Milestone 3

Group 5

Python code for the Artsonepass Web Scraper

Libraries

In [781]:
```
import requests
import bs4
from bs4 import BeautifulSoup
import json
import pandas as pd
import numpy as np
from pandas import DataFrame as df
from requests.exceptions import MissingSchema
import gspread
from oauth2client.service_account import ServiceAccountCredentials
from df2gspread import df2gspread as d2g
```

Classes

In [768]:
```
class crawly:
    def __init__(self, key, url):
        self.key = key
        self.url = url
        self.sub_pg = []

    def crawl(self): #Spider crawls pages to collect list of URLs
        if self.key == '{:04d}'.format(0): #Dada Dallas
            self.sub_pg=[]
            site = requests.get(self.url)
            site.encoding = 'ISO-885901'
            soup = BeautifulSoup(site.text, 'html.parser')
            page_list = soup.find_all(class_='event-name headliners')
            for i in page_list:
                cont = i.contents[0]
                link = cont['href']
                self.sub_pg.append(link)
        elif self.key =='{:04d}'.format(1):#texas ballet theater
            self.sub_pg=[]
            site = requests.get(self.url)
            site.encoding = 'ISO-885901'
            soup = BeautifulSoup(site.text, 'html.parser')
            page_list = soup.find_all("div", class_="bottom_btns")
            for i in page_list:
                cont = i.contents[1]
                self.sub_pg.append(i.contents[1]['href'])
        elif self.key =='{:04d}'.format(2):  #Theater3
            self.sub_pg =[]
            site = requests.get(self.url)
            site.encoding = 'ISO-885901'
            soup = BeautifulSoup(site.text, 'lxml')
            self.sub_pg = soup.find_all("a", class_="staff__item staff__item--
five-columns")
        elif self.key =='{:04d}'.format(3):#Latino Cultural Center
            self.sub_pg=[]
            site = requests.get(self.url)
            site.encoding = 'ISO-885901'
            soup = BeautifulSoup(site.text, 'html.parser')
            page_list = soup.find("iframe", class_="iframe-class")
            site = requests.get(page_list.get('src'))
            site.encoding = 'ISO-885901'
            soup = BeautifulSoup(site.text, 'lxml')
            sub_par = soup.find('div', class_='aswidget')
            for i in sub_par.find_all("a"):
                self.sub_pg.append("http:"+i.get("href"))
        elif self.key =='{:04d}'.format(5):#Trees
            self.sub_pg=[]
            site=requests.get(self.url)
            soup = BeautifulSoup(site.text,"html.parser")
            for i in soup.find_all("div",class_="thumb"):
                self.sub_pg.append(i.a.get("href"))
        elif self.key =='{:04d}'.format(6):#Dallas Arboretum
            self.sub_pg=[]
            site=requests.get(self.url)
            soup = BeautifulSoup(site.text,"html.parser")
            for i in soup.find_all("div",class_="eventCard__links"):
```

```
                self.sub_pg.append(i.a.get("href"))
            self.sub_pg = list(dict.fromkeys(self.sub_pg))
        elif self.key =='{:04d}'.format(7):#6floor museum,
            self.sub_pg=[]
            site=requests.get(self.url)
            soup = BeautifulSoup(site.text,"lxml")
            for i in soup.find_all("h3",class_="tribe-events-month-event-title
entry-title summary"):
                self.sub_pg.append(i.contents[4].get("href"))
            self.sub_pg = list(dict.fromkeys(self.sub_pg))
        else:
            print("Invalid Key")
            return[]
        return self.sub_pg
```

