Introduction to Programming Group Work: Running Dinner

by Jonas Nicolai, Justin Schütte, Marco Schmiederer and Paul Pochanke
(49361, 49708, 51241 and 50508)

Necessary Links:

GitHub Repo: Github Running Dinner Report

Main Code: Github Running Dinner Main Code

Streamlit Interface: Streamlit Interface

Login credentials:

Mail address: introtorunningprogramming@gmail.com

Password: JoJuMaPa351

1. General Information

For the group work, our group has programmed a Running Dinner Application that can be used in every city around the world.

What is a running dinner and how does it work?

The objective of the running dinner is that groups of people can get to know each other over having dinner together. Participants can register together with a partner for the event and will be allocated in groups of 6 persons (3 pairs). The groups will then have a three-course dinner, where every course will be prepared by one pair of the group at their home. After the last course, all groups will meet at a final destination where they can get to know each other and have a party.

2. Streamlit

The Running dinner application works over a Streamlit Interface which can be found on the following website: Streamlit Interface Running Dinner.

Step 1 - Distribute Survey link

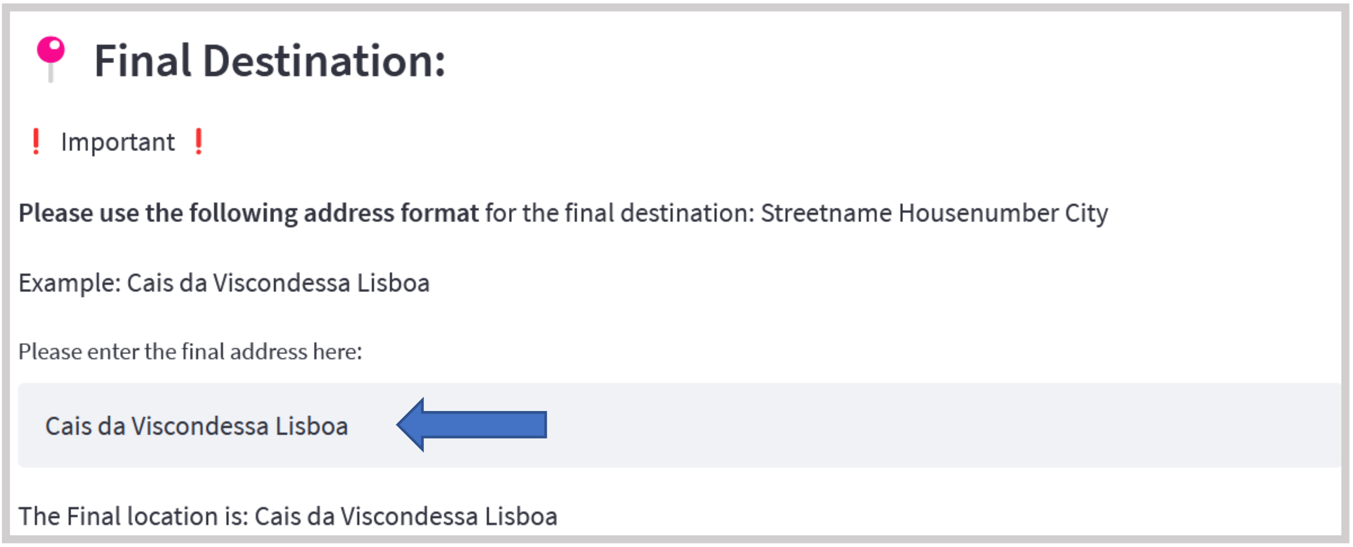
The first step to creating your own dinner is to use the Survey link on the website and share it with possible participants. The survey is linked to a google spreadsheet where the information of the participants is stored.

Step 2 – Clear Spreadsheet

This step only needs to be considered if this is not the first time you are hosting a running dinner. The purpose of this is to clear the existing spreadsheet of old survey submissions. For this, you need to enter the Spreadsheet, select all entries in the sheet “Answers” and delete them.

Step 3 – Insert Final Destination

To then allocate the members to their group and let the algorithm create the plan for the Running Dinner, the organizers need to insert the address of the final location, which is used to allocate the teams in the best possible way. It is important that the address is formatted exactly like the example address so that the API can certainly locate the address (see below).



