The Problem

Outline

We must make a program that can be used to plan what food will be bought for a class party. This will be based on student preference, the program will take student input and then generate a list of food that the teacher could buy.

Requirements

- Input mechanism for students to enter their lolly and snack preferences.
- Rating system for preferences to prioritise the selection process.
- Data analysis feature to suggest a top list of lollies and snacks.
- Data backup to a text file, ensuring the information is stored appropriately and can be accessed later.
- User-friendly interface to facilitate easy input and interaction with the program.

Additional Goals

- Easy to use textual user interface
- 1 to 5 scale to choose preference
- Limit results to a certain number of top snacks
- Sorting based on preference
- Backup data to JSON (JavaScript Object Notation) file
- Read backup file
- High user friendliness

Parameters of the problem

- Python must be used.
- I have plenty of experience writing Python as well as experience in other programming languages.
- Must be all my own work.
- Time constraints, this task is due in 13 days on the 25th of March. I have 5 lessons until it is due so it is likely that I will need to do some work at home.

Project Diary

Along with this project diary I also have the GitHub repository commit history at github.com which has line by line code changes per commit.

Date	Activity	Challenges
1/03	Assessment task received	
5/03	Begin defining parameters of the problem	
7/03	Start writing flowcharts	
8/03	Turning flowcharts into pseudocode	
11/03	Initialize GitHub repository here	
12/03	Starting python code with placeholders	
13/03	Start writing modules, removing placeholders	
18/03	Writing modules	
19/03	Replacing placeholders	
20/03	Switching to JSON, adding personal info disclaimer	Attempted to write my own data encoding scheme, but I decided to use JSON as it is a well-established method of saving data to text files for future use.
21/03	Begin implementing allergies	
22/03	Finish all code and allergy functionality, Using Visual Studio Code trunk extension for code formatting	Running out of time and python cannot sort dictionaries
23/03	Folio writing	Still running out of time

Pseudocode and Flowcharts

Pseudocode

main.py

```
BEGIN main
      read party_foods.json
      read allergens.json
      DISPLAY privacy disclaimer

      WHILE true DO
            INPUT command
            IF command IS new planner THEN
                  perform class survey
            ELSE IF command IS suggest snacks THEN
                  analyse data
            ELSE IF command IS save backup THEN
                  INPUT filename
                  OUTPUT filename.json
            ELSE IF command IS load backup THEN
                  find files with json extension
                  INPUT file to open
                  open file
            ELSE IF command IS quit THEN
                  QUIT
      ENDWHILE
END main
```

easyInput.py

```
BEGIN binaryInput
      REPEAT
            INPUT response
      UNTIL response is valid answer

      IF response is a yes response
            RETURN true
      ELIF response is a no response
            RETURN false
      ENDIF
END binaryInput

BEGIN listInput
      DISPLAY valid answers

      REPEAT
      INPUT response
      UNTIL response is valid

      RETURN response
END listInput

BEGIN multiInput
```

```
      INPUT response
      remove spaces from input
      make input lower case
      split input by separate character

      RETURN only valid responses
END multiInput
```

fileFinder.py

```
BEGIN findFiles
      GET files and folders in directory

      FOR item IN directory
         IF keyword IN item
              RETURN item
END findFiles
```

fileReadWrite.py

```
BEGIN writeFile
      encode object to JSON
      write JSON to file
END writeFile

BEGIN readFile
      open and read JSON file
      decode JSON data to object
      RETURN object
END readFile
```