In [769]:
```
class creepy: #Scrapes crawled pages
    prd = {  #Dictionary Keys for formating
        'Org Key':"",
        'Event Key':"",
        'Event Title' :"",
        'Topline': "",
        'Headliner':"",
        'Openers':"",
        'Date(s)':"",
        'Time(s)':"",
        'Price/Admission':'',
        'Age Restriction':'',
        'Event Description':'',
        'Staff/Artists':'',
        'Category':'',
        'Venue':'',
        'Venue Info':'',
        'Street Name':'',
        'Address Line 2':',',
        'City':'',
        'State':'',
        'Postal Code':'',
        'Event Image URL':"",
        'Venue Info':'',
        'Location Link':'',
        'Get Tickets':'',
        'URL':''}

    def __init__(self, key, pages):
        self.key = key#ID
        self.pages = pages#pages to scrape
        self.info = []
        self.count=0

    def creep(self, bonus=str):
        if self.key == '{:04d}'.format(0): #Dada Dallas
            for link in self.pages:
                event_info = creepy.prd.copy()
                on_sale = True
                event_info.update([('Org Key',self.key)])
                event_info.update([('Event Key','{:06d}'.format(self.count))])
                self.count+=1
                s_page = requests.get(link)
                sub_soup = BeautifulSoup(s_page.text, 'html.parser')
                sub_parse = sub_soup.find('meta', property ="og:title")
                event_info.update([('Event Title', sub_parse.attrs['content'
])])
                sub_parse = sub_soup.find('section', class_="topline-info pres
ented-by")
                if type(sub_parse) is type(None):
                    event_info.update([('Topline', "NA")])
                else:
                    event_info.update([('Topline', sub_parse.get_text())])
                sub_parse = sub_soup.find('h1', class_="event-name headliners"
)
                event_info.update([('Headliner', sub_parse.get_text())])
```

```
                sub_parse = sub_soup.find('h2', class_="supports")
                if type(sub_parse) is type(None):
                    event_info.update([('Openers', 'NA')])
                else:
                    event_info.update([('Openers', sub_parse.get_text())])
                sub_parse = sub_soup.find('span', class_="dates")
                event_info.update([('Date(s)', sub_parse.get_text())])
                sub_parse = sub_soup.find('span', class_="start")
                event_info.update([('Time(s)', sub_parse.get_text())])
                sub_parse = sub_soup.find('span', class_="sales-ended inactiv
e")
                if type(sub_parse) is type(None):
                    sub_parse = sub_soup.find('span', class_="price-range")
                    event_info.update([('Price/Admission', sub_parse.get_text
())])
                else:
                    event_info.update([('Price/Admission', sub_parse.get_text
())])
                    on_sale = False
                sub_parse = sub_soup.find('section', class_="age-restriction a
ll-ages")
                if type(sub_parse) is type(None):
                    sub_parse = sub_soup.find('section', class_="age-restricti
on over-21")
                    event_info.update([('Age Restriction', sub_parse.get_text
())])
                else:
                    event_info.update([('Age Restriction', sub_parse.get_text
())])
                sub_parse = sub_soup.find('article', class_="event-descriptio
n")
                event_info.update([('Event Description', sub_parse.contents[1]
.text + " " +sub_parse.contents[3].text)])
                event_info.update([('Category', "Performance")])
                sub_parse = sub_soup.find('address', class_="venue-info")
                event_info.update([('Venue', sub_parse.contents[2])])
                event_info.update([('Venue Info', "Night Club")])
                sub_parse = sub_soup.find('meta', property="og:street-address"
)
                event_info.update([('Street Name', sub_parse.attrs['content'
])])
                sub_parse = sub_soup.find('meta', property="og:locality")
                event_info.update([('City', sub_parse.attrs['content'])])
                sub_parse = sub_soup.find('meta', property="og:region")
                event_info.update([('State', sub_parse.attrs['content'])])
                sub_parse = sub_soup.find('meta', property="og:postal-code")
                event_info.update([('Postal Code', sub_parse.attrs['content'
])])
                sub_parse = sub_soup.find('img')
                event_info.update([('Event Image URL', sub_parse.attrs['src'
])])
                event_info.update([('URL',link)])

                if on_sale == True:
                    sub_parse = sub_soup.find('section', class_ = 'ticket-pric
e')
                    for i in sub_parse.contents:
```

```
                        if type(i) == bs4.element.Tag and 'href' in i.attrs:
                            event_info.update([('Get Tickets', i.attrs['href'
])])
                        elif type(i) == bs4.element.Tag:
                            for j in i.contents:
                                if type(j)== bs4.element.Tag and 'href' in j.a
ttrs:
                                    event_info.update([('Get Tickets', j.attrs
['href'])])
                else:
                    event_info.update([('Get Tickets', event_info['Price/Admis
sion'])])
                event_info.update([('Location Link', 'https://goo.gl/maps/d3G9
381S4j1zD73s9')])

                self.info.append(event_info)
            return self.info



