'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. Constantly run in the background
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 this is the first area where I run into the problem of *'known unknowns'*. As of now, I have no idea how exactly we're getting this program up and working. I can write it and get it to work flawlessly, but it doesn't really matter if it's not in some location where I can export the responses to. *Wherever* this program gets put, whether it's on a server or a Virtual Machine or if I just have to let this run on my computer or just *whatever* happens, I need this to run in the background all the time so it can run whenever the response CSV pops up. After some searching, I hope this provides me with the information I need to get this working.

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

1. Constantly run in the background ✓ *Needs to be run with `pythonw.exe`
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

Main

```
If the namespace is the main function,
    # File Processing
    Set a variable hold the file, which is returned from 'getFile()'
    Call 'lineProcessor()' and send it the file
```

GetFile

```
getFile() is defined
    """ This function locates the CSV file with 'BRQ' in it, and pulls from that. """
    Set a false boolean
    While said boolean is false,
        Attempt to do the following:
            -Set the file path string using the 'glob' library.
            -Set a file object to the opened file with the file path
            -Set the boolean to true
        If it doesn't work:
            -Let the program rest for some amount of time until next run.
    Return the file object
```

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
            --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

# 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 i the answer,
        --Add points to 'academic' and 'mentor'
Index 9:
    -If 1 is the answer,
        --===not sure what to add 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:
    -Add points from 'service', 'planner', and 'academic'
    -In addition, add a small bump if 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:
```

```
    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
    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 """
```

Python Code

Implementation

Testing