PICKTICK

3907, Block A, Sushant Lok Phase
I, Sector 28, Gurugram, Haryana
122022

Summer Internship Report

Facebook Events Web Scraping Project

UPES
THE NATION BUILDERS UNIVERSITY

Kavin Sharma
B.tech 4th Year
CSE - OIL & GAS INFORMATICS
UPES

Date: 29th June, 2017

TABLE OF CONTENTS

1 Acknowledgements

Firstly, I would like to express my sincere gratitude to my mentor Mr. Nishant Thapliyal for the continuous support in this internship project and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and give this unique Web Scraping Script in my final submission.

I could not have imagined having a better advisor and mentor for this internship period at Picktick India.

Besides my mentor I would like to thank the rest of Picktick India Team: Ms. Amrita Kashyap, and Mr. Nikhil Ahuja, for their help in testing, review and continuous Tech support. It was great help of Mr Rahul Jain for giving the overview of whole event creation process.
My sincere thanks also goes to Mr Sahil Bhutani who provided me an opportunity to join as intern.

2 Introduction and Overview

2.1 Picktick India – Background Information

An Augmented Reality based City Discovery Platform.
Picktick is a platform to discover 1000+ things to do near using augmented reality. Dive into a whole new world with Augmented Reality and explore your city from monuments to deals to restaurants to shopping to marathons all in one place

An Event Aggregator Platform

2.2 Introduction

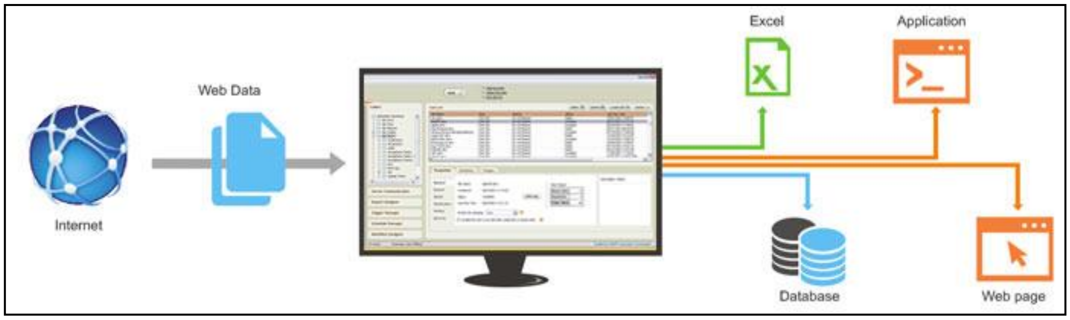
Performance of a Process affects the economic growth, labour cost, time complexity and productivity.

Process Automation has been a widely used technology to increase productivity and remove inefficiencies in various industries in the late 20th century. However, the Automated Web Scraping has been late in adopting technology solutions to increase its efficiency and productivity.

The primary reason for this anomaly is to Scrap the whole data from Facebook event page, this would not only require finding out data, but scrap there images, additionally merging images with a background then automatically store whole scraped data to firebase along with automatic blank cells and nan cells checking.

2.3 Need for Web Scraping

Web scraping is the process of extracting data from websites. Some data that is available on the web is presented in a format that makes it easier to collect and use it. Automating web scraping also allows for defining whether the process should be run at regular intervals in order to capture changes in the data



2.4 Issues in implementing Scraping

There are a variety of ways to **scrape** a website to extract information for reuse. In its simplest form, this can be achieved by copying and pasting snippets from a web page, but this can be impractical if there is a large amount of data to be extracted or if it is spread over a large number of pages. Instead, specialized tools and techniques can be used to automate this process by defining what sites to visit, what information to look for.

In current scenario we are using this simple approach for event creation in which we are just copying and pasting snipptes from a facebook event page which is having very low efficency and producitivity as well as increasing time complexity and labour cost which is totally affecting industry economy growth.

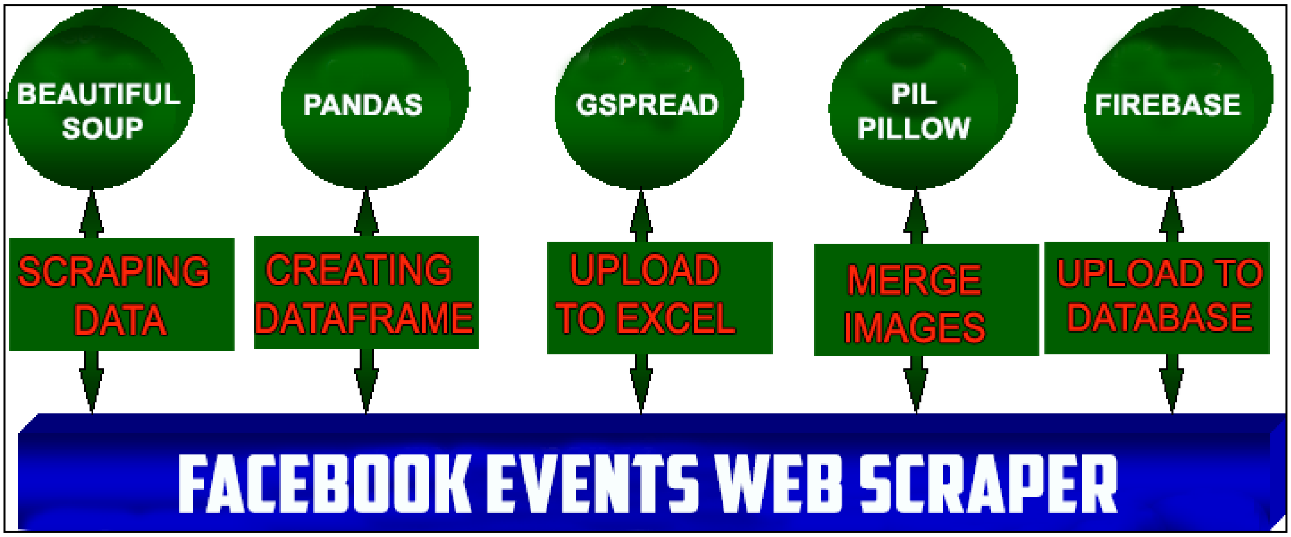
Using this simple approach for event creation also includes critical human errors which many time crashes the Picktick App so everytime we need to check whether App is crashing out or not, this increases time complexity of event creation process and increases downtime of Picktime Application.

2.5 Solution Approach

One possible option for addressing the above stated challenges is to create an automated web-scraping script in which the solutions to be developed and deployed can adopt to Python Django Webframework, make use of Python different-different libraries (BeautifulSoup, Pandas etc..).

2.6 Direction of Research at Facebook Events Web scraping

From Facebook events page we are scraping data using beautiful soup then we are creating a data frame using pandas library after data frame creation we uploading raw data to google spreadsheet and from that spreadsheet we the getting image urls and we are merging a background to all images and get a firebase Url and then we are uploading whole data to firebase



2.7 Project Overview

2.7.1 Part-I: Uploading Raw Data to Google Spreadsheet

This part of the project involved scraping the data from Facebook events page based on their urls which are putted in .txt file line by line, after extracting the required data a data frame created which is uploaded directly to google spreadsheet. Extracted information includes:

- ❑ Title, Start Date, End Date, Start Time, End Time, Start Seconds, End Seconds
- ❑ Location, Latitude Longitude, Image Url
- ❑ Ticket Information

One of the important achievements of this part was the automatically fetching of Latitude and Longitude. The major benefits of this includes:

- ❑ No App Crashing.
- ❑ It Decreases time complexity as there is no need to manually approve and disapprove the events.

2.7.2 Part-II: Merging Images Getting Firebase URL and Update to Spreadsheet

This part of project involved importing data from excel sheet to a dataframe from which we need to select image urls that are scraped from facebook after that all the images are mereged with a background along with maintaining aspect ratio.

Now merged images are uploaded to firebase and url of these uploaded images is updated on google spreadsheet

Image merging in a eqaul aspect ratio was on of the achievement. Major Benefits Include:

- ❑ No need to buy canvas license which reduces license cost as well as manual labour cost.
- ❑ Much Faster than manual image merging 1 min = approx. (30) images so decreases time complexity of image merging task.

2.7.3 Part-III: Atumated Testing : Check for Number Blank & nan cells

This part of project comes after manual testing of google spreadsheet which invloves checking of blank and null value containing cells in a column. It sums number of blank and nan vlaues in a column and if values increases then 0 then we need to manually check in google spreadsheet.

Major Benefits Include :

- ❑ Reduces Manual Work by Automated Testing of whole excel - sheet in one go.

2.7.4 Part-IV: Upload Whole Data to Firebase

It’s the last part of project which invloves uploading data of excel sheet directly to firebase by itrerating over each row along with adding Vendor ID and Event ID as a header for storing each event detail in a structured way.

Major Benefits Include :

- ❑ No Need to manually type scraped event information in a create event form which reduces time complexity and human errors

3 Project Part - I: Uploading Raw Data to Spreadsheet

3.1 Libraries Used:

- ❑ import requests

Intro: Requests will allow us to send HTTP/1.1 requests using Python. With it, we can add content like headers, form data, multipart files, and parameters via simple Python libraries. It also allows you to access the response data of Python in the same way. More Info

Installation: pip3 install requests

Usage: Here we are requesting event page URL and getting its data in the response in page variable.

