UNIVERSITY OF SCIENCE,
HO CHI MINH

Query big data in JSON file with Python using OOP to optimize the handling process

TECHNICAL REPORT
W05

submitted for the Solo Project in Semester 2 – First Year

IT DEPARTMENT

Computer Science

by

Phan Tuan Kiet

Full name: Phan Tuan Kiet

ID: 23125062

Class: 23APCS2

Tasks achieved: 04/04

Lecturer:
Professor Dinh Ba Tien (PhD)
Teaching Assistants:
Mr. Ho Tuan Thanh (MSc)
Mr. Nguyen Le Hoang Dung (MSc)

2024

Contents

I) Structure

- *There are four main ".py" files:*

+ function.py: Created to load functions into the main py file.

+ main.py: Developed to create a user interface in the console and integrate the entire program.

+ var.py: Used to define the RouteVar class and RouteVarQuery class to solve tasks and design essential classes.

- *JSON files:*

+ Include stops.json and vars.json to read data.

- *Output folder:*

+ Contains .csv files and .json files files to view results and resolve queries.

- *Drawio.png:*

+ Class Diagram depicting RouteVar, RouteVarQuery, Stop, and StopQuery classes.

- Library used in the program: JSON, CSV, and pandas. These libraries serve the role of writing and reading JSON / CSV files.

II) Comments and Initial Approach

A) RouteVar and RouteVarQuery

- In the vars.json file, each line is an object JSON containing different attributes of a route. There are two vars of a route, so we need to keep track of each routeVar such as an object of the class RouteVar with all the attributes of a route.

- Notice that the outbound attribute of the var.json has the boolean type of true and false, but in Python, the booleans are True and False. Therefore, it is easy to see that we convert to a string of the attributes we want to find when we query tasks for convenience.

B) Stop and StopQuery

- In the stops.json file, there are still lots of JSON objects. However, the JSON format is different from the vars.json as each is formatted in the type like: {Stop: {List Stop}, RouteId, RouteVarId}. In this way, we will still read each StopId as an object of the class Stop with all the attributes in the stops.json including the RouteId and RouteVarId. By doing this, we can easily query the StopId we want to find information about.

- There is a special case in the stops.json: the ward type has the null value, which is the None in Python. Therefore we have to handle this flexibly in the function when we query tasks.

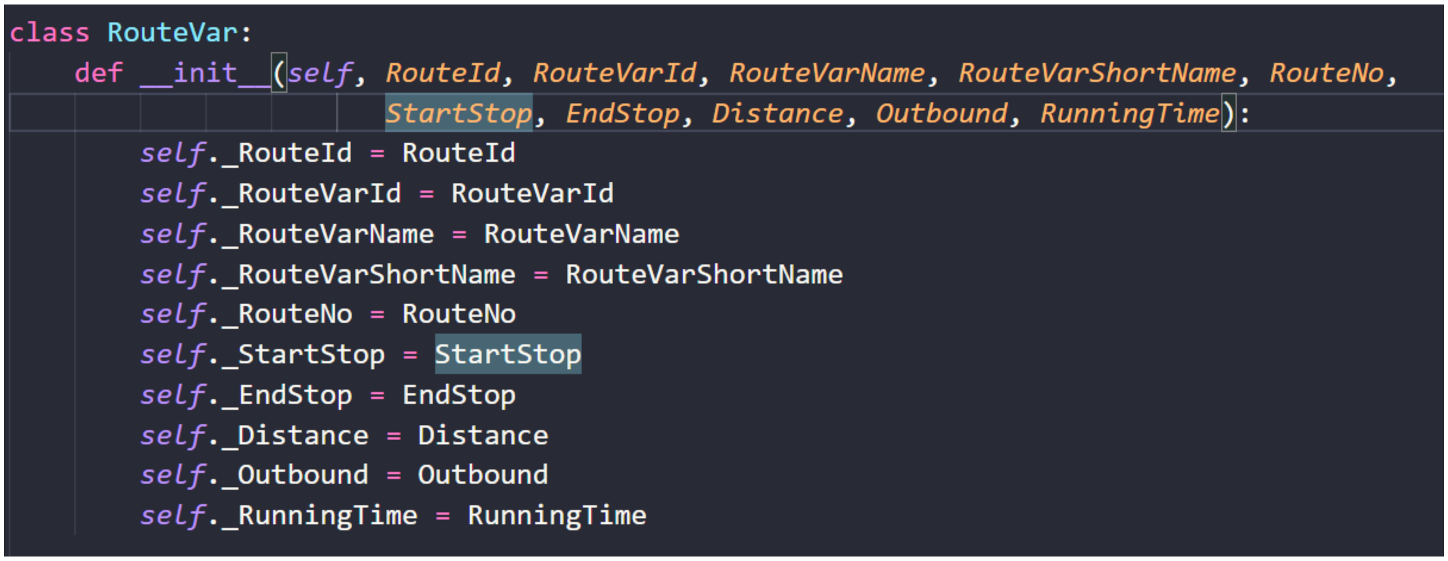
III) Process of handling data and making class

