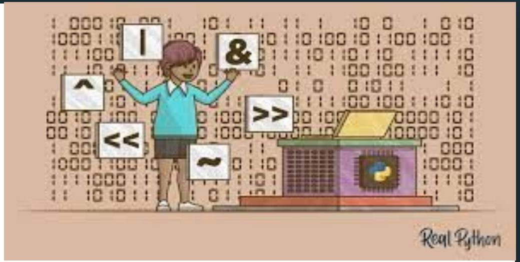
The Dr. Perlman Dell Preternship Project

The Problem

- A majority of people work 9-5 jobs, but would be very open to meeting outside of that one window
- Children and extracurricular activities such as sports, musicals, etc.
- Engagements during lunch time
- A lot of people would be happy to meet in the evening or early morning. Sometimes availability is a lot easier outside of the typical 9-5 workday
- Our problem becomes even more relevant when taking into consideration different time zones
- It is very difficult to schedule a meeting, for example, with someone in Asia and North America

General Overview



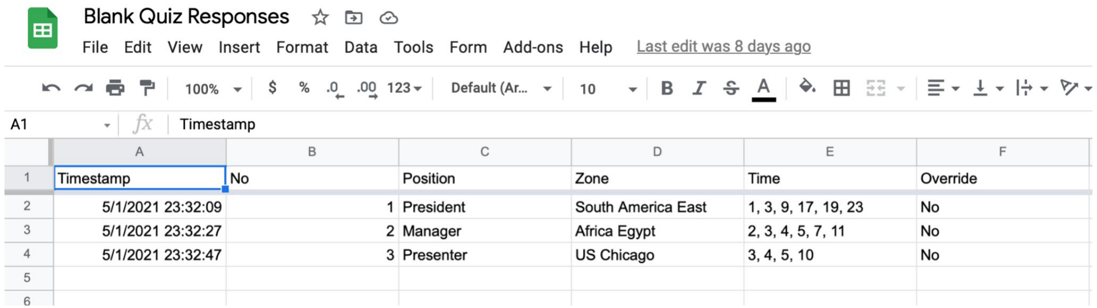
- We created a schedule optimizer to find the most optimum time for a particular group to host a meeting
- All the employees at the company have to do is fill out a Google Form which asks for company position as well as his or her availability for meeting times
- Our program then pulls the CSV formatted data from the Google Form and converts it into JSON data
- Next, our program utilizes a bitset with different weights based on the position of the person in the company
- Finally, we look for the time which has the most overlaps (including the different weights) and output the top 3 most optimal times to meet where both the most important people and majority can attend the meeting

Week 1 - Repository & Setup

- Our first step was to create a GitHub repository so that we could easily collaborate from multiple machines
- After meeting with our Project Manager regarding our initial project proposal, we changed some details regarding how we would implement our ideas
    - Decided to go with bitsets when representing availability over an array of ints or bools
- Created a CSV to JSON script
- Wrote a function that converts any timezone to standard time by returning the time difference
- By the end of Week 1, we had two different ideas on how we could get the raw data from Google Sheets so we could convert it to a JSON file
    - Google API vs. publishing the downloadable CSV version of the Google Sheet

Week 1 Deliverable

Sheets Results



	A	B	C	D	E	F
1	Timestamp	No	Position	Zone	Time	Override
2	5/1/2021 23:32:09	1	President	South America East	1, 3, 9, 17, 19, 23	No
3	5/1/2021 23:32:27	2	Manager	Africa Egypt	2, 3, 4, 5, 7, 11	No
4	5/1/2021 23:32:47	3	Presenter	US Chicago	3, 4, 5, 10	No
5						
6						

CSV

```
Timestamp,No,Position,Zone,Time,Override
5/1/2021 23:32:09,1,President,South America East,"1, 3, 9, 17, 19, 23",No
5/1/2021 23:32:27,2,Manager,Africa Egypt,"2, 3, 4, 5, 7, 11",No
5/1/2021 23:32:47,3,Presenter,US Chicago,"3, 4, 5, 10",No
```

