Target Case Study

Aaron Carter - 6/25/2018

Problem 1 - API Consumption:

Packages Used:

- Requests: http://docs.python-requests.org/en/master/

Process:

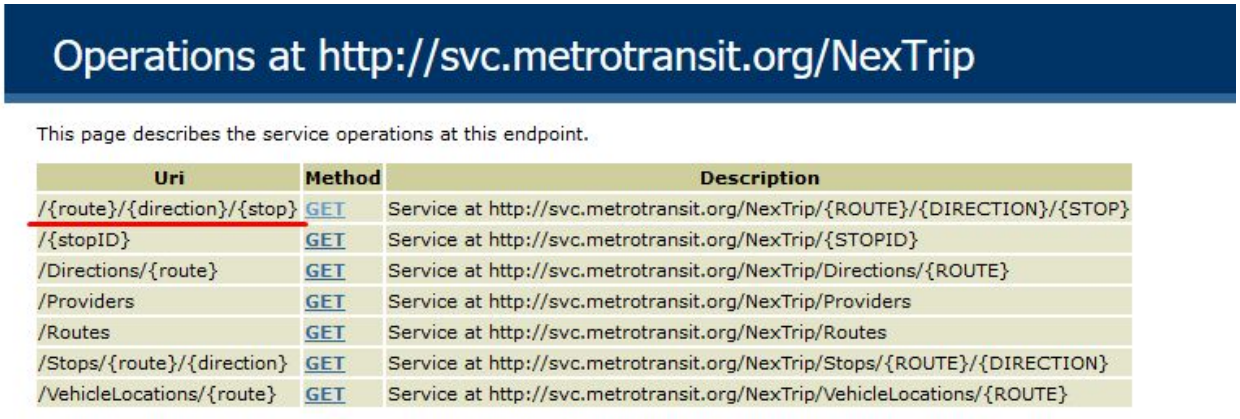
This is going to have some API stuff so we'll have to get that working in Python.
Looks like the preferred library is "requests" so lets get it:
http://docs.python-requests.org/en/master/user/install/#install

Problem: Permission denied: 'Pipfile' (while trying to install pipenv)
Solution: Run GitBash as administrator

Problem: TypeError: expected str, bytes or os.PathLike object, not NoneType
Solution: Just use pip to install instead of pipenv

Problem: How to use command line arguments in Python
Solution: This was a quick google search. Just import sys and use sys.argv to get the array of command line arguments. http://www.pythonforbeginners.com/system/python-sys-argv

Problem: Trying to use the URI found here to get a result, server responding with '400' (bad request)



Operations at http://svc.metrotransit.org/NexTrip

This page describes the service operations at this endpoint.

Uri	Method	Description
/{route}/{direction}/{stop}	GET	Service at http://svc.metrotransit.org/NexTrip/{ROUTE}/{DIRECTION}/{STOP}
/{stopID}	GET	Service at http://svc.metrotransit.org/NexTrip/{STOPID}
/Directions/{route}	GET	Service at http://svc.metrotransit.org/NexTrip/Directions/{ROUTE}
/Providers	GET	Service at http://svc.metrotransit.org/NexTrip/Providers
/Routes	GET	Service at http://svc.metrotransit.org/NexTrip/Routes
/Stops/{route}/{direction}	GET	Service at http://svc.metrotransit.org/NexTrip/Stops/{ROUTE}/{DIRECTION}
/VehicleLocations/{route}	GET	Service at http://svc.metrotransit.org/NexTrip/VehicleLocations/{ROUTE}

Solution: I was using strings like the names of the routes, which is wrong. In looking at the links on the API info page, they lead to something like this:

- http://svc.metrotransit.org/NexTrip/Stops/5/4

So the {route} and {direction} are going to be referenced by their IDs.

This means I'll need to convert the arguments from the command line to
Unless I take a snapshot of the whole thing once and can use that for multiple steps.
That might be overkill though. It seems more intuitive to get everything in a specific request like you would with an SQL query but it doesn't seem like it will work in that way.

So the information I have is the following, which I get from the user:
BUS_ROUTE
DIRECTION
BUS_STOP_NAME

I need to figure out which table or combination of tables most efficiently gets me the information I need to calculate the next bus departure time.

