable,
        Add the popped return of Index 2 to the college list
Add the value of Index 1 to the college list, but DO NOT REMOVE IT

# Point Allocation
''' If future questions are added, just add the lines here '''
Index 3:
    -Add points to 'project', 'detail',
Index 4:
    -Add points to 'political'
Index 5:
    -If 1 is the answer,
        --Add points to 'hype' and 'party' and 'athletic'
Index 6:
    -Add points to 'helpful' and 'academic' and 'detail' and 'project'
Index 7:
    -=== to be added ===
Index 8:
    -If 1 is the answer,
        --Add points to 'hype' and 'athletic' and 'spirit'
    -If 2 is the answer,
        --Add points to 'creative'
    -If 3 is the answer,
        --Add points to 'party' and 'music' and 'extroverted'
    -If 4 is the answer,
        --Add points to 'academic' and 'mentor'
Index 9:
    -If 1 is the answer,
        --===not sure what to add to here===
    -If 2 is the answer,
        --Add points to 'helpful' and 'spirit' and 'service'
    -If 3 is the answer,
        --Add points to 'mentor' and 'extroverted' and 'leader'
    -If 4 is the answer,
        --Add points to 'political' and 'leader'
    -If 5 is the answer,
        --Add points to 'detail' and 'leader'
    -If 6 is the answer,
        --Add points to 'extroverted'
Index 10:
    -Add points to 'planner', project', 'detail'

# Processing
''' If future organizations are added, add the lines here. Add them to the dictionary as well. '''
PCab:
    -Add points from 'leader', 'service', 'project', and 'political'
Fees:
    -Add points from 'detail' and 'time'
GRC:
    -Add points from 'political' and 'academic'
BlueCrew:
    -Add points from 'service', 'hype', and 'helpful'
Activities:
    -Add points from 'party', 'planner', and 'extroverted'
Series:
    -Add points from 'extroverted', 'creative', and 'music'
Traditions:
    -Add points from 'helpful', 'planner', 'extroverted', and 'spirit'
HURD:
    -Add points from 'hype', 'extroverted', and 'athletic'
Council:
    -If their college score is less than 9,
    If their college score is less than 9,
        --Add points from 'service', 'planner', and 'academic'
        --Add a small bump since they know the college their major is in.
SAA:
    -Add points from 'helpful', 'extroverted', and 'spirit'
Service:
    -Add points from 'service', 'helpful', and 'mentor'
Statesman:
    -Add points from 'creative' and 'mentor'
Radio:
    -Add points from 'creative' and 'music'
B-Light:
    -Add points from 'detail' and 'creative'
FSL:
    -Add points from 'service', 'hype', and 'parties'

# Reporting
Add the email into the important info list
Sort the organization dictionary by the highest value in descending order
For the first three keys in that dictionary,
    -Add each one of them to the important info list
If their college score is less than 9,
    -For each element in the college list,
     --Add that element's number to the end of the important info list
Return the important info list
```

EmailUser

```
emailUser() is defined, and given the list of important info
    """ This function handles the email that sends the top three organizations """
    # Setup
    Set the sending email
    Set the recipient email, which is the first element in the important info list
    Set the sending password
    Set the main message to be a MIMEMultipart Alternative
    Set a dictionary with each committee and its little blurb in plain text and HTML
    Set a college council dictionary with the link to each committee's signup websites

    # Compile Message
    Set the message subject
    Set the message "From" the sending quiz address
    Set the message "To" the recipient email
    Set a 'text' string with just the introduction
    Set the 'html' string with just the introduction
    For each element in the plain dictionary,
        -If the current element's key is equal to one of the three committees in the important info list,
            --Add its plain text blurb to the 'text' string
            --Add its HTML blurb to the 'html' string
            --If the current element is the college council,
                ---For every element in the important info list beyond Index 3,
                    ----Add its college council link to the end of the 'text' string
                    ----Add its college council link to the end of the 'html' string
    Turn the 'text' string into a MIMEText object
    Turn the 'html' string into a MIMEText object
    Attach the plain text object to the main message
    Attach the html object to the main message