A) RouteVar and RouVarQuery

a) RouteVar

- Name of class: RouteVar.

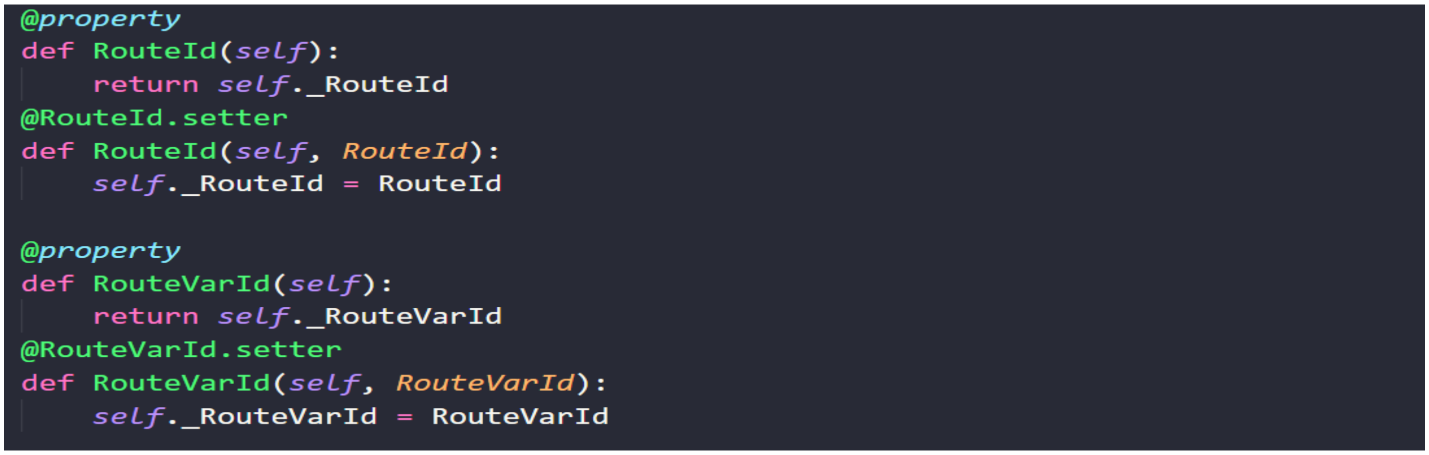
- Constructor:



```
class RouteVar:
    def __init__(self, RouteId, RouteVarId, RouteVarName, RouteVarShortName, RouteNo,
                    StartStop, EndStop, Distance, Outbound, RunningTime):
        self._RouteId = RouteId
        self._RouteVarId = RouteVarId
        self._RouteVarName = RouteVarName
        self._RouteVarShortName = RouteVarShortName
        self._RouteNo = RouteNo
        self._StartStop = StartStop
        self._EndStop = EndStop
        self._Distance = Distance
        self._Outbound = Outbound
        self._RunningTime = RunningTime
```

+ Note that for each attribute, we want to make it protected so that only inside the class and its subclasses can use this attribute. Therefore, we can use _ as a convention of private in Python. People can still access it if they know the way, but this is the convention so we can ensure that the attributes will be protected.