Used for requesting only English version of webpage

```
headers = {"Accept-Language": "en-US,en;q=0.5"}
                    page = requests.get(url, headers=headers)
```

Requesting url along with headers

- ❑ import bs4

Intro: Beautiful Soup is a Python library for pulling data out of HTML and XML files. It works with your favourite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree More

Installation: pip install beautifulsoup4

Usage: In Beautiful Soup we will pass the response of requested URL as a parameter, here response is stored in page variable.

Extracting whole html structure of given url using 'lxml' parser

```
soup = bs4.BeautifulSoup(page.text, 'lxml')
                    data = soup.findAll("div", {'class': "hidden_elem"})
```

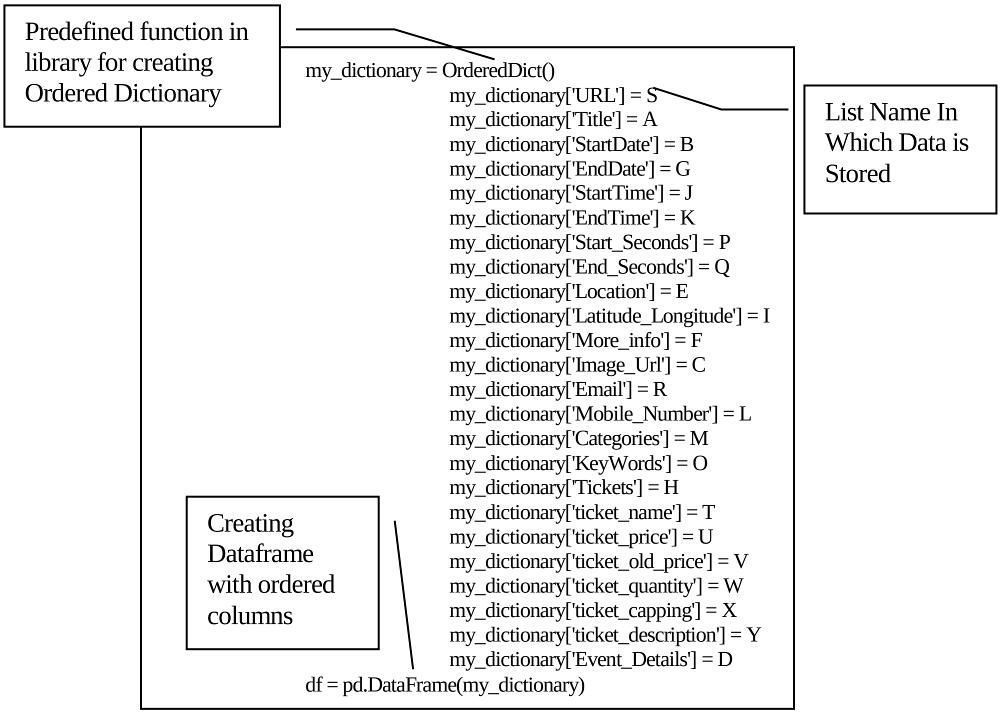
Finding text in div tag using its class

- ❑ from collections import OrderedDict

Intro: Ordered dictionaries are just like regular dictionaries but they remember the order that items were inserted. When iterating over an ordered dictionary, the items are returned in the order their keys were first added.More

Installation: pip3 install collections-extended

Usage: used for storing pandas dataframe columns to excel sheet in an ordered way.



Predefined function in library for creating Ordered Dictionary

List Name In Which Data is Stored

Creating Dataframe with ordered columns

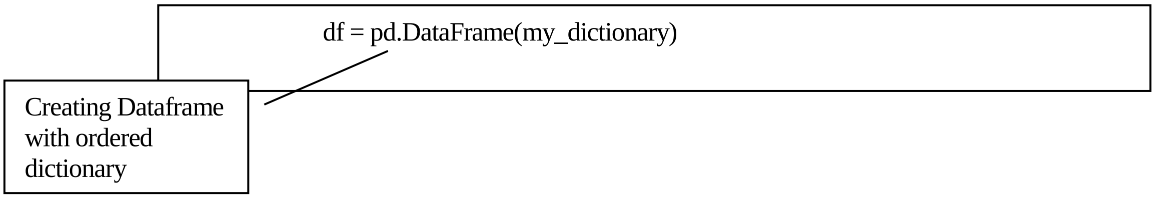
```
my_dictionary = OrderedDict()
            my_dictionary['URL'] = S
            my_dictionary['Title'] = A
            my_dictionary['StartDate'] = B
            my_dictionary['EndDate'] = G
            my_dictionary['StartTime'] = J
            my_dictionary['EndTime'] = K
            my_dictionary['Start_Seconds'] = P
            my_dictionary['End_Seconds'] = Q
            my_dictionary['Location'] = E
            my_dictionary['Latitude_Longitude'] = I
            my_dictionary['More_info'] = F
            my_dictionary['Image_Url'] = C
            my_dictionary['Email'] = R
            my_dictionary['Mobile_Number'] = L
            my_dictionary['Categories'] = M
            my_dictionary['KeyWords'] = O
            my_dictionary['Tickets'] = H
            my_dictionary['ticket_name'] = T
            my_dictionary['ticket_price'] = U
            my_dictionary['ticket_old_price'] = V
            my_dictionary['ticket_quantity'] = W
            my_dictionary['ticket_capping'] = X
            my_dictionary['ticket_description'] = Y
            my_dictionary['Event_Details'] = D
df = pd.DataFrame(my_dictionary)
```

❑ import pandas as pd

Intro: Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labelled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, **real world** data analysis in Python.More

Installation: pip3 install pandas

Usage: used for creating a dataframe which will store whole scraped data in different different columns.



```
df = pd.DataFrame(my_dictionary)
```

Creating Dataframe with ordered dictionary

❑ from df2gspread import df2gspread as d2g

Intro: Python library that provides possibility to transport table-data between Google Spreadsheets and Pandas DataFrame for further management or processing. Can be useful in all cases, when you need to handle the data located in Google Drive. See Google Spreadsheet Access Credential Guide here..
Installation: pip3 install df2gspread

Usage: used for uploading pandas dataframe to google spreadsheet

Spreadsheet Id & Sheet Name

```
spreadsheet = '1Wne9gj7CIgEtNJgcvuEgL1EMxqQRZ9UEfSJMp0hqKic'
                    wks_name = 'Sheet1'
d2g.upload(df, spreadsheet, wks_name)
```

Upload dataframe ‘ df ’ to spreadsheet

- ❑ import datetime

  Intro: The datetime module supplies classes for manipulating dates and times in both simple and complex ways.More

  Installation: pre installed in python 2.7 and above

  Usage: used for converting start datatime and end datetime to seconds

```
Stdatetime = “21 july 2017 11:00”
dt_obj = time.strptime(str(stdatetime), "%d %B %Y %H:%M")
      timestamp = time.mktime(dt_obj)
```

❑ I

Getting datetime in seconds

- ❑ from django.http import HttpResponse

  Intro: Django function used for showing response on each request.

  Installation: Predefined in Django views.py

  Usage: Printing Scraped Event Headings on WebPage.

List Which Includes Each Event Title

```
return HttpResponse(fetch)
```

- ❑ import numpy as np

  Intro: NumPy is the fundamental package for scientific computing with Python.

  Installation: pip3 install numpy

  Usage: used to appending null values

```
dat.append(np.nan)
```

3.2 User Defined Functions:

❑ **return event heading**

```
def heading(soup):
    heading = soup.find(class_="_5gmx")
    print(("Event data fectched: " + heading.string))
    head1 = str(heading.string)
    return str(head1)
```

Class Name in Which Event Heading present

Extracting Out Text In Between Tag

Return Event Heading

❑ **return event image url**

```
def ur_l(soup):
    tags = soup.findAll('img', class_="scaledImageFitWidth img")
    Url_1 = "\n".join(set(tag['src'] for tag in tags))
    tags = soup.findAll('img', class_="scaledImageFitHeight img")
    Url_2 = "\n".join(set(tag['src'] for tag in tags))
    if Url_1:
            return str(Url_1)
    else:
            return str(Url_2)
```

Class Name in Which Image Url present

Extracting Image Url from img src tag

If image URL available in Url_1 then return Url_1 else Url_2

❑ **return event details**

```
def Event_Details(data):
    for item in data:
            commentedHTML = item.find('code').contents[0]
            more_soup = bs4.BeautifulSoup(commentedHTML, 'lxml')
            Event_Details = more_soup.findAll('div', {'class': '_2qgs'})
            if Event_Details:
                    Event = Event_Details[0].text
    return Event
```

Finding all commented text along with tags in (code tag**)
** Code tag includes html commented code

Extracting whole html structure of commentedHTML using beautifulsoup

Class Name in Which Event details present

.text used for only extracting text

❑ return event location

```
def Location(data):
    for item in data:
            commentedHTML = item.find('code').contents[0]
            more_soup = bs4.BeautifulSoup(commentedHTML, 'lxml')
            Location = more_soup.findAll('a', {'class': '_5xhk'})
            if Location:
                    Locate = str(Location[0].text)
    return Locate
```

Finding all commented text along with tags in (code tag**)
** Code tag includes html commented code