    # Sendoff
    Set the context to be the default context
    With the SMTP library, given the server address, the port, and the context,
        -Login to the server
        -Send the email to the sender and give it the sending email, the recipient email, and the created message
```

Python Code

Main

```
if __name__ == '__main__':
    # File Processing
    filepath = getFile()
    fileObj = open(filepath)
    lineProcessor(fileObj)
    fileObj.close()
    os.remove(filepath)
```

GetFile

```
def getFile():
    """ This function locates the CSV file with 'BRQ' in it, and pulls from that. """
    try:
        path = glob.glob('../**/BRQ*')[0]  # Takes the first file with 'BRQ' in the directory.
    except IndexError:
        sys.exit(1)
    return path
```

LineProcessor

```
def lineProcessor(file):
    """ This function takes the data from the CSV file, rips out the necessary info, and processes it """
    # Processing
    skip = 0
    for line in file:
        if skip > 2:
            ## Ripping
            quizResponses = []
            line = line.strip('\n')
            line = line.replace('"', "")
            elementList = line.split(',')
            for el in range(len(elementList)):
                if el >= 17:
                    quizResponses.append(elementList[el])
            ## Allocating
            importantInfo = allocator(quizResponses)
            ## Reporting
            emailUser(importantInfo)
        else:
            skip += 1
                skip += 1
```

Allocator

```
def allocator(data):
    """ This function takes the orderly list of data and sorts all of the points and whatnot. """
    # Setup
    attributes = {
        "leader": 0,
        "service": 0,
        "project": 0,
        "detail": 0,
        "time": 0,
        "political": 0,
        "hype": 0,
        "helpful": 0,
        "party": 0,
        "planner": 0,
        "extroverted": 0,
        "creative": 0,
        "spirit": 0,
        "athletic": 0,
        "academic": 0,
        "mentor": 0,
        "music": 0,
    }
    orgs = {
        "pcab": 0,
        "fees": 0,
        "grc": 0,
        "blucru": 0,
        "activity": 0,
        "series": 0,
        "trad": 0,
        "hurd": 0,
        "council": 0,
        "saa": 0,
        "serve": 0,
        "states": 0,
        "radio": 0,
        "blight": 0,
        "fsl": 0
    }
    for x in range(len(data)):
        if x > 0:
            data[x] = int(data[x])

    # Council Formatting
    '''The college major question (Q1) allows multiple choices, and that heavily screws up the indexes. This is to
    remedy that. '''
    questions = 12  # Change for the set amount of questions there should be for the quiz.
    diff = len(data) - questions
    colleges = []
    if diff > 0:
        for x in range(diff):
            colleges.append(data.pop(2))
    colleges.append(data[1])

    # Point Allocation
    ''' If future questions are added, just add the lines here '''
    # Q3: --------------------------------
    attributes["project"] += data[3]
    attributes["detail"] += data[3]
    # Q4: --------------------------------
    attributes["political"] += data[4]
    # Q5: --------------------------------
    if data[5] == 1:
        attributes["hype"] += data[5]
        attributes["party"] += data[5]
        attributes["athletic"] += data[5]
    # Q6: --------------------------------
    attributes["helpful"] += data[6]
    attributes["academic"] += data[6]
    attributes["detail"] += data[6]
    attributes["project"] += data[6]
    # Q7: === to be added ===
    # Q8: --------------------------------
    if data[8] == 1:
        attributes["hype"] += 3
        attributes["athletic"] += 3
        attributes["spirit"] += 3
    elif data[8] == 2:
        attributes["creative"] += 3
    elif data[8] == 3:
        attributes["party"] += 3
        attributes["music"] += 3
        attributes["extroverted"] += 3
    elif data[8] == 4:
        attributes["academic"] += 3
        attributes["mentor"] += 3
    # Q9: --------------------------------
    if data[9] == 1:
        # ===not sure what to add to here===
        attributes["political"] += 1
    elif data[9] == 2:
        attributes["helpful"] += 3
        attributes["spirit"] += 3
        attributes["service"] += 3
    elif data[9] == 3:
        attributes["mentor"] += 3
        attributes["extroverted"] += 3
        attributes["leader"] += 3
    elif data[9] == 4:
        attributes["political"] += 3
        attributes["leader"] += 3
    elif data[9] == 5:
        attributes["detail"] += 3
        attributes["leader"] += 3
    elif data[9] == 6:
        attributes["extroverted"] += 3
    # Q10: --------------------------------
    attributes["planner"] += data[10]
    attributes["project"] += data[10]
    attributes["detail"] += data[10]