survey.py

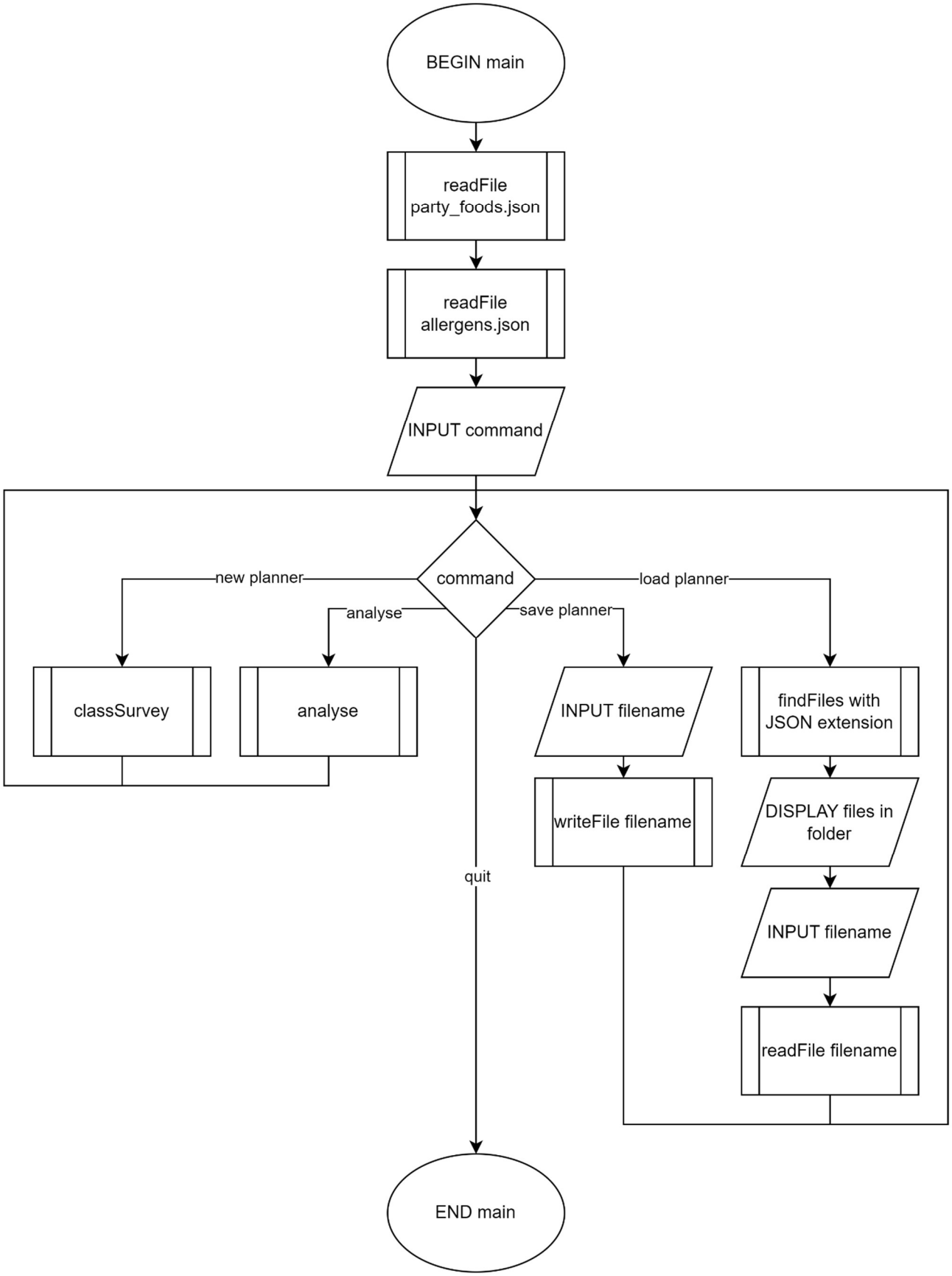
```
BEGIN classSurvey
     format snack options
     WHILE true
          INPUT name
          IF name IS exit
          BREAK
          DISPLAY welcome message and snack options
          FOR pref IN preferences
               REPEAT
                    INPUT snack choice
               UNTIL choice is valid
               add choice to list of choices
          ENDFOR
          add choices to student results
          INPUT allergies
          add allergies to student results
          add student results to class results
     ENDWHILE
     RETURN class results
END survey
```