Extracting whole html structure of commentedHTML using beautifulsoup

Class Name in Which Location present

.text used for only extracting text

- **return event More Info || Timming || Ticket Link**

```
def Tick_Time_Info(data):
      mainData = [ ]
      for item in data:
            commentedHTML = item.find('code').contents[0]
            more_soup = bs4.BeautifulSoup(commentedHTML, 'lxml')
            wanted_text = more_soup.findAll('div', {'class': '_5xhp fsm fwn fcg'})
            if wanted_text:

                        mainData.append(str(wanted_text[1].text))
                        mainData.append(wanted_text[0].text)
                        try:
                                    mainData.append(wanted_text[2].text)

                        except IndexError:
                                    mainData.append('nan')
      return mainData
```

Finding all commented text along with tags in (code tag**)
** Code tag includes html commented code

Extracting whole html structure of commentedHTML using beautifulsoup

3 classes with same name which includes More Info || Timming || Ticket Link

Appending extracted text to a list “mainData” which includes :

mainData[0] : Moreinfo
mainData[1] : (not using)**
mainData[2]: Ticket Link

** Now extracting timming from other classes {left out for future use}

If not available append null

- **return start date end date**

```
def Date(data):
      dat = [ ]
      for item in data:
            commentedHTML = item.find('code').contents[0]
            more_soup = bs4.BeautifulSoup(commentedHTML, 'lxml')
            Timming = more_soup.findAll("span", {'itemprop': "startDate"})
            if Timming:
                        try:
                                    dat.append(str(Timming[1].text))
                        except IndexError:
                                    dat.append(str(Timming[0].text))
                        try:
                                    dat.append(str(Timming[0].text))
                        except IndexError:
                                    dat.append(np.nan)
      return dat
```

Finding all commented text along with tags in (code tag**)
** Code tag includes html commented code

Extracting whole html structure of commentedHTML using beautifulsoup

2 Span tag classes with same name which includes start date end date

Appending extracted text to a list “dat” which includes :

dat [0] : end date**
dat [1] : start date

** if end date not available appending start date to list

- **return year**

Finding all commented text along with tags in (code tag**)
** Code tag includes html commented code

Extracting whole html structure of commentedHTML using beautifulsoup

```
def year(data):
      y = [ ]
      for item in data:
            commentedHTML = item.find('code').contents[0]
            more_soup = bs4.BeautifulSoup(commentedHTML, 'lxml')
            Timming = more_soup.findAll("div", {'class':
"_publicProdFeedInfo__timeRowTitle _5xhk"})
            if Timming:
                  y.append(Timming[0])
      return y
```

Class which includes whole timing info along with year

Appending whole text along with tags which will be later split out.

- **return start and end time**

Finding all commented text along with tags in (code tag**)
** Code tag includes html commented code

Extracting whole html structure of commentedHTML using beautifulsoup

```
def Timming(data):
      da = [ ]
      for item in data:
            commentedHTML = item.find('code').contents[0]
            more_soup = bs4.BeautifulSoup(commentedHTML, 'lxml')
            Timming = more_soup.findAll("span")
            if Timming:
                  try:
                        da.append((Timming[2].text))
                  except IndexError:
                        da.append("NAN")
                  try:
                        da.append((Timming[3].text))
                  except IndexError:
                        da.append("NAN")
      return da
```

Find all span tags and storing them to a list Timming

Extracting start time and entime from 2nd and 3rd SPAN tag which includes:

da[0]: start timing

da[1]: if available [endtime]
else : [NAN]

- **return time in seconds**

```
def startseconds(stdatetime):
    import time
    dt_obj = time.strptime(str(stdatetime), "%d %B %Y %H:%M")
    print(dt_obj)
    timestamp = time.mktime(dt_obj)
    print((repr(timestamp)))
    return timestamp

def endseconds(endatetime):
    import time
    try:
            dt_obj = time.strptime(str(endatetime), "%d %B %Y %H:%M")
            timestamp = time.mktime(dt_obj)
    except ValueError:
            timestamp = np.nan
```

Getting start datetime as a

Time Library

Format of datetime:
Eg: "21 july 2017 10:15"
"%d %B %Y %H:%M"

Datetime to seconds

Getting end datetime as a parameter

If enddatetime not available append null later appending startdatetime with timedelta + 3hrs

- **return Latitude and longitude**

```
def Lat_long(Locate, moreinfo):
    try:
            address = str(Locate + moreinfo)
            response = requests.get('https://maps.googleapis.com/maps/api/geocode/json?address= ' + address)
            resp_json_payload = response.json()
            Latlong = list((resp_json_payload['results'][0]['geometry']['location']).values())
    except IndexError:
            pass
    try:
            address = str(moreinfo)
            response = requests.get('https://maps.googleapis.com/maps/api/geocode/json?address= ' + address)
            resp_json_payload = response.json()
            Latlong = list((resp_json_payload['results'][0]['geometry']['location']).values())
    except IndexError:
            try:
                    address = str(Locate)
                    response = requests.get('https://maps.googleapis.com/maps/api/geocode/json?address= ' + address)
                    resp_json_payload = response.json()
                    Latlong = list((resp_json_payload['results'][0]['geometry']['location']).values())
            except IndexError:
                    Latlong = "nan"

    return Latlong
```

Getting location and more location information as parameter

Using combination of string (Locate + moreinfo) for finding latitude and longitude

Using google maps finding location lat, long

Converting address to json format

Storing latitude and longitude to Latlong

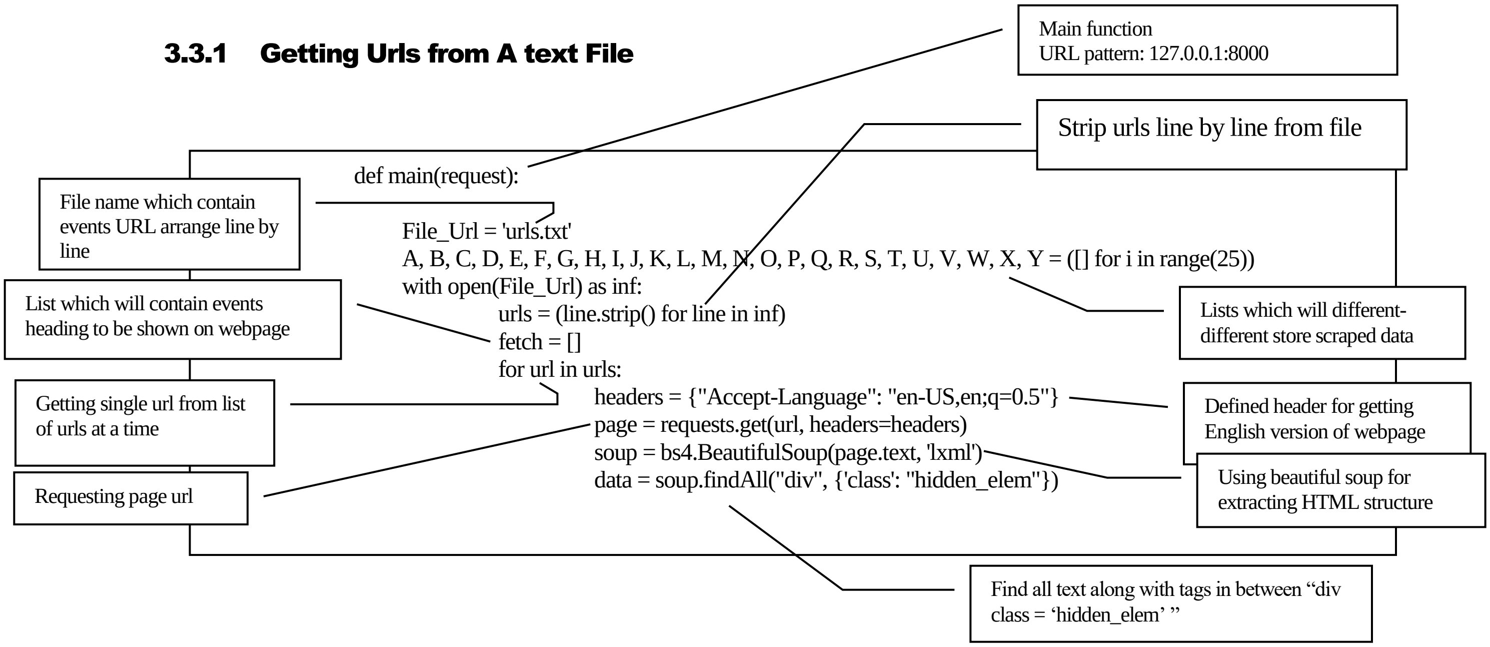
If latlong not found by combination then trying to find location using "moreinfo"

If not found using 'moreinfo' then trying using 'Locate'

If any combination does not work then appending null

3.3 Main Function :

3.3.1 Getting Urls from A text File



```
def main(request):
    File_Url = 'urls.txt'
    A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y = ([] for i in range(25))
    with open(File_Url) as inf:
        urls = (line.strip() for line in inf)
        fetch = []
        for url in urls:
            headers = {"Accept-Language": "en-US,en;q=0.5"}
            page = requests.get(url, headers=headers)
            soup = bs4.BeautifulSoup(page.text, 'lxml')
            data = soup.findAll("div", {'class': "hidden_elem"})
```