    # Processing
    ''' If future organizations are added, add the lines here. Add them to the dictionary as well. '''
    orgs["pcab"] = (attributes["leader"] + attributes["service"] + attributes["project"] + attributes["political"])
    orgs["fees"] = (attributes["detail"] + attributes["time"])
    orgs["grc"] = (attributes["political"] + attributes["academic"])
    orgs["blucru"] = (attributes["service"] + attributes["hype"] + attributes["helpful"])
    orgs["activity"] = (attributes["party"] + attributes["planner"] + attributes["extroverted"])
    orgs["series"] = (attributes["extroverted"] + attributes["creative"] + attributes["music"])
    orgs["trad"] = (attributes["helpful"] + attributes["planner"] + attributes["extroverted"] + attributes["spirit"])
    orgs["hurd"] = (attributes["hype"] + attributes["extroverted"] + attributes["athletic"])
    if data[1] < 9:
        orgs["council"] = (attributes["service"] + attributes["planner"] + attributes["academic"] + 2)
    orgs["saa"] = (attributes["helpful"] + attributes["extroverted"] + attributes["spirit"])
    orgs["serve"] = (attributes["service"] + attributes["helpful"] + attributes["mentor"])
    orgs["states"] = (attributes["creative"] + attributes["mentor"])
    orgs["radio"] = (attributes["creative"] + attributes["music"])
    orgs["blight"] = (attributes["detail"] + attributes["creative"])
    orgs["fsl"] = (attributes["service"] + attributes["hype"] + attributes["party"])

    # Reporting
    impInfo = [data[0]]
    orgsList = sorted(orgs.items(), key=lambda k: k[1], reverse=True)
    for x in range(3):
        if orgsList[x][0] == "council" and (9 or 10 in colleges):
            orgsList.pop(x)
        impInfo.append(orgsList[x][0])
    if data[1] < 9:
        for el in colleges:
            impInfo.append(el)
    return impInfo
```

EmailUser

