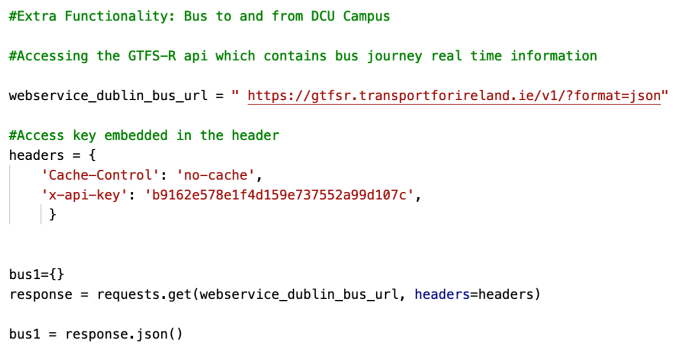
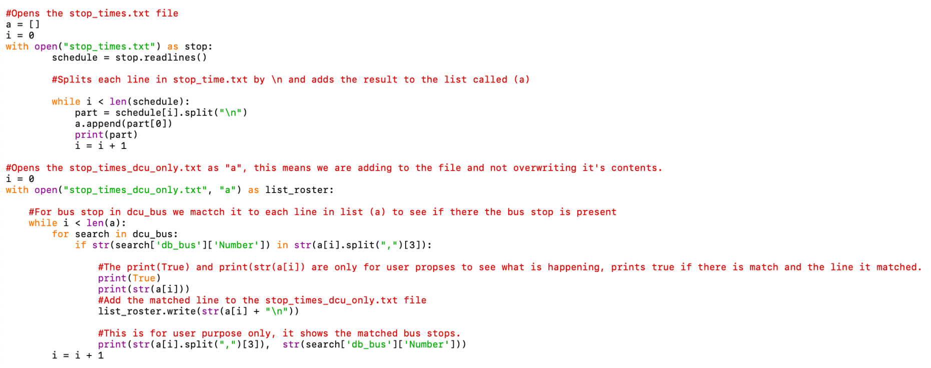
BusStopAtDcu:

For my extra functionality I chose to implement the GTFS-R API which shows real bus times. The extra functionality searches for buses arriving or departing from DCU bus stops. I had to sign up to https://developer.nationaltransport.ie/signin in order to make use of this api as it has a unique key. The following screenshot shows how
I extracted the information from the api.

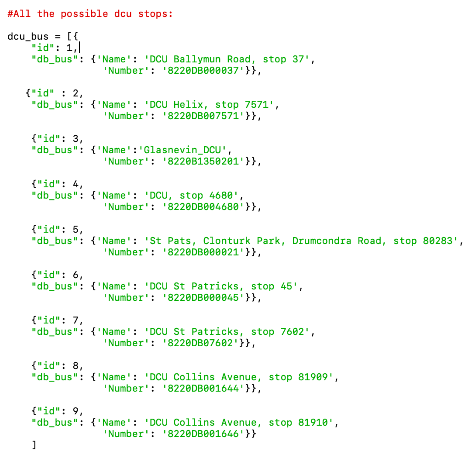


```
#Extra Functionality: Bus to and from DCU Campus

#Accessing the GTFS-R api which contains bus journey real time information

webservice_dublin_bus_url = " https://gtfsr.transportforireland.ie/v1/?format=json"

#Access key embedded in the header
headers = {
    'Cache-Control': 'no-cache',
    'x-api-key': 'b9162e578e1f4d159e737552a99d107c',
    }


bus1={}
response = requests.get(webservice_dublin_bus_url, headers=headers)

bus1 = response.json()
```

The data extracted from the api is used in conjunction with trips.txt, stop_times.txt and routes.txt. They are google transit datasets. I used the text files for translating the api data into readable data for each user. These txt files can be found at: https://www.transportforireland.ie/transitData/PT_Data.html. The stop_times.txt data set is too big to run searches through when retrieving information for the DCU campus bus stops, so I chose to run a search locally only for DCU stops and put it in a new file called stop_times_dcu_only.txt to achieve a faster load time. The script should be run each time the transit dataset is updated. This is the
python script used to separate the dataset.