- Using @property and get-set method:



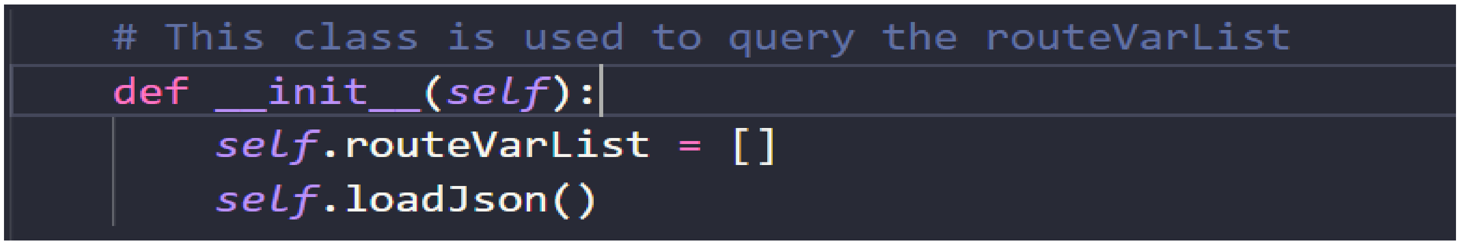
```
@property
def RouteId(self):
    return self._RouteId
@RouteId.setter
def RouteId(self, RouteId):
    self._RouteId = RouteId

@property
def RouteVarId(self):
    return self._RouteVarId
@RouteVarId.setter
def RouteVarId(self, RouteVarId):
    self._RouteVarId = RouteVarId
```

+ We can use a really fast way to make the get-set method for every attribute which is the @property. Example: For the def RouteId(self) → This is the get method, and for the def RouteId(self, RouteId) → This is the set method using property decorations.

b) RouteVarQuery

- Name of the class: RouteVarQuery()

- Constructor:

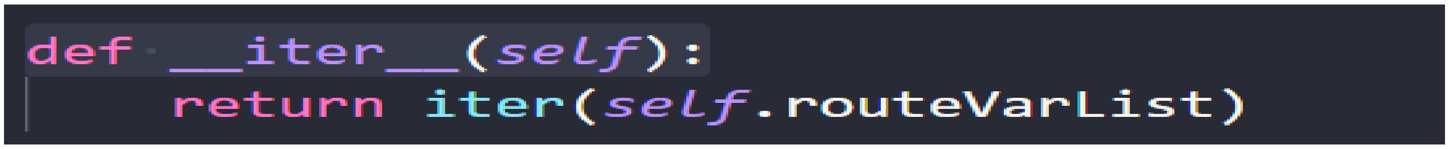


```
    # This class is used to query the routeVarList
    def __init__(self):
        self.routeVarList = []
        self.loadJson()
```

+ When we create an object with the class RouteVarQuery, we will initially make a routeVarList to store RouteVar objects.