Main function
URL pattern: 127.0.0.1:8000

Strip urls line by line from file

File name which contain events URL arrange line by line

List which will contain events heading to be shown on webpage

Lists which will different-different store scraped data

Getting single url from list of urls at a time

Defined header for getting English version of webpage

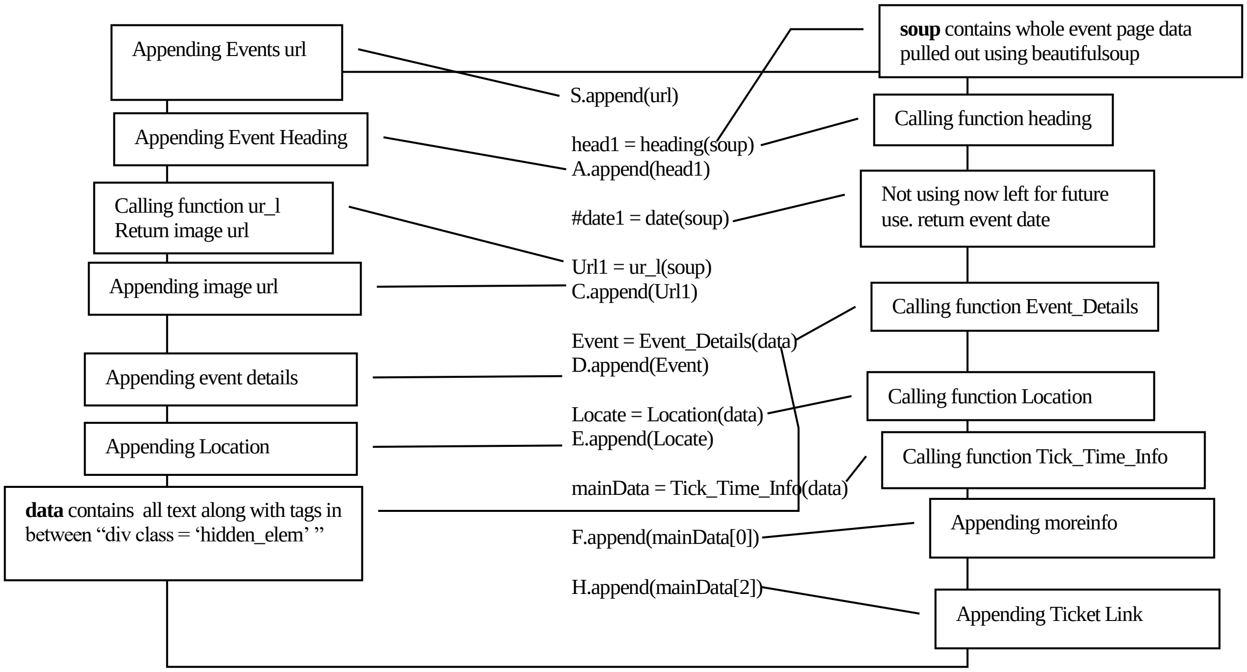
Requesting page url

Using beautiful soup for extracting HTML structure

Find all text along with tags in between “div class = ‘hidden_elem’ ”

3.3.2 Appending Data to Lists

3.3.2.1 Appending - URL, heading, image Url, Event Details, Location, Tick Time Info



```
S.append(url)

head1 = heading(soup)
A.append(head1)

#date1 = date(soup)

Url1 = ur_l(soup)
C.append(Url1)

Event = Event_Details(data)
D.append(Event)

Locate = Location(data)
E.append(Locate)

mainData = Tick_Time_Info(data)

F.append(mainData[0])

H.append(mainData[2])
```

Appending Events url

soup contains whole event page data pulled out using beautifulsoup

Appending Event Heading

Calling function heading

Calling function ur_l
Return image url

Not using now left for future use. return event date

Appending image url

Calling function Event_Details

Appending event details

Calling function Location

Appending Location

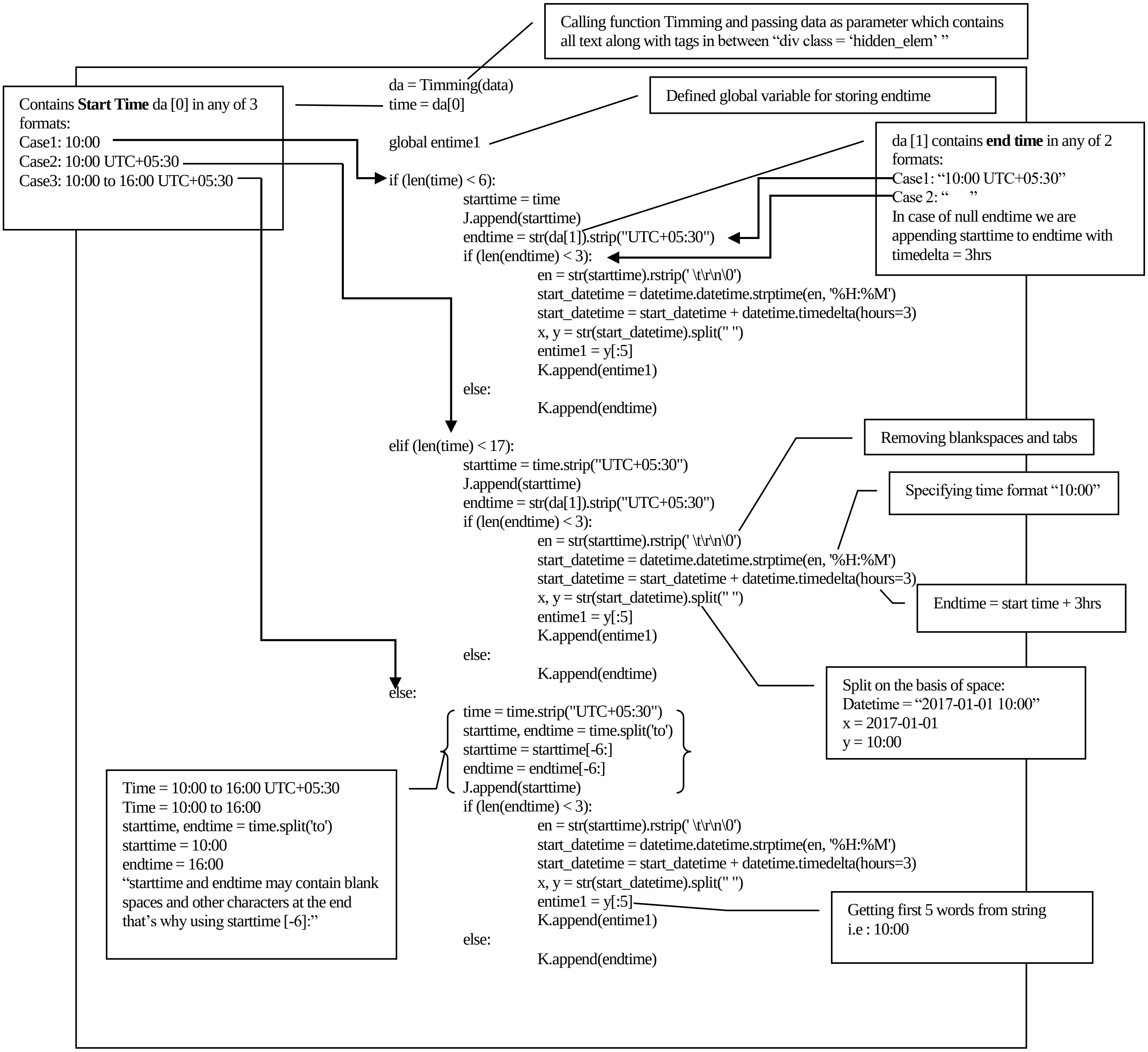
Calling function Tick_Time_Info

data contains all text along with tags in between “div class = ‘hidden_elem’ ”

Appending moreinfo

Appending Ticket Link

3.3.2.2 Appending - Start time, Endtime



Calling function Timming and passing data as parameter which contains all text along with tags in between “div class = ‘hidden_elem’ ”

Contains **Start Time** da [0] in any of 3 formats:
Case1: 10:00
Case2: 10:00 UTC+05:30
Case3: 10:00 to 16:00 UTC+05:30

Defined global variable for storing endtime

da [1] contains **end time** in any of 2 formats:
Case1: “10:00 UTC+05:30”
Case 2: “ ”
In case of null endtime we are appending starttime to endtime with timedelta = 3hrs

```
da = Timming(data)
time = da[0]

global entime1

if (len(time) < 6):
    starttime = time
    J.append(starttime)
    endtime = str(da[1]).strip("UTC+05:30")
    if (len(endtime) < 3):
        en = str(starttime).rstrip(' \t\r\n\0')
        start_datetime = datetime.datetime.strptime(en, '%H:%M')
        start_datetime = start_datetime + datetime.timedelta(hours=3)
        x, y = str(start_datetime).split(" ")
        entime1 = y[:5]
        K.append(entime1)
    else:
        K.append(endtime)

elif (len(time) < 17):
    starttime = time.strip("UTC+05:30")
    J.append(starttime)
    endtime = str(da[1]).strip("UTC+05:30")
    if (len(endtime) < 3):
        en = str(starttime).rstrip(' \t\r\n\0')
        start_datetime = datetime.datetime.strptime(en, '%H:%M')
        start_datetime = start_datetime + datetime.timedelta(hours=3)
        x, y = str(start_datetime).split(" ")
        entime1 = y[:5]
        K.append(entime1)
    else:
        K.append(endtime)
else:
    time = time.strip("UTC+05:30")
    starttime, endtime = time.split('to')
    starttime = starttime[-6:]
    endtime = endtime[-6:]
    J.append(starttime)
    if (len(endtime) < 3):
        en = str(starttime).rstrip(' \t\r\n\0')
        start_datetime = datetime.datetime.strptime(en, '%H:%M')
        start_datetime = start_datetime + datetime.timedelta(hours=3)
        x, y = str(start_datetime).split(" ")
        entime1 = y[:5]
        K.append(entime1)
    else:
        K.append(endtime)
```