Final Destination:

❗ Important ❗

Please use the following address format for the final destination: Streetname Housenumber City

Example: Cais da Viscondessa Lisboa

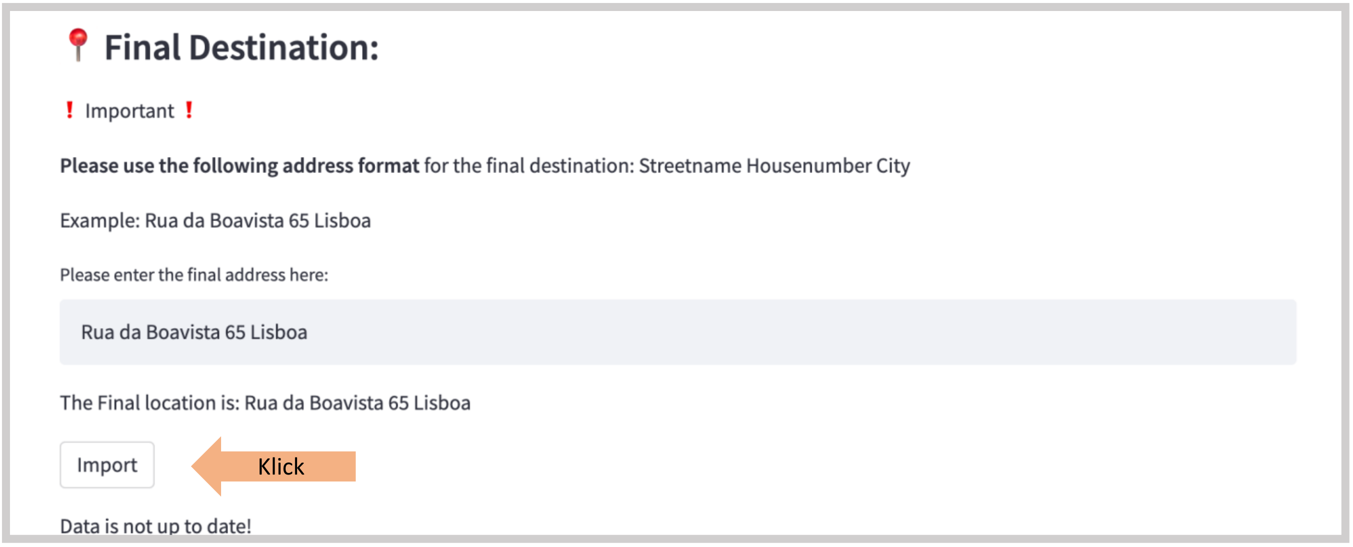
Please enter the final address here:

Cais da Viscondessa Lisboa

The Final location is: Cais da Viscondessa Lisboa

Step 4 – Import Data from Spreadsheet

I. Press Import Button



Final Destination:

❗ Important ❗

Please use the following address format for the final destination: Streetname Housenumber City

Example: Rua da Boavista 65 Lisboa

Please enter the final address here:

Rua da Boavista 65 Lisboa

The Final location is: Rua da Boavista 65 Lisboa

Data is not up to date!

In order to import the survey data from the google spreadsheet, you need to click on the "Import" button under the final location text field. This will run the python code in the backend. You know that the code is running whenever it says, so it is in the upper right corner (run time depends on data size) and it is finished whenever it shows you the text "Data is up to date!". The code uses several APIs like the Google Authenticator APIs and GeoPy APIs. This then runs through a Running Dinner algorithm to create the final dataset. For this, the two classes' "Data" and "algorithm" were created. The final dataset is then used for further analysis in the interface.

II. Class Data

The class "Data" is a combination of three Google Spreadsheet APIs authentication python modules (google_auth_oauthlib. As InstalledAppFlow, google.auth.transport.requests as Request, googleapiclient.discovery as build). All of them are combined in the class "Data". This class is then constructed around those three definitions (See GitHub).

The first " __init__" allows to initialize the attributes of this class which are "SCOPES", "SPREADSHEET_ID", and "DATA_TO_PULL". All of them are needed to correctly get access to the spreadsheet and to get the right sheet within the spreadsheet.