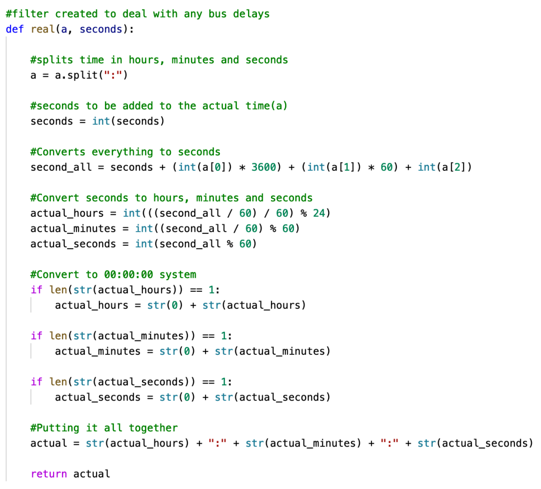


```
#Opens the stop_times.txt file
a = []
i = 0
with open("stop_times.txt") as stop:
        schedule = stop.readlines()

        #Splits each line in stop_time.txt by \n and adds the result to the list called (a)

        while i < len(schedule):
            part = schedule[i].split("\n")
            a.append(part[0])
            print(part)
            i = i + 1

#Opens the stop_times_dcu_only.txt as "a", this means we are adding to the file and not overwriting it's contents.
i = 0
with open("stop_times_dcu_only.txt", "a") as list_roster:

    #For bus stop in dcu_bus we mactch it to each line in list (a) to see if there the bus stop is present
    while i < len(a):
        for search in dcu_bus:
            if str(search['db_bus']['Number']) in str(a[i].split(",")[3]):

                #The print(True) and print(str(a[i]) are only for user propses to see what is happening, prints true if there is match and the line it matched.
                print(True)
                print(str(a[i]))
                #Add the matched line to the stop_times_dcu_only.txt file
                list_roster.write(str(a[i] + "\n"))

                #This is for user purpose only, it shows the matched bus stops.
                print(str(a[i].split(",")[3]),  str(search['db_bus']['Number']))
        i = i + 1
```



```
#All the possible dcu stops:

dcu_bus = [{
    "id": 1,
    "db_bus": {'Name': 'DCU Ballymun Road, stop 37',
               'Number': '8220DB000037'}},

   {"id" : 2,
    "db_bus": {'Name': 'DCU Helix, stop 7571',
               'Number': '8220DB007571'}},

    {"id": 3,
    "db_bus": {'Name':'Glasnevin_DCU',
               'Number': '8220B1350201'}},

    {"id": 4,
    "db_bus": {'Name': 'DCU, stop 4680',
               'Number': '8220DB004680'}},

    {"id": 5,
    "db_bus": {'Name': 'St Pats, Clonturk Park, Drumcondra Road, stop 80283',
               'Number': '8220DB000021'}},

    {"id": 6,
    "db_bus": {'Name': 'DCU St Patricks, stop 45',
               'Number': '8220DB000045'}},

    {"id": 7,
    "db_bus": {'Name': 'DCU St Patricks, stop 7602',
               'Number': '8220DB07602'}},

    {"id": 8,
    "db_bus": {'Name': 'DCU Collins Avenue, stop 81909',
               'Number': '8220DB001644'}},

    {"id": 9,
    "db_bus": {'Name': 'DCU Collins Avenue, stop 81910',
               'Number': '8220DB001646'}}
    ]
```

For any delayed arrival times at the DCU bus stops I used a filter to display the new arrival time. The filter extracts the expected arrival time for that bus stop according to it's scheduled id stored in stop_times_dcu_only.txt and adds(e.g. delay: {500}) or subtracts(e.g. delay: {-200}) the delayed(delay) time from the expected arrival time. This is the code for the filter:



```
#filter created to deal with any bus delays
def real(a, seconds):

    #splits time in hours, minutes and seconds
    a = a.split(":")

    #seconds to be added to the actual time(a)
    seconds = int(seconds)

    #Converts everything to seconds
    second_all = seconds + (int(a[0]) * 3600) + (int(a[1]) * 60) + int(a[2])

    #Convert seconds to hours, minutes and seconds
    actual_hours = int(((second_all / 60) / 60) % 24)
    actual_minutes = int((second_all / 60) % 60)
    actual_seconds = int(second_all % 60)

    #Convert to 00:00:00 system
    if len(str(actual_hours)) == 1:
        actual_hours = str(0) + str(actual_hours)

    if len(str(actual_minutes)) == 1:
        actual_minutes = str(0) + str(actual_minutes)

    if len(str(actual_seconds)) == 1:
        actual_seconds = str(0) + str(actual_seconds)

    #Putting it all together
    actual = str(actual_hours) + ":" + str(actual_minutes) + ":" + str(actual_seconds)

    return actual
```