The API would like requests to be made no more frequently than every 30 seconds.
This is probably more important for things that are constantly making requests but it would be nice to keep requests infrequent.
We can save results if we want and just adjust times ourselves in the program as time passes.
If the user tries to check more than once in 30 seconds we can use "cached" results.
I might be overthinking this, probably something that would be good to ask about.

Gotta find the route #

Also the direction which looks like
South = 1
East = 2
West = 3
North = 4
So we'll convert the string into one of those numbers

Lastly the bus stop ID:
GetStops is what we need to use.

This is starting to look like it would be simpler in JS, it would allow for a simple and effective UI

We won't need to update the bus stop IDs or the route #s very often so we can probably just save the results if we want so we don't have to ask for them from the server each time. Ask about that.

GetTimepointDepartures looks to be the one we need for the end result
So I just need to grab all the information we need and then I can just do that request.

Problem: I'm getting empty strings back from my getRouteNumber() function.
Solution: I just needed to check for containment instead of equivalency.

```
if(BUS_ROUTE.upper() == item["Description"].upper()):
if(BUS_ROUTE.upper() in item["Description"].upper()):
```

Problem: I'm getting '400' back from my getStopId() function
Solution: I was using the wrong arguments when I called the function

```
direction = getDirection(DIRECTION)

route = getRouteNumber(BUS_ROUTE)

stop_id = getStopId(route, direction, BUS_STOP_NAME)

next_bus = getNextBus(route, direction, stop_id)


return next_bus
```

These are the correct arguments. Previously I was using BUS_ROUTE and DIRECTION. I also changed them for getNextBus().

Now I have to get it to run in the terminal using arguments.

Problem: Argument list is too long for some reason.
Solution: Looks like the first item in sys.argv is the file path for the program. I just changed it to look for the others and check for 4 arguments instead of 3.

So the program is working. Some things I want to implement now:

- Better error handling
- Feedback to the user
- More comments
- Save static data (Route IDs) to a file so we don't have to request it from the server each time. This would cut down on API calls.

I've entertained the idea of making a more flexible function for making the request, something that only requires a single function for making any type of request (since all the functions that currently make API calls are so similar).
Function would maybe look something like this:

```
def requestData({url}, {keyGiven}, {valGiven}, {keyNeeded}):
```

This would require another function to build the url that you wanted.
Probably asks too much of the user. Requires knowledge of the API which we want to avoid.
Alternatively we could check each of the fields in an item and string match. Seems kind of scrappy though.
Benefit of this would be a more expandable program. It could more easily get different values later on while reusing functions. (To be honest this is probably too complicated and not worth it)
The more I think about this the uglier it gets.

I've added more comments and error feedback.

```
def getRouteNumber(BUS_ROUTE):
    '''Gets Route Number from the input route name'''

    route = ""

    # Request the list of bus routes
    response = requests.get("http://svc.metrotransit.org/NexTrip/Routes?format=json")

    if(response.status_code == 200):
        # Store results into a list
        route_array = json.loads(response.content.decode('utf-8'))

        for item in route_array:
            # Find the route number of the specified route
            if(BUS_ROUTE.upper() in item["Description"].upper()):
                route = item["Route"]
    else:
        print("Error: [BUS ROUTE] The requested bus route does not exist.")

    return route
```

Docstrings could be better/ more descriptive/ more conventional.

- https://www.python.org/dev/peps/pep-0257/

Now time to just start testing.

Problem: Errors are not showing properly.
Solution: The error should be checking after checking all the fields for string containment intead of after the request on getRouteNumber() and getStopId(). The request will always succeed on them as long as that pages exist and work. I've adjusted the handling and feedback.

```
def getRouteNumber(BUS_ROUTE):
    '''Gets Route Number from the input route name'''

    route = ""

    # Request the list of bus routes
    response = requests.get("http://svc.metrotransit.org/NexTrip/Routes?format=json")

    if(response.status_code == 200):
        # Store results into a list
        route_array = json.loads(response.content.decode('utf-8'))

        for item in route_array:
            # Find the route number of the specified route
            if(BUS_ROUTE.upper() in item["Description"].upper()):
                route = item["Route"]
        if(not route):
            print("Error: [BUS ROUTE] The requested bus route does not exist.")
    else:
        print("Error: [BUS ROUTE] Code - " + response.status_code)

    return route
```

Now it notifies the user correctly.

Now I want to put the routes data in a file so I don't have to make a request for it every time.