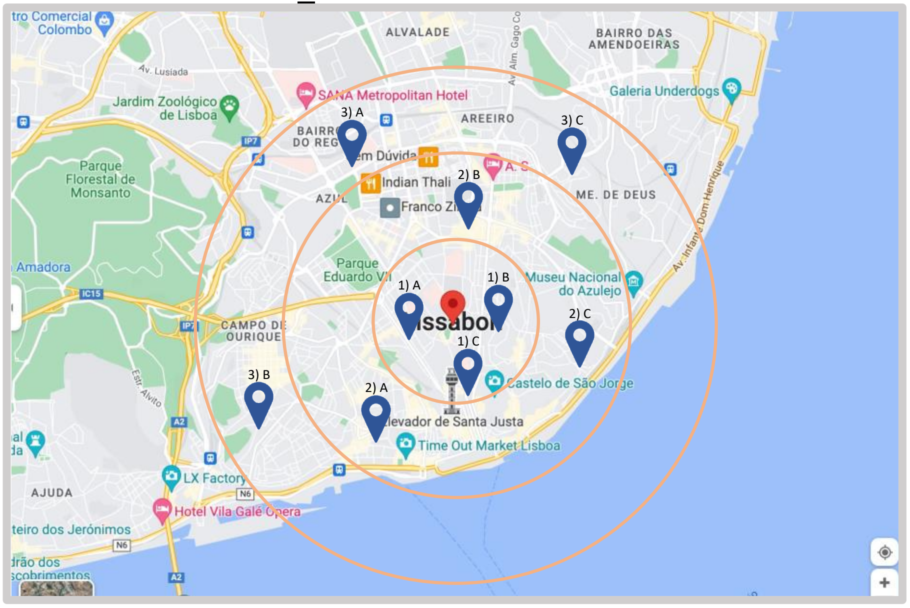
The other two definitions are nested within each other. However, the main definition is the "pull_sheet_data" definition. It first requests an API check with the definition "gsheet_api_check" and will create new credentials if needed. After the access is granted, it will pull the specific sheet with all the data and stores it into a dictionary called "df".

III. Class algorithm

The class "algorithm" has four main functions (See GitHub). First, with the definition "get_data" it initializes the class "Data" and imports the dictionary "df" into this class. Combined with the definition "start", which takes as input the final address from step 3, the definition "geo" is calculating each team's distance towards the final destination. This is done by the "GeoPy" API that, in a first step, takes an address and transforms this then into latitudes and longitudes. With those, the APIs function "great_circle" calculates in a next step the great-circle distance, which is the shortest distance between two points on the surface of a sphere. All this new data is then appended to the dictionary and transformed into a pandas dataframe. This new dataframe "dataT" is then called by the definition "team". The purpose of this definition is to clean the data first and then randomly create final teams of three where the distance to the final destination is the key.

First, to clean the data, the addresses where ever the GeoPy API couldn't create a distance are stored in a different data frame called "wrong_address". In those cases, the teams will get a placeholder latitude of -89.999 and longitude of -179.999. Since the running dinner exists of three courses (Appetizer, Main Course and Dessert), the length of the dataset also needs to be a multiple of 3. As a solution, whenever this is not the case, the last one or two entries are extracted and stored into the dataset "lostData".