Removing blankspaces and tabs

Specifying time format “10:00”

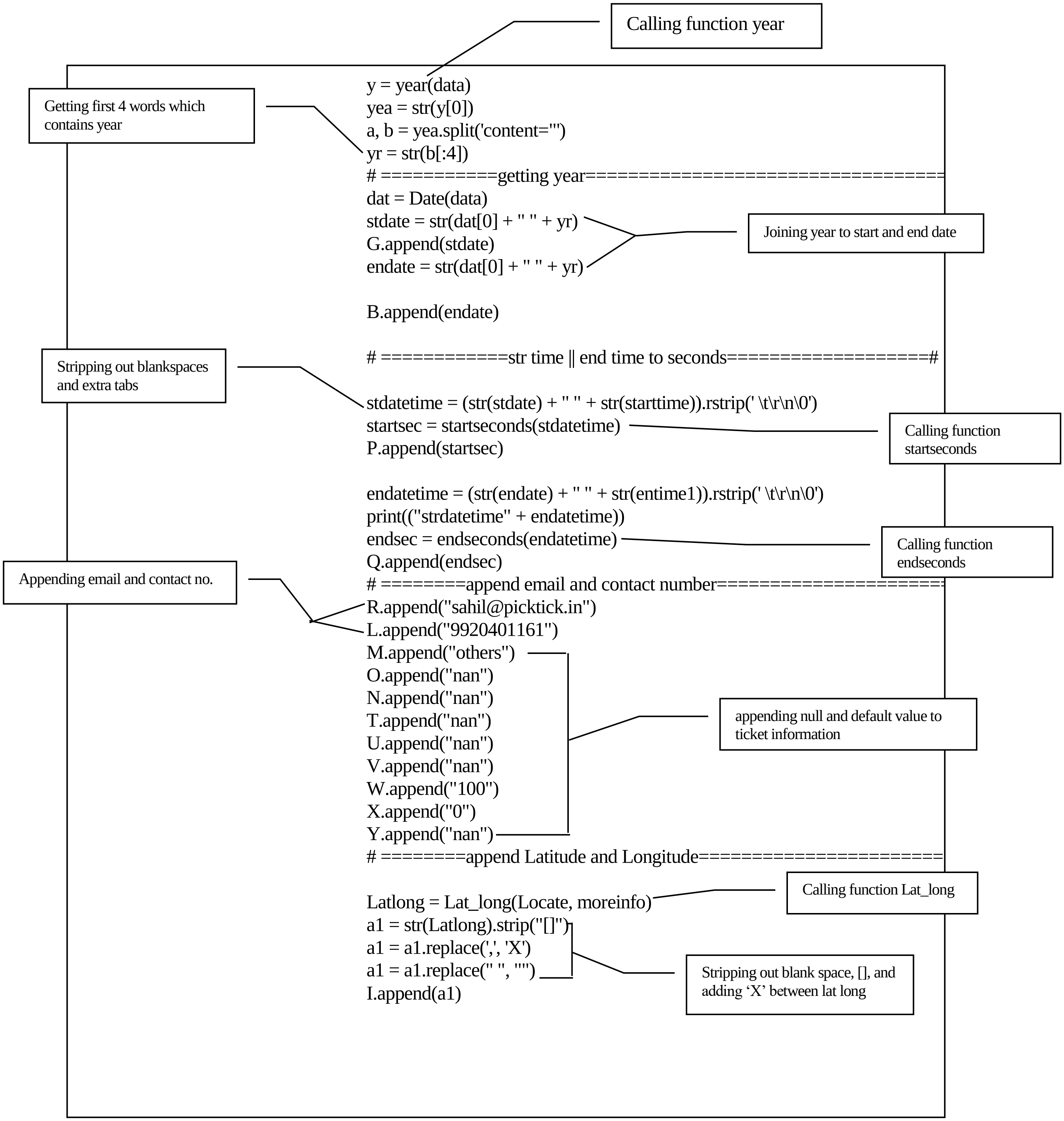
Endtime = start time + 3hrs

Split on the basis of space:
Datetime = “2017-01-01 10:00”
x = 2017-01-01
y = 10:00

Time = 10:00 to 16:00 UTC+05:30
Time = 10:00 to 16:00
starttime, endtime = time.split('to')
starttime = 10:00
endtime = 16:00
“starttime and endtime may contain blank spaces and other characters at the end that’s why using starttime [-6]:”

Getting first 5 words from string
i.e : 10:00

3.3.2.3 Appending year, start - end seconds, email, contact no., Latitude-Longitude

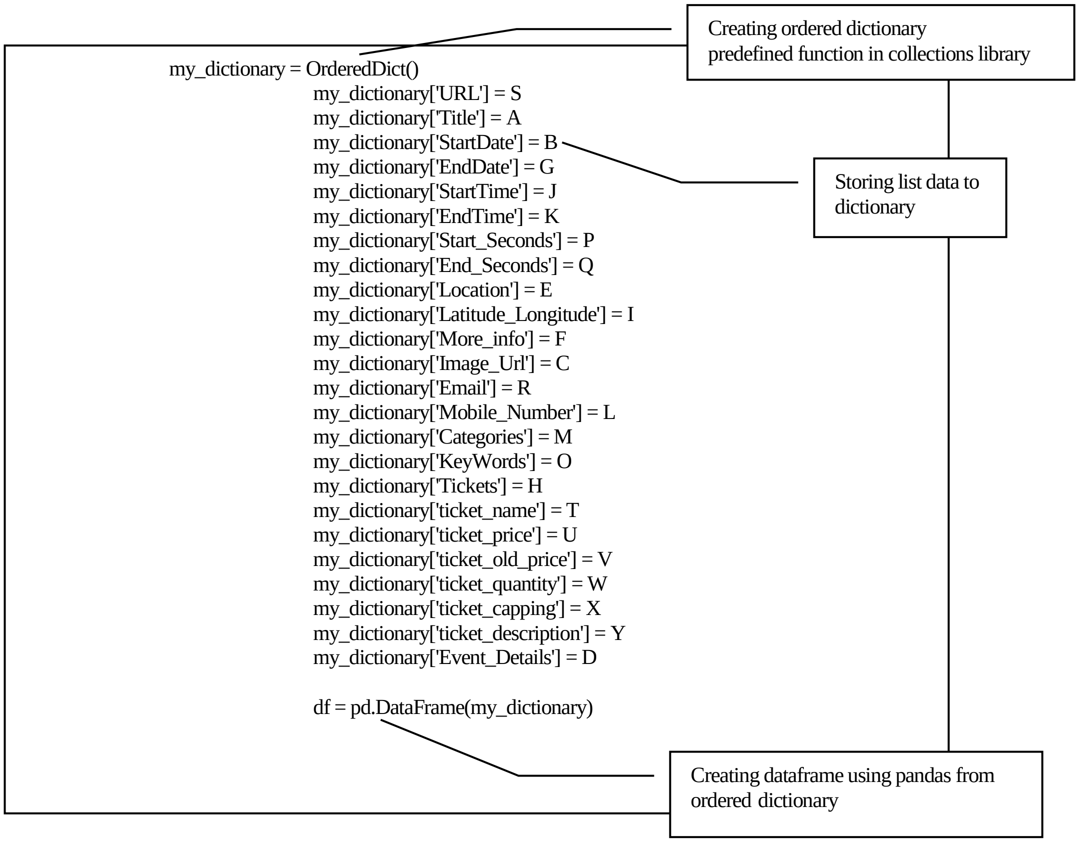


```
y = year(data)
yea = str(y[0])
a, b = yea.split('content="')
yr = str(b[:4])
# ===========getting year==================================
dat = Date(data)
stdate = str(dat[0] + " " + yr)
G.append(stdate)
endate = str(dat[0] + " " + yr)

B.append(endate)

# ============str time || end time to seconds===================#

stdatetime = (str(stdate) + " " + str(starttime)).rstrip(' \t\r\n\0')
startsec = startseconds(stdatetime)
P.append(startsec)

endatetime = (str(endate) + " " + str(entime1)).rstrip(' \t\r\n\0')
print(("strdatetime" + endatetime))
endsec = endseconds(endatetime)
Q.append(endsec)
# ========append email and contact number=====================
R.append("sahil@picktick.in")
L.append("9920401161")
M.append("others")
O.append("nan")
N.append("nan")
T.append("nan")
U.append("nan")
V.append("nan")
W.append("100")
X.append("0")
Y.append("nan")
# ========append Latitude and Longitude======================

Latlong = Lat_long(Locate, moreinfo)
a1 = str(Latlong).strip("[]")
a1 = a1.replace(',', 'X')
a1 = a1.replace(" ", "")
I.append(a1)
```