analysis.py

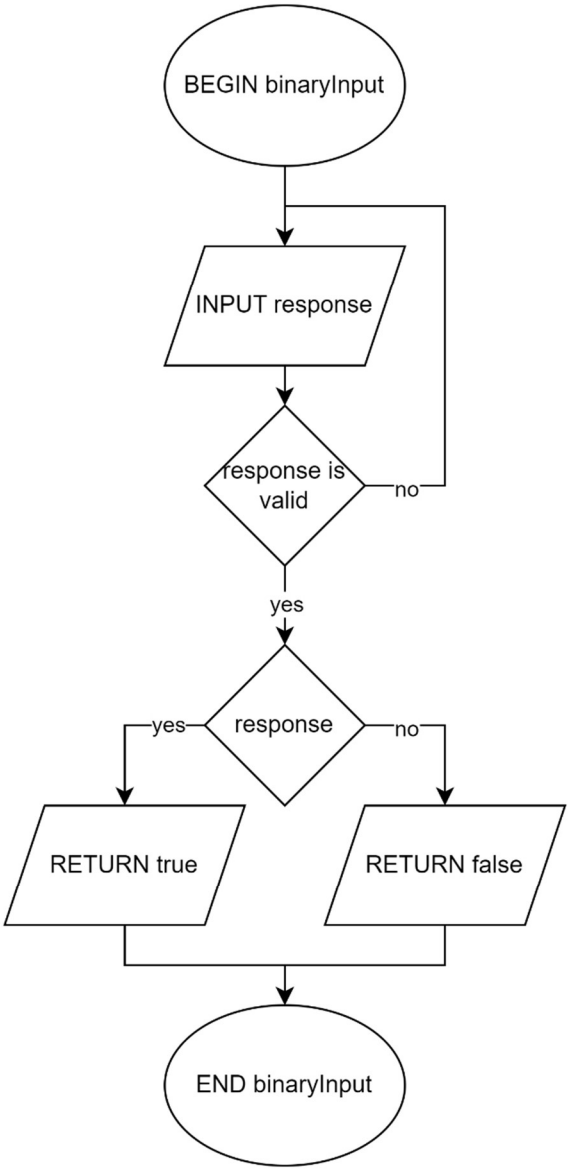
```
BEGIN analysis
     FOR student IN surveyResults
          FOR preference IN student picks
               add preference to food rankings
          ENDFOR
          FOR allergy IN student allergies
               add student to list of allergies
          ENDFOR
     ENDFOR
     remove any food with a score of 0 or less from rankings
     sort rankings by score
     match foods left in rankings with students allergic to it
     FOR food, vote IN rankings
          DISPLAY food = vote
     ENDFOR
     IF allergies are present THEN
          DISPLAY food and its allergic students

     RETURN rankings
END analysis
```