(We could put the function to update in its own py file and set it to run once a day or something. Another thing we could do is set it to save the time when it was last updated. Then if it has been more than 24 hours, it updates the file before running the rest of the program.)

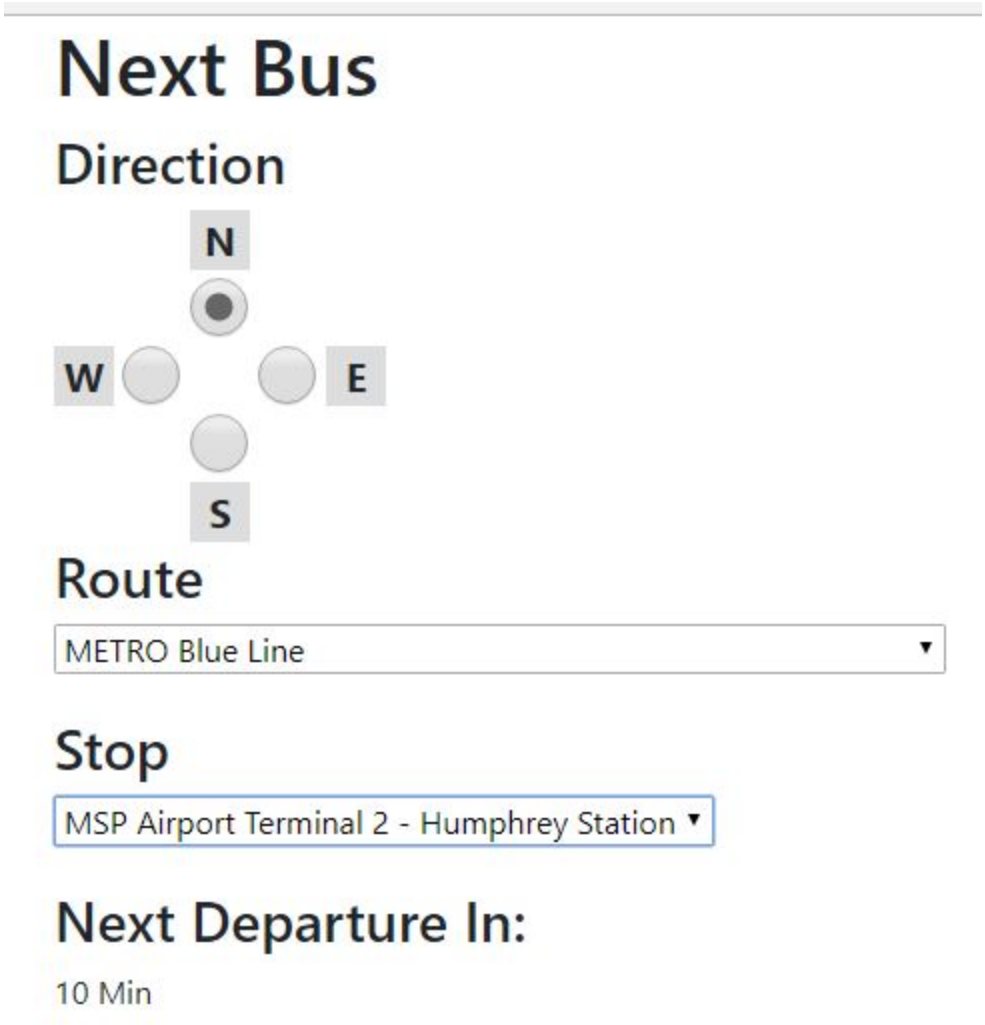
I'll need 2 functions, one for retrieving the content from the file, the other for updating the file.

readRoutes() is easy: open file for reading, get contents, close file, return contents.
updateRoutes() is basically just the same as the getRouteNumber() function except it opens a file and writes the entire response content to the file.

Now I have the option to run the program only making 2 requests instead of 3.

I've also whipped up a rough version using JS.
It is using jquery to check for input field changes and update as necessary. It is pretty much the same as the python version as far as functionality goes, despite the different syntax for making requests.

I like that in this version you present the options to the user instead of making them type out the whole string on the command line. Makes the experience a lot better.



Things I would add/change on this version:

- Make the field updating more robust. It's really scrappy right now.
- Styling, it would be nice if it looked better because OOOOOHH BOY is it ugly.
- Responsive. Not that it really needs it but making it mobile friendly would be neat.
- This would be a good time to use Angular.

Now I just need to write a README.md for running the program.