Calling function year
Getting first 4 words which contains year
Joining year to start and end date
Stripping out blankspaces and extra tabs
Calling function startseconds
Calling function endseconds
Appending email and contact no.
appending null and default value to ticket information
Calling function Lat_long
Stripping out blank space, [], and adding ‘X’ between lat long

3.3.3 Converting Data to Data Frames Using Pandas & Upload data to spreadsheet

3.3.3.1 Converting Lists Data To Data Frames Using Pandas



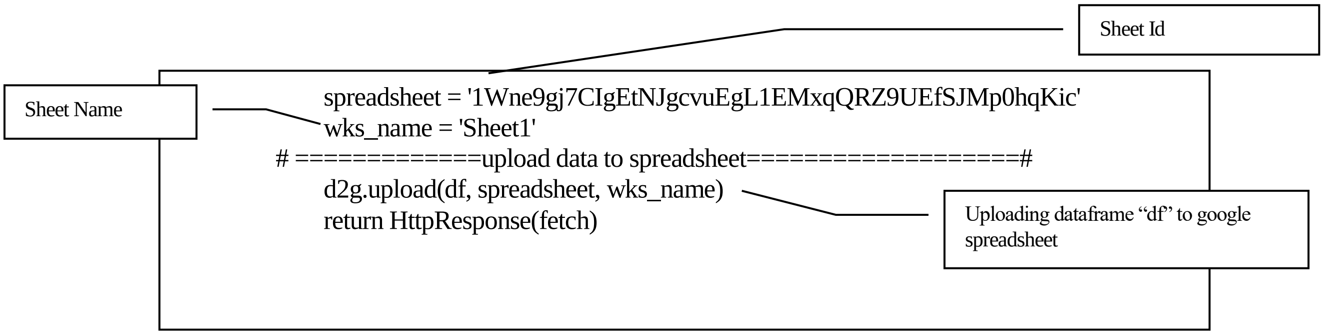
```
my_dictionary = OrderedDict()
          my_dictionary['URL'] = S
          my_dictionary['Title'] = A
          my_dictionary['StartDate'] = B
          my_dictionary['EndDate'] = G
          my_dictionary['StartTime'] = J
          my_dictionary['EndTime'] = K
          my_dictionary['Start_Seconds'] = P
          my_dictionary['End_Seconds'] = Q
          my_dictionary['Location'] = E
          my_dictionary['Latitude_Longitude'] = I
          my_dictionary['More_info'] = F
          my_dictionary['Image_Url'] = C
          my_dictionary['Email'] = R
          my_dictionary['Mobile_Number'] = L
          my_dictionary['Categories'] = M
          my_dictionary['KeyWords'] = O
          my_dictionary['Tickets'] = H
          my_dictionary['ticket_name'] = T
          my_dictionary['ticket_price'] = U
          my_dictionary['ticket_old_price'] = V
          my_dictionary['ticket_quantity'] = W
          my_dictionary['ticket_capping'] = X
          my_dictionary['ticket_description'] = Y
          my_dictionary['Event_Details'] = D

          df = pd.DataFrame(my_dictionary)
```

Creating ordered dictionary predefined function in collections library

Storing list data to dictionary

Creating dataframe using pandas from ordered dictionary

3.3.3.2 Uploading Dataframe to Google Spreadsheet



```
    spreadsheet = '1Wne9gj7CIgEtNJgcvuEgL1EMxqQRZ9UEfSJMp0hqKic'
    wks_name = 'Sheet1'
# ============upload data to spreadsheet==================#
    d2g.upload(df, spreadsheet, wks_name)
    return HttpResponse(fetch)
```

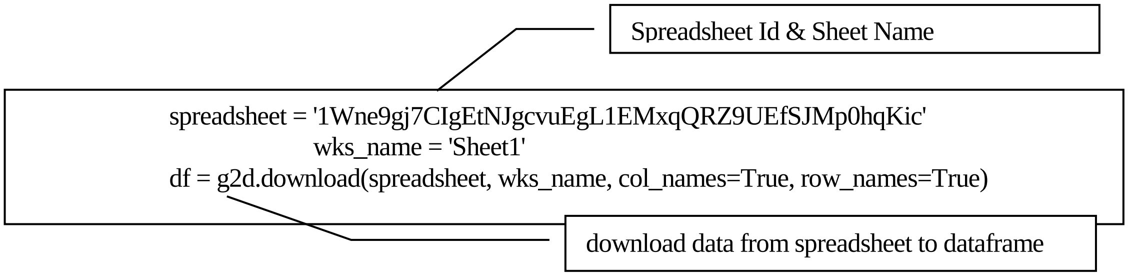
4 Project – II: Automated Testing: Check for Number Blank & Nan cells

4.1 Libraries Used

- from df2gspread import gspread2df as g2d

Intro: Python library that provides possibility to transport table-data between Google Spreadsheets and Pandas DataFrame for further management or processing. Can be useful in all cases, when you need to handle the data located in Google Drive. See Google Spreadsheet Access Credential Guide here..
Installation: pip3 install df2gspread

Usage: used for downloading data from spreadsheet to pandas dataframe

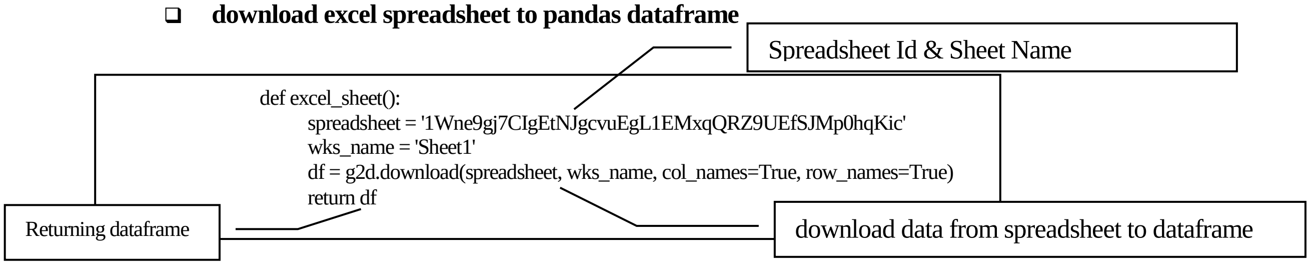


```
spreadsheet = '1Wne9gj7CIgEtNJgcvuEgL1EMxqQRZ9UEfSJMp0hqKic'
              wks_name = 'Sheet1'
df = g2d.download(spreadsheet, wks_name, col_names=True, row_names=True)
```

Spreadsheet Id & Sheet Name
download data from spreadsheet to dataframe

4.2 User Defined Function's:

- **download excel spreadsheet to pandas dataframe**



```
def excel_sheet():
    spreadsheet = '1Wne9gj7CIgEtNJgcvuEgL1EMxqQRZ9UEfSJMp0hqKic'
    wks_name = 'Sheet1'
    df = g2d.download(spreadsheet, wks_name, col_names=True, row_names=True)
    return df
```

Spreadsheet Id & Sheet Name
Returning dataframe
download data from spreadsheet to dataframe

4.3 Main Function:

```
def excel_spreadsheet(request):
    df = excel_sheet()
    dd = df.to_html()

    # ======== None and Empty Cells Checking In Dataframe==============================#
    event = str((~df.Event_Details.str.contains(r' ')).sum())
    Image_Url = str((df.Image_Url.str.contains(r' ')).sum())
    Location = str(((~df.Location.str.contains(r' ')).sum()) - 1)
    More_info = str((~df.More_info.str.contains(r' ')).sum())
    StartDate = str((~df.StartDate.str.contains(r' ')).sum())
    Tickets = str((df.Tickets.str.contains(r'nan')).sum())
    EndTime = str((df.EndTime.str.contains(r' ')).sum())
    Title = str((~df.Title.str.contains(r' ')).sum())
    EndDate = str((~df.EndDate.str.contains(r' ')).sum())
    StartTime = str((df.StartTime.str.contains(r' ')).sum())
    Start_Seconds = str((df.Start_Seconds.str.contains(r' ')).sum())
    End_Seconds = str((df.End_Seconds.str.contains(r'nan')).sum())
    Latitude_Longitude = str((df.Latitude_Longitude.str.contains(r'nan')).sum())
    Email = str((df.Email.str.contains(r' ')).sum())
    Mobile_Number = str((df.Mobile_Number.str.contains(r' ')).sum())
    Categories = str((df.Categories.str.contains(r' ')).sum())
    Ticket_Types = str((df.ticket_name.str.contains(r'nan')).sum())
    ticket_price = str((df.ticket_name.str.contains(r'nan')).sum())
    ticket_old_price = str((df.ticket_name.str.contains(r'nan')).sum())
    ticket_description = str((df.ticket_name.str.contains(r'nan')).sum())
    KeyWords = str((df.KeyWords.str.contains(r'nan')).sum())

    cg = ( "No. Of Blank Cells:" "</br> " + "Title: " + "<b>" + Title + "</b>" + "StartDate: " + StartDate +
    "EndDate: " + EndDate + "StartTime: " + StartTime + "EndTime: " + EndTime + "Start_Seconds: " + Start_Seconds +
    "End_Seconds: " + End_Seconds + "Location: " + Location + "Latitude_Longitude: " + Latitude_Longitude + "More_info: " +
     More_info + "Image_Url: " + Image_Url + "Tickets: " + Tickets + "Email: " + Email + "Mobile_Number: " + Mobile_Number +
     "Categories: " + Categories + "\nTicket_Types: \n" + Ticket_Types + "ticket_price" + ticket_price + "ticket_old_price"+
     ticket_old_price+"ticket_description"+ticket_description+"KeyWords: " + KeyWords + "Event_Details: " + event )

    de = cg + '<br/>' + dd
    return render(request, 'layout1.html', {'df': de})
```

url pattern :127.0.0.1:8000/excel

Calling function excel_sheet

Converting dataframe to html format

Using str.contains for check no. of blank and null cells in a column

String contains no of null and blank cells in each column

Joing dataframe html table with null cells resulted string

5 Merging Images Getting Firebase Url and Update To Spreadsheet

5.1 Libraries Used:

- ❑ from PIL import Image

  Intro: The Python Imaging Library (PIL) adds image processing capabilities to your Python interpreter. This library supports many file formats, and provides powerful image processing and graphics capabilities.More