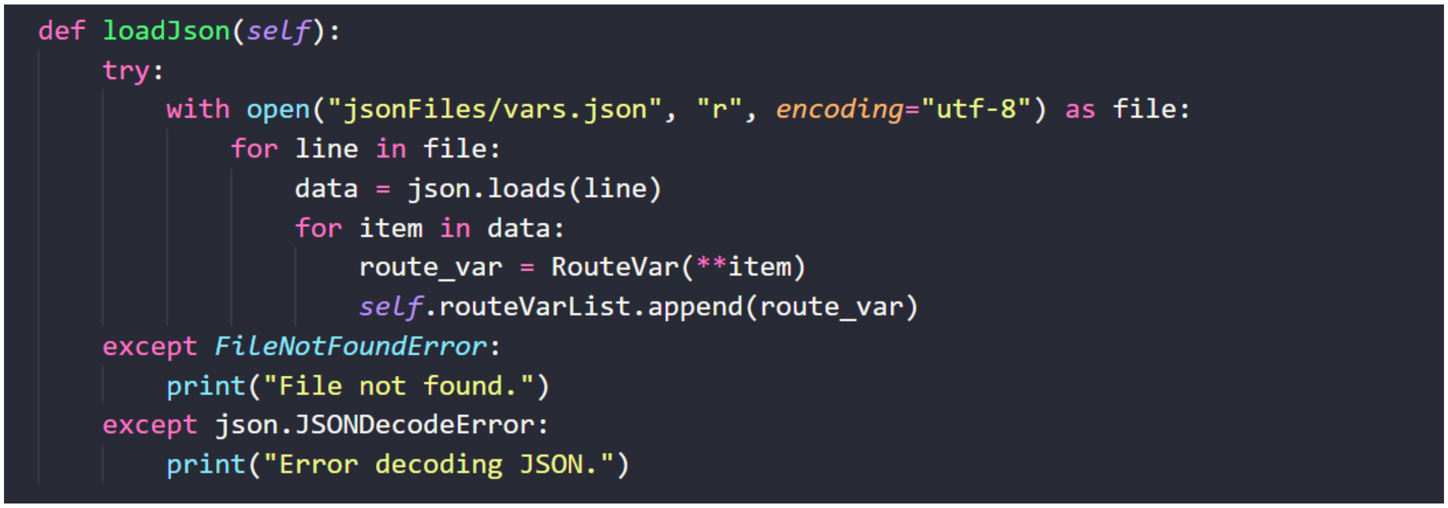
+ Also, we will need to load the JSON files when opening the object.

- Functions describing:



```
def __iter__(self):
    return iter(self.routeVarList)
```