```
 def emailUser(info):
""" This function handles the email that sends the top three organizations """
# Setup
sender = ""  # Undetermined as of yet
receiver = info[0]
password = ""  # Undetermined as of yet
message = MIMEMultipart("alternative")
orgs = {
    "pcab": ["http://labs.jensimmons.com/2016/examples/images/testscreen-large.jpg",
             "President's Cabinet",
             "Work on initiatives set by the USUSA President. It’s function/form is different from year-to-year, but for our pur
             ""],
    "fees": ["http://labs.jensimmons.com/2016/examples/images/testscreen-large.jpg",
             "Student Fee Board",
             "Advisory board to the University President to decide on student fees. Receives presentations, discusses, reviews,
             ""],
    "grc": ["http://labs.jensimmons.com/2016/examples/images/testscreen-large.jpg",
            "Government Relations Council (GRC)",
            "Advocate for student interest to state & local officials & raise awareness of civic issues on campus. Plans Gripe N
            ""],
    "blucru": ["http://labs.jensimmons.com/2016/examples/images/testscreen-large.jpg",
               "Blue Crew",
               "Assist in the marketing of events & helps set up/take down for events",
               ""],
    "activity": ["http://labs.jensimmons.com/2016/examples/images/testscreen-large.jpg",
                 "Activities Committee",
                 "Plans & executes The Howl, Mardi Gras, & End of Year Bash",
                 ""],
    "series": ["http://labs.jensimmons.com/2016/examples/images/testscreen-large.jpg",
               "Series Committee",
               "Plans & executes regular/recurring events that are generally more artistic like PoBev & Moonlight & Music",
               ""],
    "trad": ["http://labs.jensimmons.com/2016/examples/images/testscreen-large.jpg",
             "Traditions Committee",
             "Plans & executes Homecoming week, Mr.USU, Sweater Swap, & High Stakes Bingo",
             ""],
    "hurd": ["http://labs.jensimmons.com/2016/examples/images/testscreen-large.jpg",
             "HURD Committee",
             "Spread awareness & excitement around USU athletics, plan events & initiatives that increases attendance at games,
             ""],
    "council": ["http://labs.jensimmons.com/2016/examples/images/testscreen-large.jpg",
                "College Councils",
                "Advocate for student interests to their college’s administration (deans & department heads), plan & execute ser
                ""],
    "saa": ["http://labs.jensimmons.com/2016/examples/images/testscreen-large.jpg",
            "Student Alumni Association",
            "Create lifelong aggies. Focuses on donors and guest speakers. Engage alumni. Plan & execute initiatives to connect
            ""],
    "serve": ["http://labs.jensimmons.com/2016/examples/images/testscreen-large.jpg",
              "Service Center Programs",
              "Some of the programs run out of the center are SNAC, gleaning, best buddies, Special Olympics, Athletics United,
              ""],
    "states": ["http://labs.jensimmons.com/2016/examples/images/testscreen-large.jpg",
               "Statesman",
               "Student news. Gives students an opportunity to write",
               ""],
    "radio": ["http://labs.jensimmons.com/2016/examples/images/testscreen-large.jpg",
              "Aggie Radio",
              "Provides students with opportunities to DJ, create radio talk shows, podcasts, & plan/coordinate events (mainly c
              ""],
    "blight": ["http://labs.jensimmons.com/2016/examples/images/testscreen-large.jpg",
               "Blue Light",
               "Meet content creation needs for clients, primarily USUSA",
               ""],
    "fsl": ["http://labs.jensimmons.com/2016/examples/images/testscreen-large.jpg",
            "Fraternity & Sorority Life",
            "Advance personal development through community service, philanthropic fundraising, & activities.",
            ""]
}
councils = {
    1: ["Arts Application:", ""],
    2: ["Agriculture Application:", ""],
    3: ["Engineering Application:", ""],
    4: ["Humanities & Social Sciences Application:", ""],
    5: ["Science Application:", ""],
    6: ["Education Application:", ""],
    7: ["Huntsman Application:", ""],
    8: ["Natural Resources Application:", ""],
}