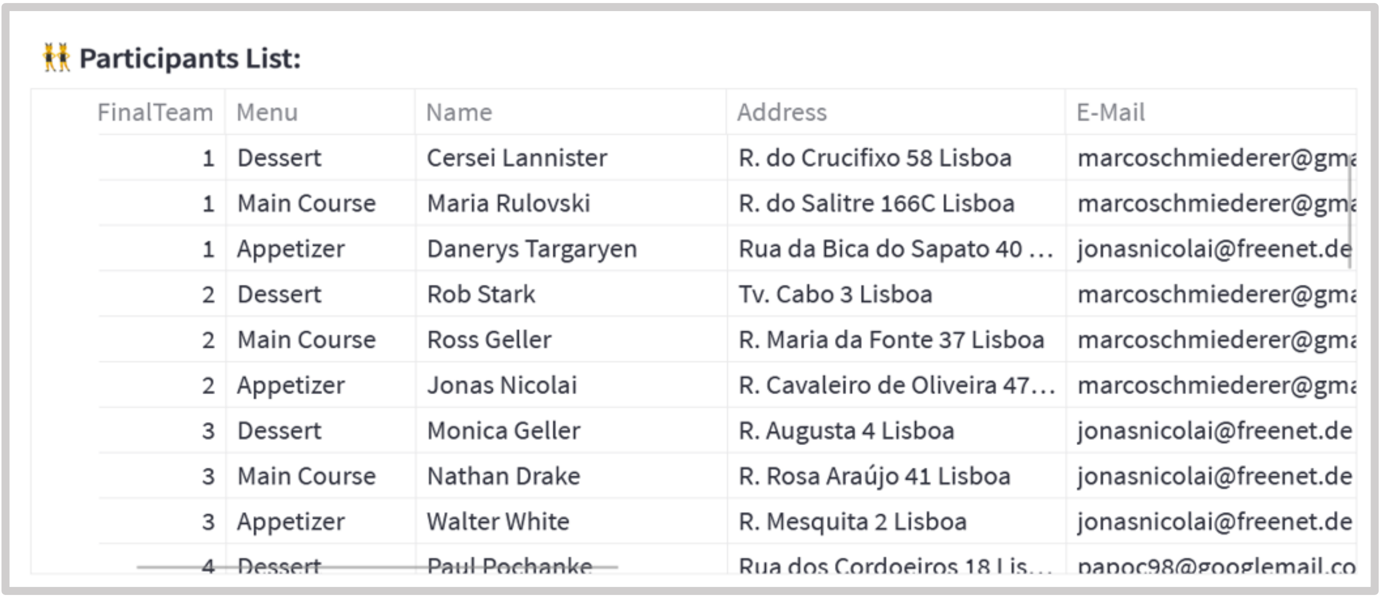
Next, with this cleaned dataset, the random running dinner algorithm distributes the teams into its final teams, where each final team consists of three teams. First, the teams are put into three equal-sized buckets based on distance, calculated by the pandas function "qcut" which is a quantile-based discretization function. The bucket with the shortest distance represents the teams that are going to do the dessert (label=1), the middle bucket the main course (label=2) and the bucket furthest away will do the appetizer (label=3). In a second step, it randomly picks one team out of each bucket. Those will form one final team. After they were picked, they will be removed from the buckets. This will be executed till all buckets are empty. In a final step, all three datasets dataT (main dataset), lostData, and wrong_data are put back together and returned as dataT_concat to the interface code.



Step 5 – Main Interface

I. Participants

Here you can see the full participants list with all the necessary information. The Running Dinner Program already allocated all Participants from this list to their final teams with a random algorithm, which you can find in the first column. Teams were allocated according to their distance to the Final Destination, meaning that the ones with the largest distance from the Final Destination prepare the appetizer, the ones in the middle prepare the main course, and the ones closest to the Final Destination, the dessert. Consequently, all teams of the Running Dinner are close to each other after the dessert, making it easy to meet for a drink afterward. Additionally, all necessary information (address, e-mail, phone number) of both team members is included in the participants' list.