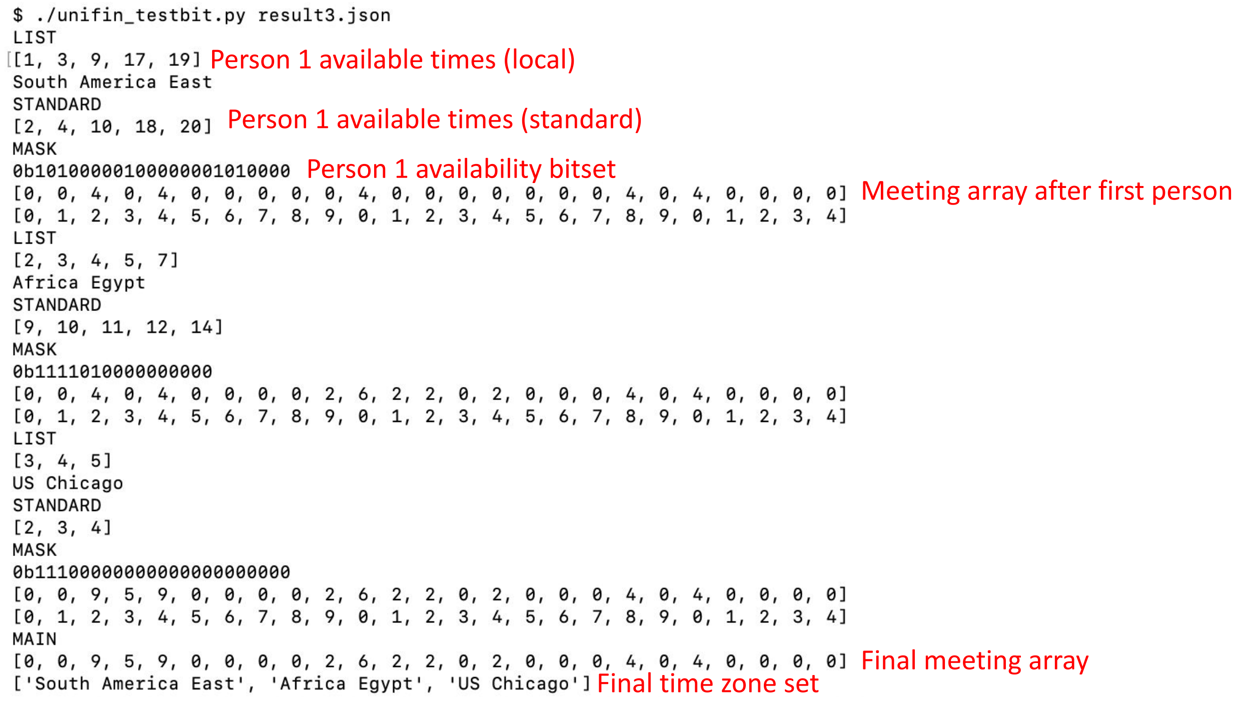
JSON

```
{
    "1": {
        "Timestamp": "5/1/2021 23:32:09",
        "No": "1",
        "Position": "President",
        "Zone": "South America East",
        "Time": "1, 3, 9, 17, 19, 23",
        "Override": "No"
    },
    "2": {
        "Timestamp": "5/1/2021 23:32:27",
        "No": "2",
        "Position": "Manager",
        "Zone": "Africa Egypt",
        "Time": "2, 3, 4, 5, 7, 11",
        "Override": "No"
    },
    "3": {
        "Timestamp": "5/1/2021 23:32:47",
        "No": "3",
        "Position": "Presenter",
        "Zone": "US Chicago",
        "Time": "3, 4, 5, 10",
        "Override": "No"
    }
}
```

Week 2 - Meeting Array

- Once we had a properly formatted JSON, we used for loops to go through each key/value person and create a bitset array of their available meeting times
- Before creating the array, their times were formatted into a list and converted to EST
- If there was no override case, the bitsets searched through with logical operators and the member's position weight was appended to the single meeting availability array
- In the case of an override, the bitset is also compared with the override member's bitset
- The final printed array will indicate the best meeting times as the array index of the highest value corresponding to that respective time