        elif self.key == '{:04d}'.format(1): #texas ballet
            for link in self.pages:
                event_info = self.prd.copy()
                event_info.update([('Org Key',self.key)])
                event_info.update([('Event Key','{:06d}'.format(self.count))])
                self.count+=1
                s_page = requests.get(link)
                sub_soup = BeautifulSoup(s_page.text, 'html.parser')
                sub_par = sub_soup.find('div', class_='title')
                vTitle = (sub_par.h1.text.replace('\n','') if sub_par.h1 else
contents[0]).strip()
                event_info.update([("Event Title",vTitle)])
                sub_par = sub_soup.find('div', class_='additional_info')
                vDates =(sub_par.p.text.replace('\n','') if sub_par.p else con
tents[0]).strip()
                event_info.update([("Date(s)",vDates)])
                sub_par = sub_soup.find("div", class_='main_title')
                desc=[]
                desc.append((sub_par.h3.text if sub_par.h3 else contetns[0]).s
trip())
                sub_par = sub_soup.find("div", class_='main_info')
                desc.append((sub_par.p.text if sub_par.p else contetns[0]).str
ip())
                sub_par = sub_soup.find_all("div", class_='single_info')
                for i in range(len(sub_par)):
                    desc.append((sub_par[i].h4.text.replace('\n','') if sub_pa
r[i].h4 else contents[0]).strip())
                    desc.append(sub_par[i].p.text)
                sub_par = sub_soup.find_all("div", class_='additional_info')
                vdes = (sub_par[1].text.replace('\n','') if sub_par[1].text el
se contents[0]).strip()
                desc.append(vdes)
                des=''
                event_info.update([("Topline", desc.pop(0))])
                for i in desc:
                    if i == desc[len(desc)-1]:
                        event_info.update([("Venue",i)])
```

```
                    else:
                        des+= i
                        des+="\n\n"
                event_info.update([("Event Description",des)])
                event_info.update([('Venue Info', "Performance Hall")])
                event_info.update([('Category', "Performance")])
                sub_par = sub_soup.find_all("div", class_='single_person')
                ar=[]
                for i in range(len(sub_par)):
                    ar.append((sub_par[i].text.replace('\n',"") if sub_par[0].
text else contents[0]).strip())
                art=""
                for i in ar:
                    art += i
                    art += "\n\n"
                event_info.update([("Staff/Artists",art)])
                sub_par = sub_soup.find(class_='image_holder_single')
                event_info.update([('Event Image URL', sub_par.contents[1].get
('src'))])
                event_info.update([('URL',link)])

                #
                self.info.append(event_info)

            spec = requests.get("https://texasballettheater.org/special-event
s/")
            spec.encoding = 'ISO-885901'
            sub_soup = BeautifulSoup(spec.text, 'lxml')
            page_list = sub_soup.find_all("div", class_="item same_height")
            bal_list = []
            for i in page_list:
                cont = i.contents[1]
                bal_list.append(cont['href'])

            for link in bal_list: #special events
                event_info = self.prd.copy()
                event_info.update([('Org Key',self.key)])
                event_info.update([('Event Key','{:06d}'.format(self.count))])
                self.count+=1
                s_page = requests.get(link)
                sub_soup = BeautifulSoup(s_page.text, 'html.parser')
                sub_par = sub_soup.find('div', class_='title')
                vTitle = (sub_par.h1.text.replace('\n','') if sub_par.h1 else
contents[0]).strip()
                event_info.update([("Event Title",vTitle)])
                sub_par = sub_soup.find('section', class_='post_image')
                event_info.update([("Event Image URL",sub_par.contents[1].get(
"src"))])
                event_info.update([('Category', "Visual,")])
                sub_par = sub_soup.find_all('div', class_='col-md-3 col-sm-6 c
ol-12')
                dinfo = (sub_par[0].text.replace('  ',"").replace('\n',' ') if
sub_par[0].text else contents[0]).strip()
                event_info.update([("Date(s)", dinfo)])
                sub_par.pop(0)
                tinfo = (sub_par[0].text.replace('  ',"").replace('\n',' ') if
sub_par[0].text else contents[0]).strip()
```

```
                event_info.update([("Time(s)", tinfo)])
                sub_par.pop(0)
                vinfo = (sub_par[0].text.replace('  ',"").replace('\n',' ') if
sub_par[0].text else contents[0]).strip()
                event_info.update([("Venue", vinfo)])
                event_info.update([('Venue Info', "Night Club")])
                sub_par.pop(0)
                pinfo = (sub_par[0].text.replace('  ',"").replace('\n',' ') if
sub_par[0].text else contents[0]).strip()
                event_info.update([("Price/Admission", pinfo)])
                sub_par.pop(0)
                sub_par = sub_soup.find('section', class_='content_text')
                event_info.update([('Event Description', sub_par.text.replace(
"\n"," ").replace("\xa0",""))])
                event_info.update([('URL',link)])
                sub_par = sub_soup.find('a', class_='dark_btn')
                event_info.update([('Get Tickets',sub_par.get("href"))])


                self.info.append(event_info)