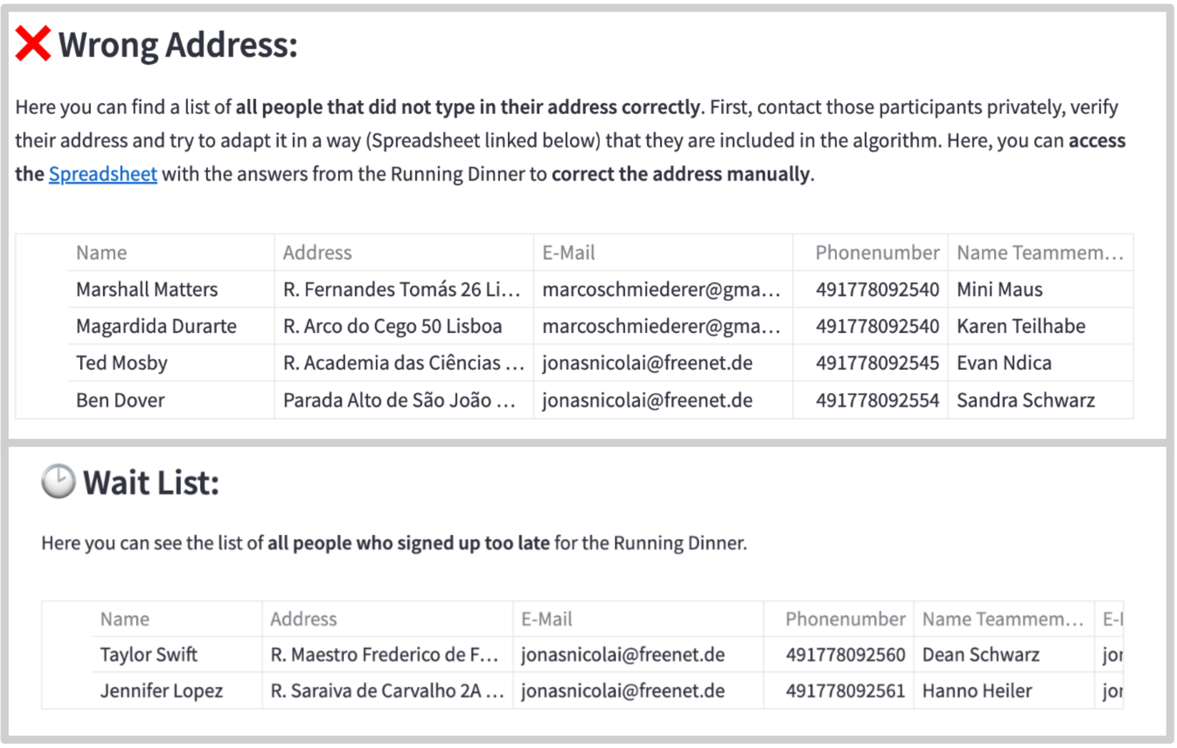
Participants List:

FinalTeam	Menu	Name	Address	E-Mail
1	Dessert	Cersei Lannister	R. do Crucifixo 58 Lisboa	marcoschmiederer@gma
1	Main Course	Maria Rulovski	R. do Salitre 166C Lisboa	marcoschmiederer@gma
1	Appetizer	Danerys Targaryen	Rua da Bica do Sapato 40 …	jonasnicolai@freenet.de
2	Dessert	Rob Stark	Tv. Cabo 3 Lisboa	marcoschmiederer@gma
2	Main Course	Ross Geller	R. Maria da Fonte 37 Lisboa	marcoschmiederer@gma
2	Appetizer	Jonas Nicolai	R. Cavaleiro de Oliveira 47…	marcoschmiederer@gma
3	Dessert	Monica Geller	R. Augusta 4 Lisboa	jonasnicolai@freenet.de
3	Main Course	Nathan Drake	R. Rosa Araújo 41 Lisboa	jonasnicolai@freenet.de
3	Appetizer	Walter White	R. Mesquita 2 Lisboa	jonasnicolai@freenet.de
4	Dessert	Paul Pochanke	Rua dos Cordoeiros 18 Lis…	papoc98@googlemail.co

II. Data Problems

There can be two data problems within the running dinner program.

The first one covers all teams that **submitted too late**. A team submitted too late whenever a final team of 3 cannot be built. This is done with a modulo condition within the code. Hence, if this is the case, it will filter out the last survey submissions and store them into the dataset “lostData” and will display it under the header Wait List. The second data problem can be that the **GeoPy API cannot correctly read the Address** that is submitted by the participants via the survey. Possible solutions have to first contact those participants privately, verify their address and try to adapt it in a way (Spreadsheet linked below) that they are included in the algorithm and then import the data again. Second, if they don’t respond or the problem cannot be solved, they won’t be invited to the running dinner.



❌ Wrong Address:

Here you can find a list of **all people that did not type in their address correctly**. First, contact those participants privately, verify their address and try to adapt it in a way (Spreadsheet linked below) that they are included in the algorithm. Here, you can **access the Spreadsheet** with the answers from the Running Dinner to **correct the address manually**.