  Installation: pip3 install pillow

  Usage: used for merging scraped facebook events images with a background

- ❑ import glob || import os

  Intro: The glob module finds all the pathnames matching a specified pattern according to the rules used by the Unix shell, although results are returned in arbitrary order More

  Installation: pip3 install glob

  Usage: used for removing all images from local directory after getting image url.

- ❑ from firebase import firebase

  Intro: Python interface to the Firebase's REST API more

  Installation: pip3 install python-firebase

  Usage: used for storing merged images in a database

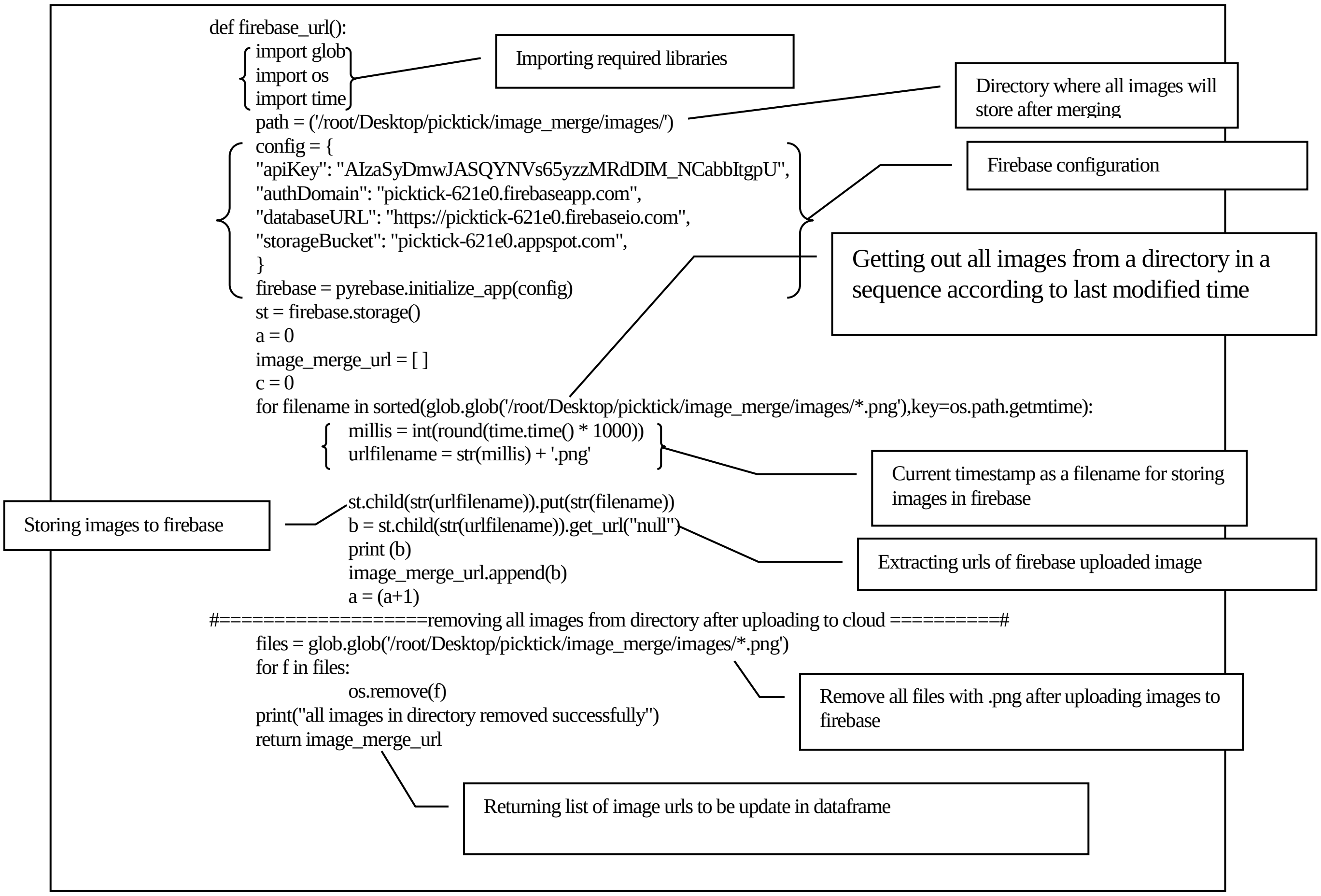
- ❑ import pyrebase

  Intro: A simple python wrapper for the Firebase API.more

  Installation: pip3 install pyrebase

  Usage: used for storing merged images in firebase.storage() and getting out urls

5.2 User Defined Function:



```
def firebase_url():
    import glob
    import os
    import time
    path = ('/root/Desktop/picktick/image_merge/images/')
    config = {
    "apiKey": "AIzaSyDmwJASQYNVs65yzzMRdDIM_NCabbItgpU",
    "authDomain": "picktick-621e0.firebaseapp.com",
    "databaseURL": "https://picktick-621e0.firebaseio.com",
    "storageBucket": "picktick-621e0.appspot.com",
    }
    firebase = pyrebase.initialize_app(config)
    st = firebase.storage()
    a = 0
    image_merge_url = [ ]
    c = 0
    for filename in sorted(glob.glob('/root/Desktop/picktick/image_merge/images/*.png'),key=os.path.getmtime):
            millis = int(round(time.time() * 1000))
            urlfilename = str(millis) + '.png'

            st.child(str(urlfilename)).put(str(filename))
            b = st.child(str(urlfilename)).get_url("null")
            print (b)
            image_merge_url.append(b)
            a = (a+1)
#==================removing all images from directory after uploading to cloud =========#
    files = glob.glob('/root/Desktop/picktick/image_merge/images/*.png')
    for f in files:
            os.remove(f)
    print("all images in directory removed successfully")
    return image_merge_url
```

Importing required libraries

Directory where all images will store after merging

Firebase configuration

Getting out all images from a directory in a sequence according to last modified time