# Compile Message
message["Subject"] = "Blue Quiz Involvement Survey Results!"
message["From"] = sender
message["To"] = receiver
text = """\
    “The only way you can taste life is with involvement.” Sadhguru
    Thank you for filling out our survey. You are a great match for the following three committees. Our goal here
    at USU is to cultivate an environment of development and growth and we need your help. Every single student
    can make a difference in our community and help others feel welcomed and heard, and it starts with
    involvement. Check out the options listed below or browse for other opportunities listed on our website. We
    can’t wait to hear your perspectives and ideas and together change our school for the better. """
html = """\
    <!DOCTYPE html>
    <html>
        <head>
            <style type="text/css" media="screen">
                img {
                    display: block;
                    margin-left: auto;
                    margin-right: auto;
                    width: 100%;
                    max-width: 900px;
                    height:  auto;
                    object-fit: contain;
                    align-content: center;
                }
                h1{
                    font-family:sans-serif;
                    text-align: center;
                }
                p{
                    font-family: Georgia, serif;
                    text-align: center;
                    font-size: 12pt;
                }
                table{
                    background-color: #0F2439;
                    border-spacing: 0;
                    border-collapse: collapse;
                    table-layout: fixed;
                    margin: 0;
                }
                td{
                    display: block;
                    margin-left: auto;
                    margin-right: auto;
                }
                ol{
                    font-family: Georgia, serif;
                    font-size: 12pt;
                }
            </style>
        </head>
        <body>
            <div style="background-color: #0F2439; backface-visibility: 75%;" >
                <img src="https://www.usu.edu/admissions/images/USU_Get_Involved.jpg" alt="Students playing pool">
            </div>
            <div style="background-color:#eee; padding:8px; word-wrap: break-word;">
                <h1 style="font-family:garamond; text-align:center;">"The only way you can taste life is with involvement.” - <e
                <p style="font-family:garamond; font-size: 16pt;">Thank you for filling out our survey. You are a great match fo
            </div>
            <table role="presentation" cellspacing="0" cellpadding="0" border="0" width="100%">
                <tbody>
    """
for x in range(3):
    if info[x+1] == "council":
        text += """
        """ + orgs[info[x+1]][1] + """
            Purpose: """ + orgs[info[x+1]][2] + """
        """
        html += '''
        <tr>
            <td aria-hidden="true" height="40" style="font-size: 0px; line-height: 0px;">&nbsp;</td>
        </tr>
        <tr>
            <td>
                <img src="''' + orgs[info[x+1]][0] + '''" style="width: 100%; max-width: 600px;">
            </td>
        </tr>
        <tr>
            <td style="padding: 10px 20px 20px 20px; font-family:sans-serif; background-color: #eee; width: 100%; max-width: 560
                <h1 style="margin: 0 0 10px; text-align: left; font-family:Georgia, serif;">''' + orgs[info[x+1]][1] + '''</h1>
                <p style="text-align: left;"><b>Purpose: </b>''' + orgs[info[x+1]][2] + '''</p>
        '''
        x = 4
        while x < len(info):
            text += councils[info[x]][0] + ": " + councils[info[x]][1] + "\n"
            html += '''<p style="text-align: left; margin: 0 0 10px;"><b>''' + councils[info[x]][0] + '''</b><a href="''' + coun
            x += 1
        html += """
                       </td>
                   </tr>"""
    elif info[x+1] == "pcab":
        text += """
        """ + orgs[info[x+1]][1] + """
        Purpose: """ + orgs[info[x+1]][2] + """
        \t 1. Enhancing equity on USU campuses
        \t 2. Increasing access to involvement opportunities
        \t 3. Amplifying student voices
        \t 4. Promoting & fundraising for scholarship opportunities that incentivize service & leadership.
        Application Link: """ + orgs[info[x+1]][3] + """
        """
        html += '''
        <tr>
            <td aria-hidden="true" height="40" style="font-size: 0px; line-height: 0px;">&nbsp;</td>
        </tr>
        <tr>
            <td>
                <img src="''' + orgs[info[x+1]][0] + '''" alt="TestImage" style="width: 100%; max-width: 600px;">
            </td>
        </tr>
        <tr>
            <td style="padding: 10px 20px 20px 20px; font-family:sans-serif; background-color: #eee; width: 100%; max-width: 560
                <h1 style="margin: 0 0 10px; text-align: left; font-family:Georgia, serif;">''' + orgs[info[x+1]][1] + '''</h1>
                <p style="text-align: left;"><b>Purpose: </b>''' + orgs[info[x+1]][2] + '''</p>
                <ol>
```