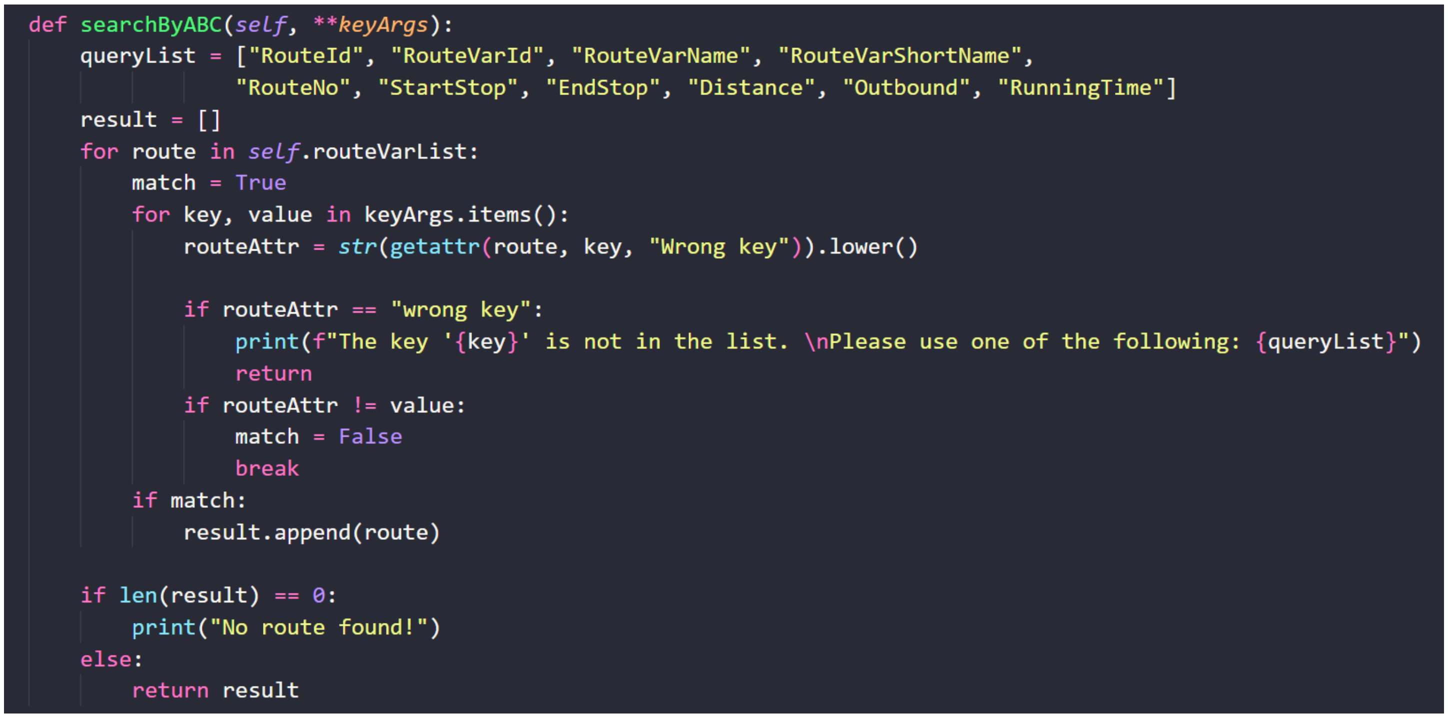
+ Make the routeVarList iterateable.



```
def loadJson(self):
    try:
        with open("jsonFiles/vars.json", "r", encoding="utf-8") as file:
            for line in file:
                data = json.loads(line)
                for item in data:
                    route_var = RouteVar(**item)
                    self.routeVarList.append(route_var)
    except FileNotFoundError:
        print("File not found.")
    except json.JSONDecodeError:
        print("Error decoding JSON.")
```

+ This function is used to load vars.json. As I mentioned above, each line is an object JSON. Therefore, we have to load it as data, and then for an item in data, we will make route_var as an object of RouteVar by using

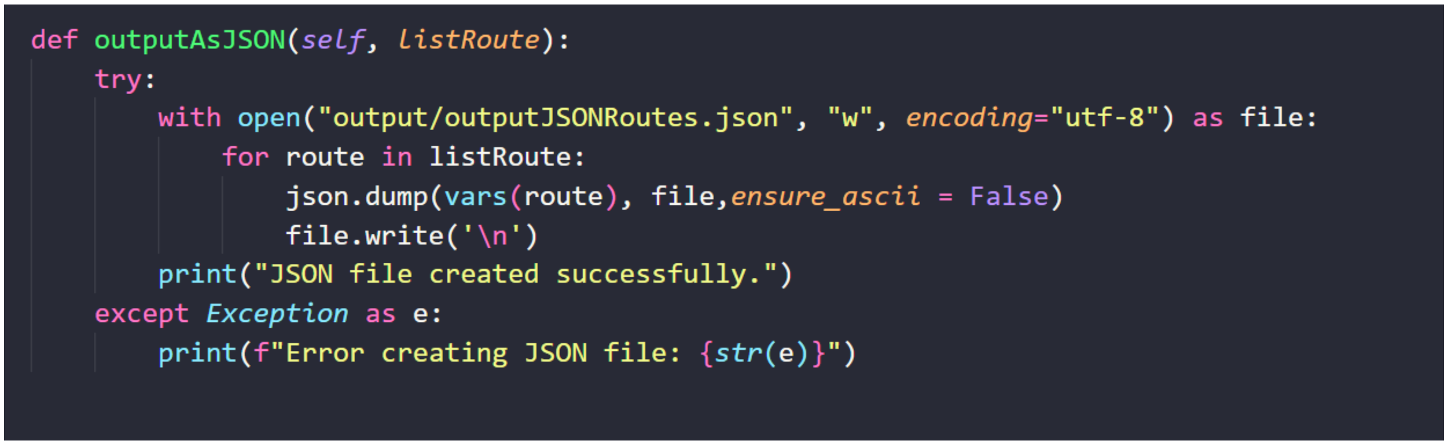
RouteVar(**items). “**” is used to make the list become keyword-arguments and then it is transformed into RouteVar class as an object.



```python
def searchByABC(self, **keyArgs):
    queryList = ["RouteId", "RouteVarId", "RouteVarName", "RouteVarShortName",
                 "RouteNo", "StartStop", "EndStop", "Distance", "Outbound", "RunningTime"]
    result = []
    for route in self.routeVarList:
        match = True
        for key, value in keyArgs.items():
            routeAttr = str(getattr(route, key, "Wrong key")).lower()

            if routeAttr == "wrong key":
                print(f"The key '{key}' is not in the list. \nPlease use one of the following: {queryList}")
                return
            if routeAttr != value:
                match = False
                break
        if match:
            result.append(route)

    if len(result) == 0:
        print("No route found!")
    else:
        return result
```