Current timestamp as a filename for storing images in firebase

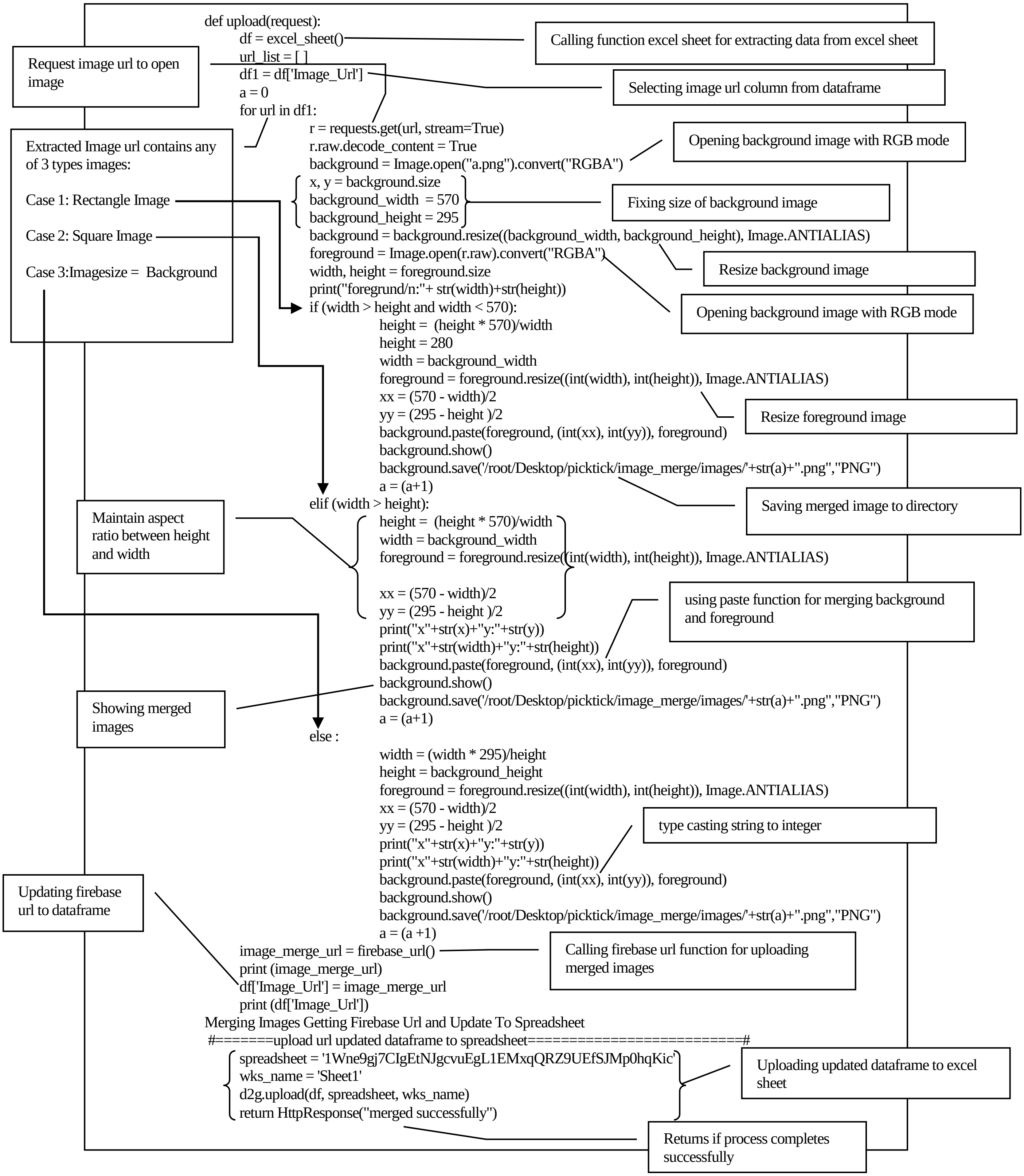
Storing images to firebase

Extracting urls of firebase uploaded image

Remove all files with .png after uploading images to firebase

Returning list of image urls to be update in dataframe

5.3 Main Function:


Calling function excel sheet for extracting data from excel sheet
Request image url to open image
Selecting image url column from dataframe
Extracted Image url contains any of 3 types images:
Case 1: Rectangle Image
Case 2: Square Image
Case 3:Imagesize = Background
Opening background image with RGB mode
Fixing size of background image
Resize background image
Opening background image with RGB mode
Resize foreground image
Saving merged image to directory
Maintain aspect ratio between height and width
using paste function for merging background and foreground
Showing merged images
type casting string to integer
Updating firebase url to dataframe
Calling firebase url function for uploading merged images
Uploading updated dataframe to excel sheet
Returns if process completes successfully

```
def upload(request):
    df = excel_sheet()
    url_list = [ ]
    df1 = df['Image_Url']
    a = 0
    for url in df1:
        r = requests.get(url, stream=True)
        r.raw.decode_content = True
        background = Image.open("a.png").convert("RGBA")
        x, y = background.size
        background_width  = 570
        background_height = 295
        background = background.resize((background_width, background_height), Image.ANTIALIAS)
        foreground = Image.open(r.raw).convert("RGBA")
        width, height = foreground.size
        print("foregrund/n:"+ str(width)+str(height))
        if (width > height and width < 570):
            height =  (height * 570)/width
            height = 280
            width = background_width
            foreground = foreground.resize((int(width), int(height)), Image.ANTIALIAS)
            xx = (570 - width)/2
            yy = (295 - height )/2
            background.paste(foreground, (int(xx), int(yy)), foreground)
            background.show()
            background.save('/root/Desktop/picktick/image_merge/images/'+str(a)+".png","PNG")
            a = (a+1)
        elif (width > height):
            height =  (height * 570)/width
            width = background_width
            foreground = foreground.resize((int(width), int(height)), Image.ANTIALIAS)

            xx = (570 - width)/2
            yy = (295 - height )/2
            print("x"+str(x)+"y:"+str(y))
            print("x"+str(width)+"y:"+str(height))
            background.paste(foreground, (int(xx), int(yy)), foreground)
            background.show()
            background.save('/root/Desktop/picktick/image_merge/images/'+str(a)+".png","PNG")
            a = (a+1)
        else :
            width = (width * 295)/height
            height = background_height
            foreground = foreground.resize((int(width), int(height)), Image.ANTIALIAS)
            xx = (570 - width)/2
            yy = (295 - height )/2
            print("x"+str(x)+"y:"+str(y))
            print("x"+str(width)+"y:"+str(height))
            background.paste(foreground, (int(xx), int(yy)), foreground)
            background.show()
            background.save('/root/Desktop/picktick/image_merge/images/'+str(a)+".png","PNG")
            a = (a +1)
    image_merge_url = firebase_url()
    print (image_merge_url)
    df['Image_Url'] = image_merge_url
    print (df['Image_Url'])
Merging Images Getting Firebase Url and Update To Spreadsheet
 #=======upload url updated dataframe to spreadsheet==========================#
    spreadsheet = '1Wne9gj7CIgEtNJgcvuEgL1EMxqQRZ9UEfSJMp0hqKic'
    wks_name = 'Sheet1'
    d2g.upload(df, spreadsheet, wks_name)
    return HttpResponse("merged successfully")
```

6 Upload Whole Data to Firebase

6.1 Libraries Used:

- ❑ import time

  Intro: This module provides various time-related functionsI more

  Installation: preinstalled in python 2.7 and above

  Usage: used for getting current time stamp

- ❑ from firebase import firebase

  Intro: Python interface to the Firebase's REST API more

  Installation: pip3 install python-firebase

  Usage: used for storing merged images in a database

- ❑ import pyrebase

  Intro: A simple python wrapper for the Firebase API.more

  Installation: pip3 install pyrebase

  Usage: used for storing merged images in firebase.storage() and getting out urls

6.2 User Defined Function:

- **download excel spreadsheet to pandas dataframe**

Spreadsheet Id & Sheet Name

```
def excel_sheet():
        spreadsheet = '1Wne9gj7CIgEtNJgcvuEgL1EMxqQRZ9UEfSJMp0hqKic'
        wks_name = 'Sheet1'
        df = g2d.download(spreadsheet, wks_name, col_names=True, row_names=True)
        return df
```

Returning dataframe

download data from spreadsheet to dataframe

6.3 Main Function

url pattern : 127.0.0.1:8000/database_push

```
def database_push(request):
    import time
    df = excel_sheet()
    print (len(df))
    config = {
    "apiKey": "AIzaSyDmwJASQYNVs65yzzMRdDIM_NCabbItgpU",
    "authDomain": "picktick-621e0.firebaseapp.com",
    "databaseURL": "https://picktick-621e0.firebaseio.com",
    "storageBucket": "picktick-621e0.appspot.com",}
    vendor_id = '10110269'
    firebase = pyrebase.initialize_app(config)
    database = firebase.database()
    event_ids = database.child('vendor').child(vendor_id).shallow().child('events').get().val()
    lis=[]
    for item in event_ids:
            f, l = item.split('_')
            lis.append(l)
    lists = sorted(lis, key=int)
    event_id =  (int(lists[-1]) + 1)
    b = 0
    c = len(df)
    while b< c:
            df = excel_sheet()
            df.rename(columns={'Title': 'name', 'Start_Seconds': 'start_time', 'End_Seconds':'end_time','Location':'venue_name',
             'Latitude_Longitude':'venue','Image_Url':'image','Email':'email','Mobile_Number':'mobile','KeyWords':'keyword',
             'Categories':'category', 'Event_Details':'description'}, inplace=True)
            df = df.iloc[[b]]
            tktInfo = df[['ticket_name', 'ticket_price', 'ticket_quantity','ticket_capping','ticket_description',
            'ticket_old_price' ]].copy()
            tktInfo = tktInfo.to_dict(orient='records')
            tktId = int(round(time.time() * 1000))
            newEventId = str(vendor_id)+"_"+str(event_id)
            data = df[['keyword', 'category', 'image', 'end_time', 'start_time', 'name', 'email', 'mobile', 'venue_name',
            'venue',  'description']].copy()
            data['event_id']= newEventId
            data['status']='pending'
            data['page_views']='0'
            data = data.to_dict(orient='records')
            dataV = df[['category', 'image', 'end_time', 'start_time', 'name']].copy()
            dataV['event_id']= newEventId
            dataV = dataV.to_dict(orient='records')
            print(newEventId)
            #print (dataV)
            print("it ends=======================================")
            #print (data)
            firebase = pyrebase.initialize_app(config)
            database = firebase.database()

            database.child('event').child(newEventId).child('details').set(data[0])
            database.child('vendor').child(vendor_id).child('events').child(newEventId).set(dataV[0])
            database.child('event').child(newEventId).child('details').child('ticket_category').child(tktId).set(tktInfo[0])

            event_id +=1
            b +=1
    return HttpResponse("datapushed successfully")
```

Calling function excel sheet for downloading data from sheet

Firebase configuration

Initialization and selecting firebase database

Getting out all event ids form firebase

1. Storing all event_ids to list
2. Splitting on the basis of "_"
3. Sorting them and selecting largest one \
4. Increasing by one for getting new event id

Renaming column name according to database structure

Selecting specific row from dataframe to store in firebase

Converting dataframe to records format

Created new dataframes according to firebase structure for directly storing data to firebase

Queries for storing data to firebase stores:
1. Data
2. Vendor data
3. Ticketinfo

Increment event id and store data from new row of excel sheet to firebase

Return when process completed successfully