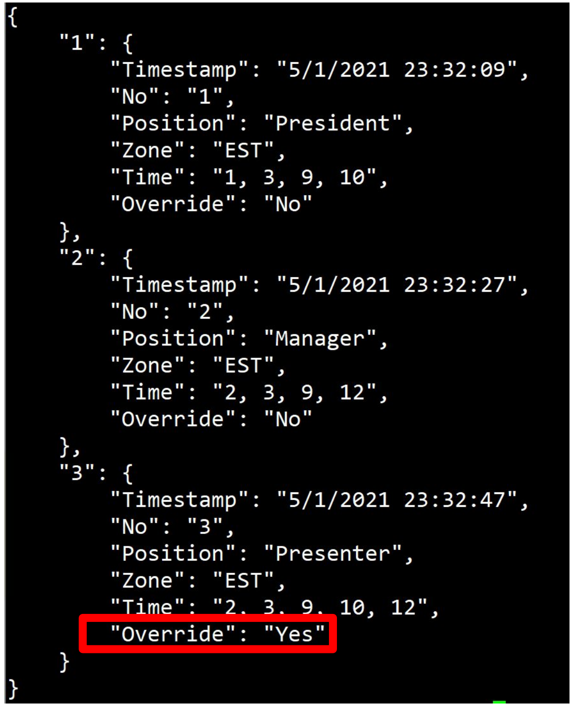
Week 2 Deliverable



```
$ ./unifin_testbit.py result3.json
LIST
[1, 3, 9, 17, 19] Person 1 available times (local)
South America East
STANDARD
[2, 4, 10, 18, 20]  Person 1 available times (standard)
MASK
0b101000001000000001010000  Person 1 availability bitset
[0, 0, 4, 0, 4, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 4, 0, 4, 0, 0, 0, 0] Meeting array after first person
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4]
LIST
[2, 3, 4, 5, 7]
Africa Egypt
STANDARD
[9, 10, 11, 12, 14]
MASK
0b1111010000000000
[0, 0, 4, 0, 4, 0, 0, 0, 0, 2, 6, 2, 2, 0, 2, 0, 0, 0, 4, 0, 4, 0, 0, 0, 0]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4]
LIST
[3, 4, 5]
US Chicago
STANDARD
[2, 3, 4]
MASK
0b111000000000000000000000
[0, 0, 9, 5, 9, 0, 0, 0, 0, 2, 6, 2, 2, 0, 2, 0, 0, 0, 4, 0, 4, 0, 0, 0, 0]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4]
MAIN
[0, 0, 9, 5, 9, 0, 0, 0, 0, 2, 6, 2, 2, 0, 2, 0, 0, 0, 4, 0, 4, 0, 0, 0, 0] Final meeting array
['South America East', 'Africa Egypt', 'US Chicago'] Final time zone set
```