            return self.info
        elif self.key == '{:04d}'.format(2):
            for i in self.pages:
                event_info = self.prd.copy()
                event_info.update([('Org Key',self.key)])
                event_info.update([('Event Key','{:06d}'.format(self.count))])
                self.count+=1
                event_info.update([('Event Image URL', i.contents[1].contents[
1].get("data-src"))])



                if len(i.contents[3])<3 :
                    event_info.update([('Event Title',"NA")]) #title
                    event_info.update([('Event Description',"Event data TBD"
)])
                elif type(i.contents[3].contents[1].h5) == bs4.element.Tag:
                    event_info.update([('Event Title',i.contents[3].contents[1
].h5.text)]) #title
                    l =i.contents[3].find_all('p')
                    event_info.update([('Topline',l[0].text)]) #topline
                    event_info.update([('Event Description',l[1].text)]) # des
c
                    event_info.update([('Time(s)',l[2].text)]) #time
                elif type(i.contents[3].p.text.split("\n",1)[0]) == str:
                    event_info.update([("Event Title",i.contents[3].p.text.spl
it("\n",1)[0])])
                    event_info.update([("Event Description",i.contents[3].p.te
xt.split("\n",1)[1])])
                    event_info.update([("Time(s)",i.contents[3].find_all('p')[
1].text)])

                event_info.update([('Category', "Performance")])

                event_info.update([('URL', bonus)])
                self.info.append(event_info)
```

```
            return self.info


        elif self.key == '{:04d}'.format(3):#Latino Cultural Center
            for link in self.pages:
                event_info = self.prd.copy()
                event_info.update([('Org Key',self.key)])
                event_info.update([('Event Key','{:06d}'.format(self.count))])
                self.count+=1
                s_page = requests.get(link)
                sub_soup = BeautifulSoup(s_page.text, 'html.parser')
                event_info.update([("Event Title",sub_soup.find('h1', class_=
'title').text)])
                event_info.update([("Event Image URL",("http:"+str(sub_soup.fi
nd("span",class_="header_thumbnail").contents[1]).split("\"",2)[1]))])
                event_info.update([("URL",link)])
                event_info.update([("Venue",sub_soup.find('span', class_='met
a').text)])
                event_info.update([("Event Description",sub_soup.find("div",cl
ass_="entry-content").p.contents[0].get("content").replace("\r","").replace("
\xa0","").replace("\xa0A","").replace("\n"," "))])


                self.info.append(event_info)
            return self.info


        elif self.key == '{:04d}'.format(5):# Trees
            for link in self.pages:
                event_info = self.prd.copy()
                event_info.update([('Org Key',self.key)])
                event_info.update([('Category',"Performance")])
                event_info.update([('Venue Info',"Night Club")])
                event_info.update([('Event Key','{:06d}'.format(self.count))])
                self.count+=1
                event_info.update([("URL",link)])
                s_page = requests.get(link)
                sub_soup = BeautifulSoup(s_page.text, 'html.parser')
                event_info.update([("Event Title",sub_soup.find("div",class_=
"page_header_left").h1.text)])
                event_info.update([("Openers", sub_soup.find("div", class_="pa
ge_header_left").h4.text.replace("\n"," - "))])
                event_info.update([("Get Tickets",sub_soup.find("div", class_=
"page_header_right").a.get("href"))])
                event_info.update([("Event Image URL",sub_soup.find("div", cla
ss_="content_item event_image gutter-bottom").img.get("src"))])
                sub_par = sub_soup.find("ul",class_="details")
                event_info.update([("Date(s)",sub_par.contents[1].span.text.re
place("\n","").replace("\t",""))])
                if len(sub_par.find_all("span")) >4:
                    time = "Start time: {}. Doors: {}".format(sub_par.find_all
("span")[2].text,sub_par.find_all("span")[4].text)
                elif len(sub_par.find_all("span"))>2:
                    time = "Start Time: {}".format(sub_par.find_all("span")[2]
.text)
                event_info.update([("Time(s)", time)])
                if type(sub_soup.find("span", class_="age_res")) != type(None
```

```
):
                    event_info.update([("Age Restriction", sub_soup.find("spa
n", class_="age_res").text)])
                if type(sub_soup.find("div", class_="collapse-wrapper")) != ty
pe(None):
                    event_info.update([("Event Description",sub_soup.find("di
v", class_="collapse-wrapper").text.replace("\n"," "))])
                vinfo=sub_soup.find("div", class_="venueinfo").text.replace("
\n","").replace("\t"," ").split("       ")
                for i in range(len(vinfo)):
                    vinfo[i] = vinfo[i].replace("  ","")
                event_info.update([("Venue",vinfo[0])])
                event_info.update([("Street Name",vinfo[1])])
                event_info.update([("City",vinfo[2].split()[0])])
                event_info.update([("State",vinfo[2].split()[1])])
                event_info.update([("Postal Code",vinfo[2].split()[2])])
                self.info.append(event_info)
            return self.info
        elif self.key == '{:04d}'.format(6): #Dallas Arboretum
            for link in self.pages:
                event_info = self.prd.copy()