```
                    <li>Enhancing equity on USU campuses</li>
                    <li>Increasing access to involvement opportunities</li>
                    <li>Amplifying student voices</li>
                    <li>Promoting & fundraising for scholarship opportunities that incentivize service & leadership.</li>
                </ol>
                <p style="text-align: left; margin: 0 0 10px;"><b>Application Link: </b>''' + orgs[info[x+1]][3] + '''</p>
            </td>
        </tr>
        '''
    else:
        text += """
            """ + orgs[info[x+1]][1] + """
            Purpose: """ + orgs[info[x+1]][2] + """
            Application Link: """ + orgs[info[x+1]][3] + """
            """
        html += '''
            <tr>
                <td aria-hidden="true" height="40" style="font-size: 0px; line-height: 0px;">&nbsp;</td>
            </tr>
            <tr>
                <td>
                    <img src="''' + orgs[info[x+1]][0] + '''" style="width: 100%; max-width: 600px;">
                </td>
            </tr>
            <tr>
                <td style="padding: 10px 20px 20px 20px; font-family:sans-serif; background-color: #eee; width: 100%; max-width:
                    <h1 style="margin: 0 0 10px; text-align: left; font-family:Georgia, serif;">''' + orgs[info[x+1]][1] + '''</
                    <p style="text-align: left;"><b>Purpose: </b>''' + orgs[info[x+1]][2] + '''</p>
                    <p style="text-align: left; margin: 0 0 10px;"><b>Application Link: </b>''' + orgs[info[x+1]][3] + '''</p>
                </td>
            </tr>
            '''
html += """
                    <tr>
                        <td aria-hidden="true" height="40" style="font-size: 0px; line-height: 0px;">&nbsp;</td>
                    </tr>
                </tbody>
            </table>
        </body>
    </html>
    """
pPart = MIMEText(text, "plain")
hPart = MIMEText(html, "html")
message.attach(pPart)
message.attach(hPart)

# Sendoff
context = ssl.create_default_context()
with smtplib.SMTP_SSL("smtp.gmail.com", 465, context=context) as server:
    server.login(sender, password)
    server.sendmail(sender, receiver, message.as_string())
    print("Email Sent!")
```

Implementation

Implementation actually went pretty dang well all things considered! There were a few small syntax errors here and there and a few small changes to be made, but all in all it actually ran really smoothly once I figured out how to get things running with the email function. The biggest challenges that came up with the initial implementation were:

1. When I organized the dictionary to find the top three committees, at the last second I changed it from `orgs.items` to `orgs.keys` thinking that was the correct way to go about it. Definitely wasn't. So I fixed that. I also had some trouble at the beginning because it tried sending the email to 'hurd', but that was just because I put a 1 instead of a 0 for the index for the important info.
2. I forgot to actually add the code that attached the committees to the email. That was a problem.
3. The emails looked like hot garbage.

But I got all those fixed pretty quickly. Really, most my time with this program past the design phase was just getting the emails to look nice. I based them off of the weekly newsletter USU sends out, and so while they're not perfect I think they look pretty nice. They're missing the links and the images still, but that's nothing I can't fix whenever I get those from the others.

With all of that, the program is, as of August 14th, as done as it *can* be at the moment. I still don't have a housing, I still don't have a proper email to send from, I still don't have a finished survey, I still don't have the application links, and I don't have the pictures to go with each committee. Everything should be really easy to implement within the program when the time comes, but at the moment it's done! The only thing I have left I really want to do is clean up some of the clutter. For example, I realized I just copy/pasted the HTML for the committees, and I think I can come up with a much more efficient way to do that. My next commit will hopefully have that version, but I'm putting this up now so I can have something to fall back on in case I really screw things up. I'll add an update when that's done about how that went. I'll update all the stuff in this as well when that comes around.

Update: Got it! With a staggering *34%* decrease in line count, the program runs much more efficiently and looks far nicer. Rather than sticking the parts to append to the email in the dictionary, the dictionaries only hold the *key* information such as photo, name, description, and application link. Then the email composition section just pulls that information inside the text that's being appended. The only thing that really needed to be changed was to add an exception for PCab because there's so much more in the 'Purpose' section, and I had to add the code to ensure if someone doesn't know their council that it doesn't get recommended. Even if the *very* unlikely situation happens where they somehow select a college and the option saying they don't know what college they're in. I swear, that question is the *only* one that can break this program, but I took care of it!

Testing

Testing was done using a CSV file I manually exported. I've just been using that over and over to run the program and ensure the emails worked. The emails were just sent from and to my college email, and that's worked very nicely! Once the program is fully done, I'm going to have the rest of PCab use it just to ensure there's no issues. I've also been brainstorming any ways the user could screw this up, but thankfully since it's a multiple choice quiz there isn't very many. The only way I'd found thus far is that selecting multiple colleges could be screwy with the CSV file, but I fixed that with what might not be the most *elegant* solution, but it worked nonetheless. :)