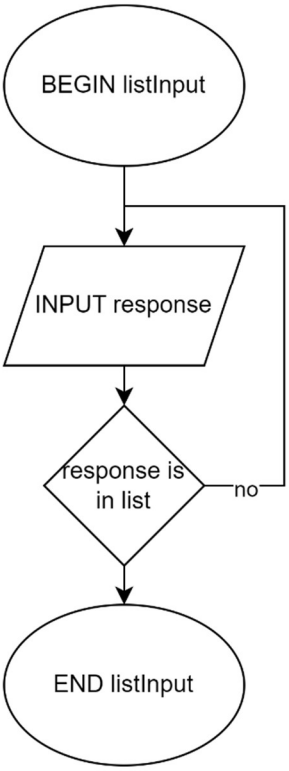
Flowcharts

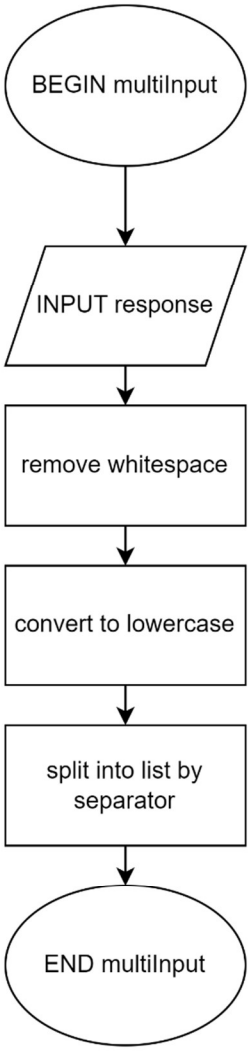
main.py



easyInput.py

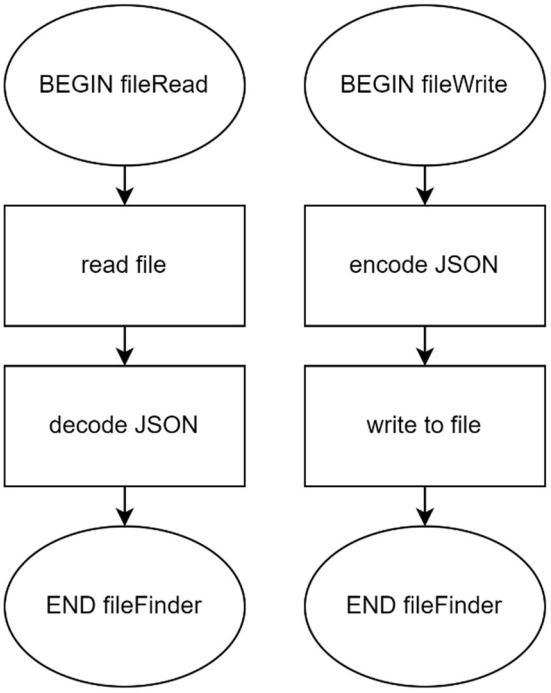




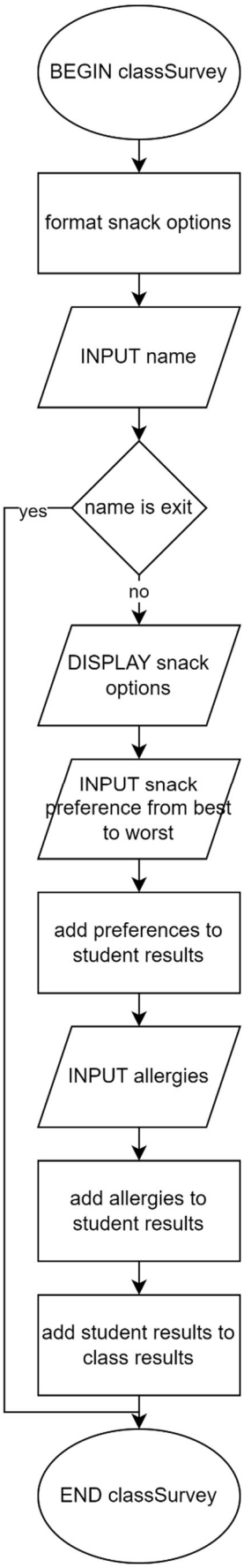


fileFinder.py

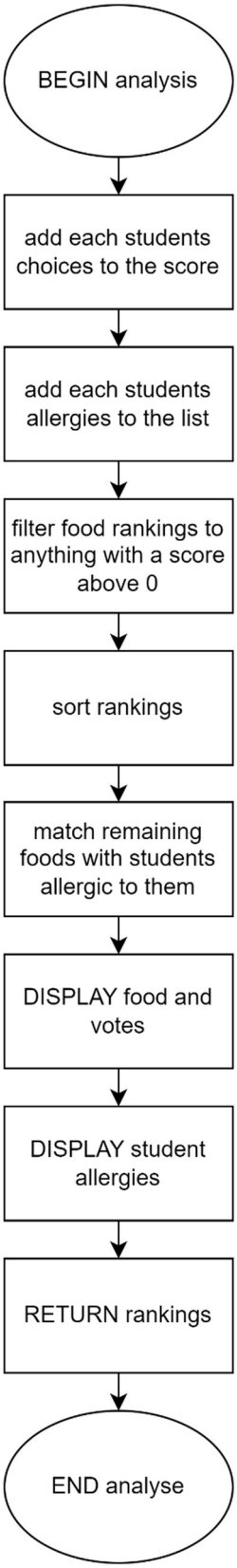
fileReadWrite.py



survey.py


BEGIN classSurvey
format snack options
INPUT name
name is exit
yes
no
DISPLAY snack options
INPUT snack preference from best to worst
add preferences to student results
INPUT allergies
add allergies to student results
add student results to class results
END classSurvey

analyse.py


BEGIN analysis
add each students choices to the score
add each students allergies to the list
filter food rankings to anything with a score above 0
sort rankings
match remaining foods with students allergic to them
DISPLAY food and votes
DISPLAY student allergies
RETURN rankings
END analyse

Testing

To ensure that my program worked as intended and without bugs, I used it myself a couple of times, which allowed me to decide if there were any parts that needed to be improved for a better user experience. During my testing, I attempted to induce every condition that could make the program fail, such as in cases where the user submits a blank or invalid input. My solution to most cases like this is to get another input from the user and hope that it is valid, although in some cases (see the multiInput function in the easyInput module) the program will simply skip invalid responses.

During this testing, encountered multiple points in my program where exceptions would be thrown, which I had to fix to ensure the best user experience possible.

Evaluation

In my program, I have achieved all the goals I set out for myself at the beginning which included a high level of user friendliness, JSON backups and all the sorting and filtering. It required some creative thinking to work around certain issues. Some of the challenges I faced along the way included time constraints which I solved by simplifying some of my goals and python limitations which were solved by using external libraries such as JSON and the os library. I kept my code readable by regularly including comments and spacing to group sections of code with different functionality. I ensured that my code was optimized and performed well by using python's built-in functions such as sorted() and by placing my code into separate files that work as modules and defined functions to avoid having to write the same pieces of code multiple times. By using modules, I also gain the benefit of being able to reuse to save time when I write code in the future. Along with this, I made use of industry standard programs such as Visual Studio Code as my python IDE, which includes many features that help save time, and GitHub for version control, which is better than using something such as OneDrive since it allows easy uploading of code while maintaining extensive version history with the ability to see changes made line by line in my code.

In conclusion, I believe my program met and exceeded the success criteria while maintaining a high level of readability and optimization.