                event_info.update([('Org Key',self.key)])
                event_info.update([('Category',"Science and Nature")])
                event_info.update([('Venue Info',"Botanical Garden")])
                event_info.update([('Event Key','{:06d}'.format(self.count))])
                self.count+=1
                event_info.update([("URL",link)])
                s_page = requests.get(link)
                sub_soup = BeautifulSoup(s_page.text, 'html.parser')
                sub_par=sub_soup.find("div", class_="centered__card")
                event_info.update([("Event Title",sub_par.h1.text.replace("\n"
,"").strip())])
                event_info.update([("Street Name","8525 Garland Road")])
                event_info.update([("City","Dallas")])
                event_info.update([("State","TX")])
                event_info.update([("Postal Code","75218")])
                sub_par=sub_soup.find_all("div", class_="wp-block-column")
                event_info.update([("Event Image URL", sub_par[0].find('img').
get("data-src"))])
                sub_par= sub_par[1].find_all('p')
                for i in range(len(sub_par)):
                    if sub_par[i].text == "Venue: ":
                        event_info.update([("Venue","{} @ The Dallas Arboretu
m".format(sub_par[i+1].text))])
                    elif sub_par[i].text == "Pricing: ":
                        event_info.update([("Price/Admission",sub_par[i+1].tex
t)])
                    elif sub_par[i].text == "Description: ":
                        event_info.update([("Event Description", sub_par[i+1].
text)])
                sub_par=sub_soup.find_all("div", class_="wp-block-columns has-
4-columns spacing__mtn")
                dates = []
                times=[]
                for i in range(len(sub_par)):
                    for j in sub_par[i].find_all("span",class_="styles__event-
date"):
```

```
                        dates.append(j.text.strip())
                for i in range(len(sub_par)):
                    for j in sub_par[i].find_all("span",class_="styles__event-
time"):
                        times.append(j.text.strip())
                event_info.update([("Date(s)", dates)])
                event_info.update([("Time(s)", times)])
                tl = []
                for i in sub_soup.find_all("a", class_="button__primary button
__emphasis button__icon"):
                    tl.append(i.get("href"))
                for i in sub_soup.find_all("a", class_="button__primary button
__emphasis button__icon button__small"):
                    tl.append(i.get("href"))
                event_info.update([("Get Tickets", tl)])
                self.info.append(event_info)
            return self.info


        elif self.key == '{:04d}'.format(7):#6 floor museum
            for link in self.pages:
                event_info = self.prd.copy()
                event_info.update([('Org Key',self.key)])
                event_info.update([('Category',"History")])
                event_info.update([('Venue Info',"Historic Museum")])
                event_info.update([('Event Key','{:06d}'.format(self.count))])
                self.count+=1
                event_info.update([("URL",link)])
                s_page = requests.get(link)
                sub_soup = BeautifulSoup(s_page.text, 'html.parser')
                if type(sub_soup.find("div", class_="centered__card")) != type
(None):
                    event_info.update([("Event Title",sub_soup.find("div", cla
ss_="centered__card").h1.text.replace("\n","").strip())])
                elif(type(sub_soup.find("div", class_="tribe-events-single"))
!= type(None)):
                    event_info.update([("Event Title",sub_soup.find("div", cla
ss_="tribe-events-single").h1.text.replace("\n","").strip())])
                dates=[]
                for i in sub_soup.find_all("abbr"):
                    if len(i.text.strip()) >4:
                        dates.append(i.text.strip())
                event_info.update([("Date(s)", dates)])
                event_info.update([("Time(s)", sub_soup.find("div", class_="tr
ibe-events-schedule tribe-clearfix").h2.text)])
                event_info.update([("Price/Admission", sub_soup.find("span", c
lass_="tribe-events-cost").text)])
                event_info.update([("Event Description", sub_soup.find("div",
class_="tribe-events-single-event-description tribe-events-content").text.repl
ace("\n"," "))])
                event_info.update([("Venue", sub_soup.find("dd",class_="tribe-
venue").text.strip())])
                for i in event_info["Venue"].split():
                    if str(i) == "Online" or str(i)=="Virtual" or str(i) == "o
nline" or str(i)=="virtual":
                        event_info.update([("Venue Info", "Virtual")])
                if event_info["Venue Info"] != "Virtual":
```

```
                    event_info.update([("Street Name", sub_soup.find("span", c
lass_="tribe-street-address").text)])
                    event_info.update([("City", sub_soup.find("span", class_=
"tribe-locality").text)])
                    event_info.update([("State", sub_soup.find("abbr", class_=
"tribe-region tribe-events-abbr").text)])
                    event_info.update([("Postal Code", sub_soup.find("span", c
lass_="tribe-postal-code").text)])
                    event_info.update([("Location Link", sub_soup.find("a",cla
ss_="tribe-events-gmap").get("href"))])
                self.info.append(event_info)
            return self.info
        else:
            print("Invalid Key")
            return[]
```