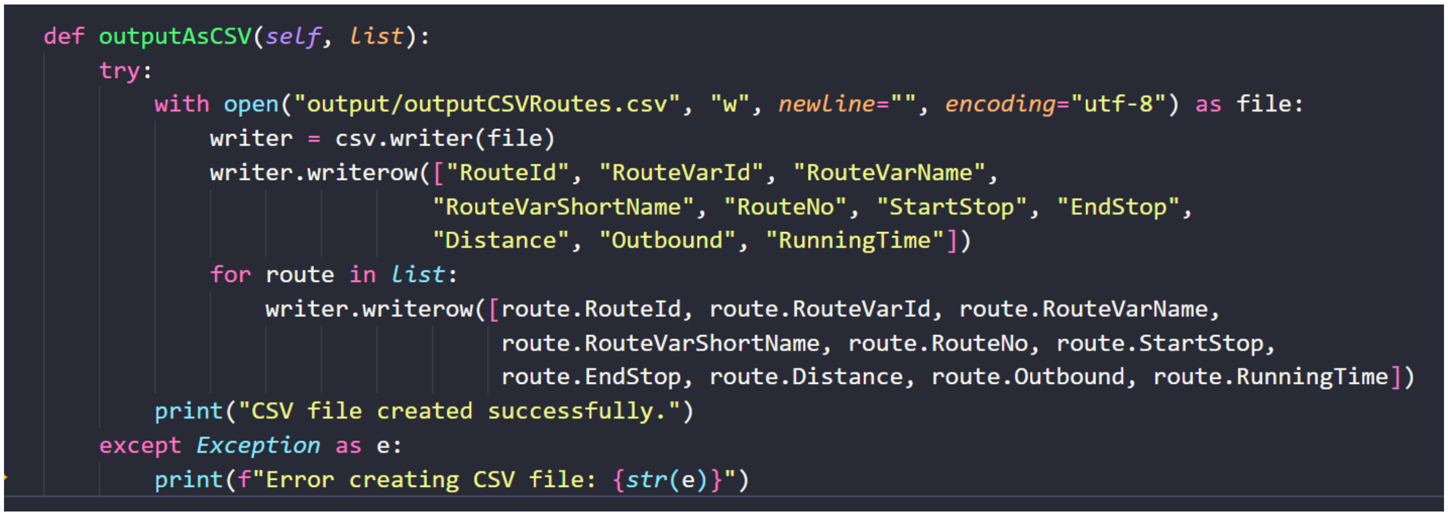
+ The hardest part of the program is making this function. For each route in routeVarList (or each object RouteVar) that we initially created above, we will search for the value in the Queries that the user enters. If that object is suitable for all of the attributes of the queries, it will be appended to the result. In the end, we return the result.