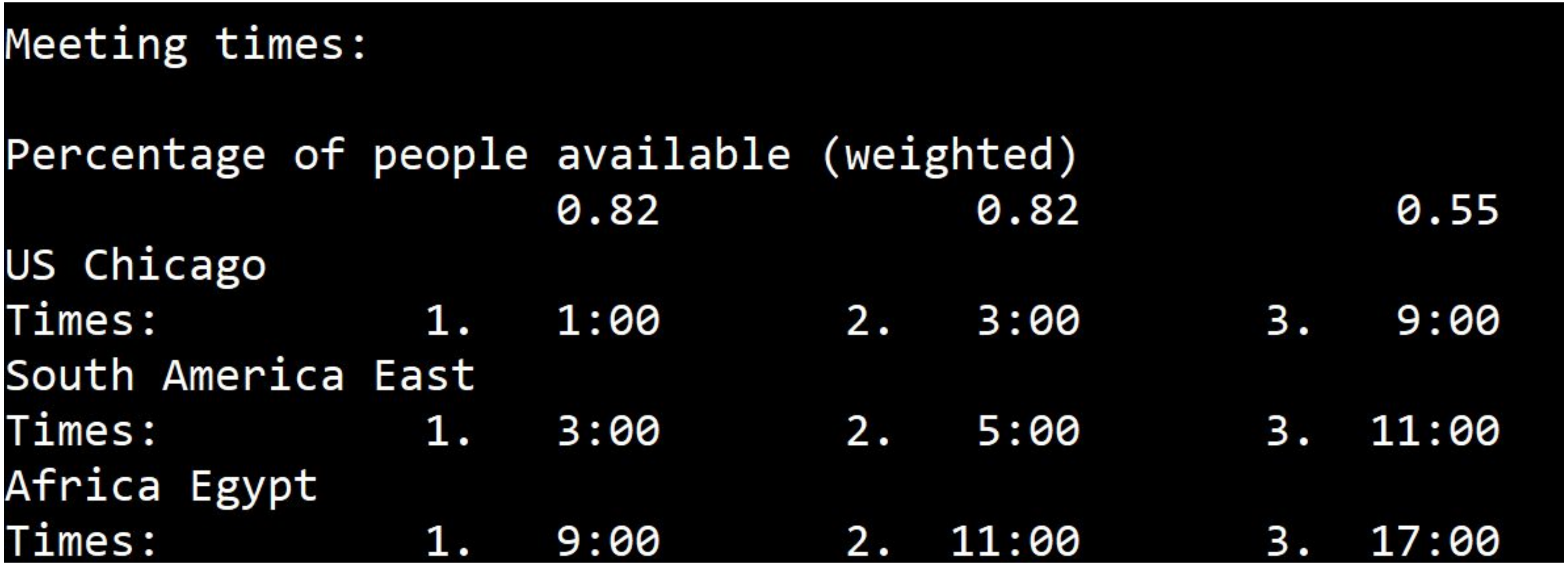
Week 3 - Override and Display

- Implemented an override option using a for loop to find the meeting participant that is required to be at the meeting (if any)
  - The new "weighted array" would be based on the availability bitset of that person
- Finding up to three optimal times based on a specified percentage threshold
  - Locating maximum values in the weighted array in order
- Formatting and displaying the times
  - Converting times to the applicable time zones
  - Displaying the "weighted" percentage of people available for each time



```
{
    "1": {
        "Timestamp": "5/1/2021 23:32:09",
        "No": "1",
        "Position": "President",
        "Zone": "EST",
        "Time": "1, 3, 9, 10",
        "Override": "No"
    },
    "2": {
        "Timestamp": "5/1/2021 23:32:27",
        "No": "2",
        "Position": "Manager",
        "Zone": "EST",
        "Time": "2, 3, 9, 12",
        "Override": "No"
    },
    "3": {
        "Timestamp": "5/1/2021 23:32:47",
        "No": "3",
        "Position": "Presenter",
        "Zone": "EST",
        "Time": "2, 3, 9, 10, 12",
        "Override": "Yes"
    }
}
```

Week 3 Deliverable


```
Meeting times:

Percentage of people available (weighted)
                    0.82                0.82                0.55
US Chicago
Times:          1.   1:00       2.   3:00       3.   9:00
South America East
Times:          1.   3:00       2.   5:00       3.  11:00
Africa Egypt
Times:          1.   9:00       2.  11:00       3.  17:00
```

Our Product!

Procedure:

Email sent out to invited members → Members complete the Google Form → Google Forms collects responses into a Google Sheets CSV → setup.sh runs the python scripts to get a meeting

setup.sh:

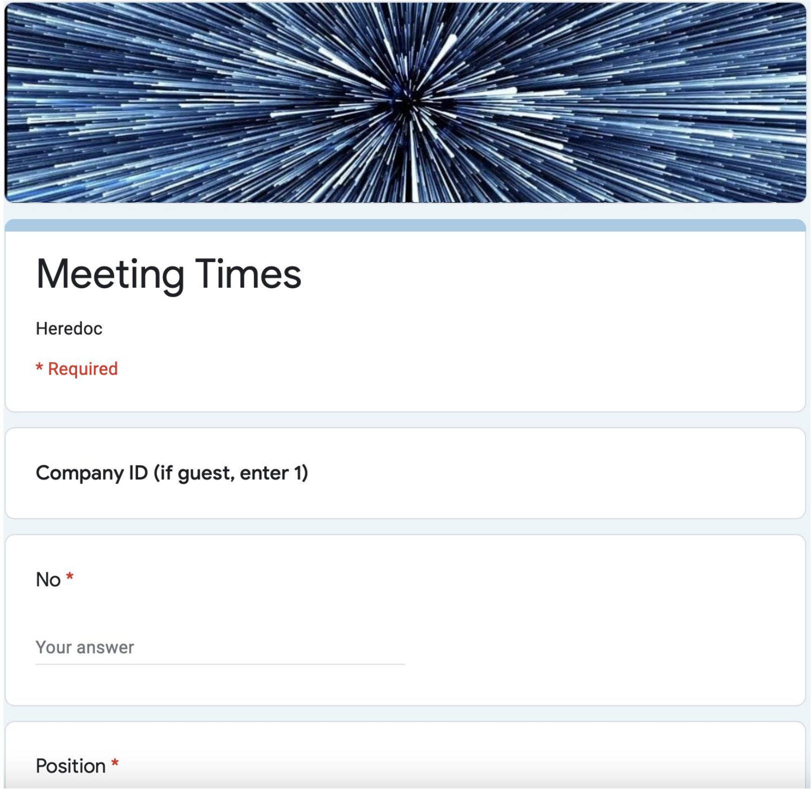
GStoCSV.py curls the CSV into the terminal as a file data.csv → csvTOjson.py converts data.csv into result3.json → final_meet.py uses the .json file to find and display meeting options

How we applied what we learned from Data Structures

- Types of data structures
  - Hash Table: set of time zones, dictionaries of people from the JSON file
  - Static Array: array of length 25 to store the cumulative weights of availabilities for each time
  - Bit Array: bitset saved for the override member, individual bitsets for each member’s available standardized meeting times; this format is memory efficient and is easy to use to compare using logic operators
- Algorithms
  - Finding the three maximum values in the availability array in order
  - Comparing availability percentage with the threshold to determine times that work

Check your inbox!

- Fill out the emailed form to see your response be included in the live demonstration
- If Override selected as “Yes,” only times that work in your schedule will be displayed
- For this demonstration, please enter 7 for the No field



Meeting Times

Heredoc

* Required

Company ID (if guest, enter 1)

No *

Your answer

Position *