The Affiliate Organization class is the primary class for scraping data and manging organizations

In [770]:
```
class af_org:

    id_count=0
    orgs ={}                                                #List of Orgs
    org_data =pd.DataFrame()                               #Scraped Data
    flag_data = pd.DataFrame()                              #Flagged Data
    send_data = pd.DataFrame()                              #Akkio prediction in
put data
    pre_data = pd.DataFrame()                              #Akkio Prediction Val
ues
    par_AK = {"flow_key":"<FlowKey>", "api_key":"<ApiKey>", "data":""}


    @classmethod
    def to_local(cls):
#to Local Machine CSV
        df.to_csv(af_org.org_data, "M3.csv", sep = ",", index = False)
        df.to_csv(af_org.flag_data, "F3.csv", sep = ",", index = False)
        df.to_csv(af_org.pre_data, "P3.csv", sep = ",", index = False)
    @classmethod
    def akkio_run(cls):                                     #Akkil AI Predicts v
alues for events with missing categories
        af_org.send_data = af_org.org_data.copy()
        af_org.send_data= af_org.send_data.reset_index().drop(columns = ["inde
x","Org Key","Event Key","Street Name","Address Line 2","City","State","Postal
Code","Event Image URL","Location Link","Get Tickets","URL"])

        da=[]
        pred=[]
        tpos =-1
        for i in range(len(af_org.send_data.index)):
            temp={}

            if af_org.send_data.loc[i,"Age Restriction"] == "" or af_org.send_
data.loc[i,"Category"] == "" or af_org.send_data.loc[i,"Venue Info"] == "":

                for j in range(len(ajj.columns)):
                    if af_org.send_data.keys()[j] != "Age Restriction" and af_
org.send_data.keys()[j] != "Category" and af_org.send_data.keys()[j] != "Venue
Info":
                        if af_org.send_data.iat[i,j] != "" and type(af_org.sen
d_data.iat[i,j])!=list:
                            temp.update([(af_org.send_data.keys()[j],af_org.se
nd_data.iat[i,j])])
                        elif af_org.send_data.iat[i,j] != "" and type(af_org.s
end_data.iat[i,j])==list and len(af_org.send_data.iat[i,j]) != 0:
                            temp.update([(af_org.send_data.keys()[j],af_org.se
nd_data.iat[i,j][0])])
                da.append(temp)
                tpos+=1
                ev=[]
                ev.append(da[tpos])
                ev = json.dumps(ev, indent=4)
                af_org.par_AK.update([("data",ev)])
                ml = requests.get("https://api.akk.io/api", params= af_org.par
_AK).json()#Collect prediction values
```

```
                pred.append([af_org.send_data.index[i],ml])
        pre_par = []
        for i in pred:
            l={}
            l.update([("Index",i[0])])
            l.update([("Org Key",af_org.org_data.iloc[i[0],0])])
            l.update([("Event Key",af_org.org_data.iloc[i[0],1])])
            l.update([("Event Title",af_org.org_data.iloc[i[0],2])])
            for j in i[1]:
                for y in j.items():
                    if y[0]=="Age Restriction" or y[0]=="Category" or y[0]=="V
enue Info":
                        l.update([("PV: {}".format(y[0]),y[1])])
                    else:
                        cut = y[0].split()
                        short=""
                        for i in cut[cut.index("is"):]:
                            short = short+" "+i
                        l.update([(short,y[1])])
            pre_par.append(l)
        af_org.pre_data = df.from_records(pre_par)

    @classmethod
    def g_run(cls):                                    #Sends parsed data to Go
ogle Sheets Cloud
        scope = ['https://spreadsheets.google.com/feeds']
        credentials = ServiceAccountCredentials.from_json_keyfile_name('<Servi
ce Account Credentials File>', scope)
        gc = gspread.authorize(credentials)
        d2g.upload(af_org.org_data, "<Sheets ID>", "Affiliate Organizations",
credentials=credentials, row_names=True)
        d2g.upload(af_org.flag_data, "<Sheets ID>", "Flagged Affiliate Organiz