Name	Address	E-Mail	Phonenumber	Name Teammem…
Marshall Matters	R. Fernandes Tomás 26 Li…	marcoschmiederer@gma…	491778092540	Mini Maus
Magardida Durarte	R. Arco do Cego 50 Lisboa	marcoschmiederer@gma…	491778092540	Karen Teilhabe
Ted Mosby	R. Academia das Ciências …	jonasnicolai@freenet.de	491778092545	Evan Ndica
Ben Dover	Parada Alto de São João …	jonasnicolai@freenet.de	491778092554	Sandra Schwarz

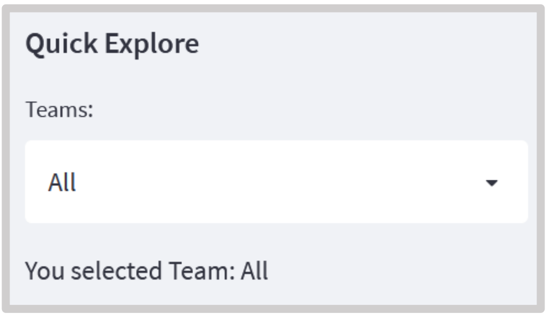
🕑 Wait List:

Here you can see the list of **all people who signed up too late** for the Running Dinner.

Name	Address	E-Mail	Phonenumber	Name Teammem…	E-I
Taylor Swift	R. Maestro Frederico de F…	jonasnicolai@freenet.de	491778092560	Dean Schwarz	jor
Jennifer Lopez	R. Saraiva de Carvalho 2A …	jonasnicolai@freenet.de	491778092561	Hanno Heiler	jor

III. Filter

For a better overview of the different group compositions, the organizers can use the Quick explore menu on the left-hand side. In the drop-down menu one can select each different team and get an overview with all information of the group members, as well as the map with the three different locations.



Quick Explore

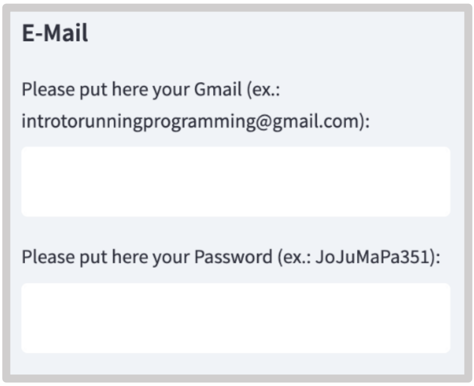
Teams: All

You selected Team: All

Step 6 – E-Mail Server

I. Google Login

In a final step, to inform all final teams, wait list teams, and teams with a wrong address, a Gmail email server is constructed with the API email.message as Email Message. However, in order to work, please insert your Gmail address as well as your user password in the corresponding text fields.

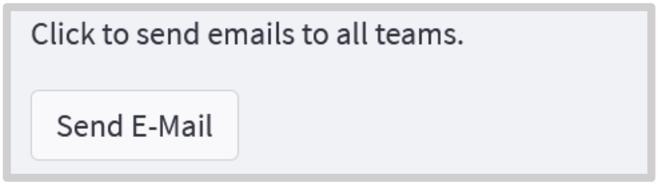

E-Mail

Please put here your Gmail (ex.: introtorunningprogramming@gmail.com):

Please put here your Password (ex.: JoJuMaPa351):

II. Send E-Mail Button

Lastly, after all Teams are allocated and all possible errors are checked or fixed the Running Dinner can be completed by sending out a confirmation email to all the participants. Therefore one needs to fill in the email address from which the mail will be sent as well as the login credentials for this account. For this, to work, one needs to use a gmail-account, as the API is accessing only google mail servers. If there is a problem with sending the email, please check the privacy settings of set account.


Click to send emails to all teams.

3. <u>Possible Error Problems</u>

- **GeoPy Timeout**
  - ⇒ Whenever too many calls are done to GeoPy, the data import doesn't respond. However, it is solved by waiting a few seconds and then running it again.
- **Error Message**: google.auth.exceptions.RefreshError: ('invalid_grant: Token has been expired or revoked.', {'error': 'invalid_grant', 'error_description': 'Token has been expired or revoked.'})
  - ⇒ This means that the token.pickle file with the necessary token is no longer valid. In this, the code needs to be run within a python terminal that creates a new pickle (e.g. Jupyter) which then needs to be uploaded again into GitHub. Unfortunately, adapting an automatic update of the pickle file within GitHub is beyond the scope of work for this group work. However, if needed, feel free to contact any of our team members, and we will update the token.
- **Email Server can't respond**
  - ⇒ Whenever you use an account that is not from Gmail or a Gmail account that has a two-factor authentication emails can't be sent out to the participants.