ations", credentials=credentials, row_names=True)
        d2g.upload(af_org.pre_data, "<Sheets ID>", "Akkio Predictions", creden
tials=credentials, row_names=True)


    @classmethod
    def g_get(cls):                                    #Pulls data from a Googl
e Sheets Cloud
        scope = ['https://spreadsheets.google.com/feeds']
        credentials = ServiceAccountCredentials.from_json_keyfile_name('<Servi
ce Account Credentials File>', scope)
        gc = gspread.authorize(credentials)
        spreadsheet_key = 'sheets key'
        book = gc.open_by_key(spreadsheet_key)
        worksheet = book.worksheet("Akkio Predictions")
        table = worksheet.get_all_values()
        stat = df.from_records(table).drop(0, axis = 1)
        stem = stat.iloc[0]
        stat = stat[1:]
        stat.columns = stem
        return(stat)


    def __init__(self, name):
# Initialize Event Org
```

```
        self.name = name
#Org Name
        self.sUrl =""
#URL to crawl
        self.ssurl = ""
# alternate url to crawl
        self.ID = '{:04d}'.format(af_org.id_count)
#org ID
        af_org.id_count+=1
        self.sub_pg = []
#pages to scrape
        self.data = []
#event data
        self.info= {"Org ID": self.ID,
#info
                    "Org Name": self.name,
                    "Org URL": self.sUrl}
        af_org.orgs[self.ID] = self.name

    def set_sUrl(self, sUrl):
# set URL
        try:
            requests.get(sUrl)
        except InvalidSchema as exception:
            print("URL is not complete: Please try again")
        except requests.ConnectionError as exception:
            print("URL does not exist on Internet. Please try Annother URL")
        except MissingSchema as exception:
            print("URL is not complete: Please try again")
        self.sUrl = sUrl
        self.info['Org URL'] = sUrl

    def update(self, data):                                  #new data is added
and sorted in the master dataframe
        self.data = data
        for i in range(len(af_org.org_data)):
            if af_org.org_data.at[i,"Org Key"]=='{:04d}'.format(int(self.ID)):
                af_org.org_data = af_org.org_data.drop(i)
        af_org.org_data = af_org.org_data.append(df.from_records(self.data), i
gnore_index = True)
        af_org.org_data = af_org.org_data.sort_values(by=['Org Key',"Event Ke
y"])

        flags = af_org.org_data.copy()
        flags = flags.replace(r'^\s*$', np.nan, regex=True)
        flags = flags.fillna("FLAG")
        for i in range(len(flags.columns)):              #Data is parsed and em
pty cells are flagged
            if flags.columns[i] == "Category":
                for j in range(len(flags[flags.columns[i]])):
                    if flags.iat[j,i] == "FLAG":
                        flags.iat[j,i] +=": CATEGORY"
            elif flags.columns[i] == "Age Restriction":
                for j in range(len(flags[flags.columns[i]])):   #Important inf
ormatrion have alternate flag tags
                    if flags.iat[j,i] == "FLAG":
                        flags.iat[j,i] +=": AGE"
```

```
            elif flags.columns[i] == "Event Image URL":
                for j in range(len(flags[flags.columns[i]])):
                    if flags.iat[j,i] == "FLAG":
                        flags.iat[j,i] +=": IMAGE"
            elif flags.columns[i] == "Venue":
                for j in range(len(flags[flags.columns[i]])):
                    if flags.iat[j,i] == "FLAG":
                        flags.iat[j,i] +=": VENUE"
            elif flags.columns[i] == "Event Title":
                for j in range(len(flags[flags.columns[i]])):
                    if flags.iat[j,i] == "FLAG":
                        flags.iat[j,i] +=": TITLE"
            elif flags.columns[i] == "Event Description":
                for j in range(len(flags[flags.columns[i]])):
                    if flags.iat[j,i] == "FLAG":
                        flags.iat[j,i] +=": DESCRIPTION"
        af_org.flag_data = flags
```

Running crawl and scrape functions

This creates the organization classes and runs the scraping and crawling methods

In [771]:
```
dada = af_org("Dada Dallas")
dada.set_sUrl("https://www.dadadallas.com/calendar/")
dada.sub_pg = crawly(dada.ID, dada.sUrl).crawl()
dada.update(creepy(dada.ID, dada.sub_pg).creep())
```

In [772]:
```
tbt = af_org("Texas Ballet Theater")
tbt.set_sUrl('https://texasballettheater.org/20-21season/')
tbt.ssUrl = "https://texasballettheater.org/special-events/" #Special secconda
y page
tbt.sub_pg = crawly(tbt.ID, tbt.sUrl).crawl()
tbt.update(creepy(tbt.ID, tbt.sub_pg).creep())
```

In [773]:
```
the = af_org("Theater3")
the.set_sUrl("https://www.theatre3dallas.com/shows-tickets/")
the.ID = '{:04d}'.format(2)
the.sub_pg = crawly(the.ID, the.sUrl).crawl()
the.update(creepy(the.ID, the.sub_pg).creep(the.sUrl))
```

In [774]:
```
LCC = af_org("Latino Cultural Center")
LCC.ID = '{:04d}'.format(3)
LCC.set_sUrl("http://lcc.dallasculture.org/programs/event-calendar/")
LCC.sub_pg = crawly(LCC.ID, LCC.sUrl).crawl()
LCC.update(creepy(LCC.ID, LCC.sub_pg).creep())
```

In [775]:
```
tre = af_org("Trees")
tre.ID = '{:04d}'.format(5)
tre.set_sUrl("https://www.treesdallas.com/events/all")
tre.sub_pg = crawly(tre.ID, tre.sUrl).crawl()
tre.update(creepy(tre.ID,tre.sub_pg).creep())
```

In [776]:
```
arb = af_org("The Dallas Arboretum")
arb.ID = '{:04d}'.format(6)
arb.set_sUrl("https://www.dallasarboretum.org/events-activities/calendar/")
arb.sub_pg = crawly(arb.ID, arb.sUrl).crawl()
arb.update(creepy(arb.ID, arb.sub_pg).creep())
```

In [777]:
```
jfk = af_org("6 floor museum")
jfk.ID = '{:04d}'.format(7)
jfk.set_sUrl("https://www.jfk.org/events/")
jfk.sub_pg = crawly(jfk.ID, jfk.sUrl).crawl()
jfk.update(creepy(jfk.ID, jfk.sub_pg).creep())
```

Connect To the Cloud

Run the class methods that sends data to the cloud

In [782]:
```
#af_org.akkio_run()                                    #Low on Akkio Resources
af_org.pre_data = af_org.g_get().copy()
#Pulls previous ^ data from gsheets
```

In [779]:
```
af_org.g_run()                                                #Sends scraped data to
 gsheets cloud
```

In [780]:
```
af_org.to_local()                                               #saves scraped data
 to Machine as CSV
```