+ Notice that for the queries, we have to put the correct key value (key in the query list). We can do this for a lot of queries. For example, we want to find the RouteId:3 and OutBound: True or RunningTime: 70.



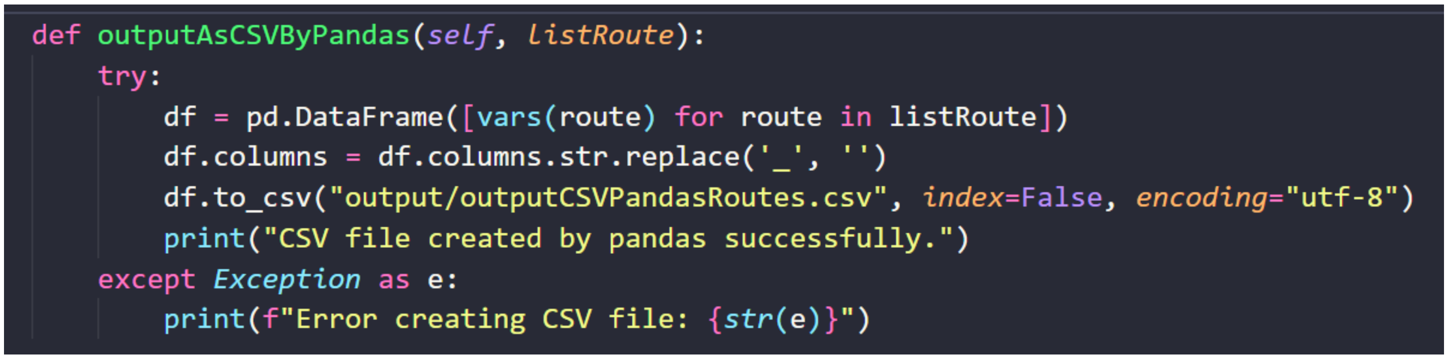
```python
def outputAsJSON(self, listRoute):
    try:
        with open("output/outputJSONRoutes.json", "w", encoding="utf-8") as file:
            for route in listRoute:
                json.dump(vars(route), file,ensure_ascii = False)
                file.write('\n')
        print("JSON file created successfully.")
    except Exception as e:
        print(f"Error creating JSON file: {str(e)}")
```

+ This function is used to output JSON files. Notice that we have to make the ensure_acsii = False because there is Unicode in our file!



```python
def outputAsCSV(self, list):
    try:
        with open("output/outputCSVRoutes.csv", "w", newline="", encoding="utf-8") as file:
            writer = csv.writer(file)
            writer.writerow(["RouteId", "RouteVarId", "RouteVarName",
                                "RouteVarShortName", "RouteNo", "StartStop", "EndStop",
                                "Distance", "Outbound", "RunningTime"])
            for route in list:
                writer.writerow([route.RouteId, route.RouteVarId, route.RouteVarName,
                                    route.RouteVarShortName, route.RouteNo, route.StartStop,
                                    route.EndStop, route.Distance, route.Outbound, route.RunningTime])
        print("CSV file created successfully.")
    except Exception as e:
        print(f"Error creating CSV file: {str(e)}")
```

+ This function is used to output CSV files. Each column is an attribute and we will put data in each cell for the result that we find in SearchByABC



```python
def outputAsCSVByPandas(self, listRoute):
    try:
        df = pd.DataFrame([vars(route) for route in listRoute])
        df.columns = df.columns.str.replace('_', '')
        df.to_csv("output/outputCSVPandasRoutes.csv", index=False, encoding="utf-8")
        print("CSV file created by pandas successfully.")
    except Exception as e:
        print(f"Error creating CSV file: {str(e)}")
```

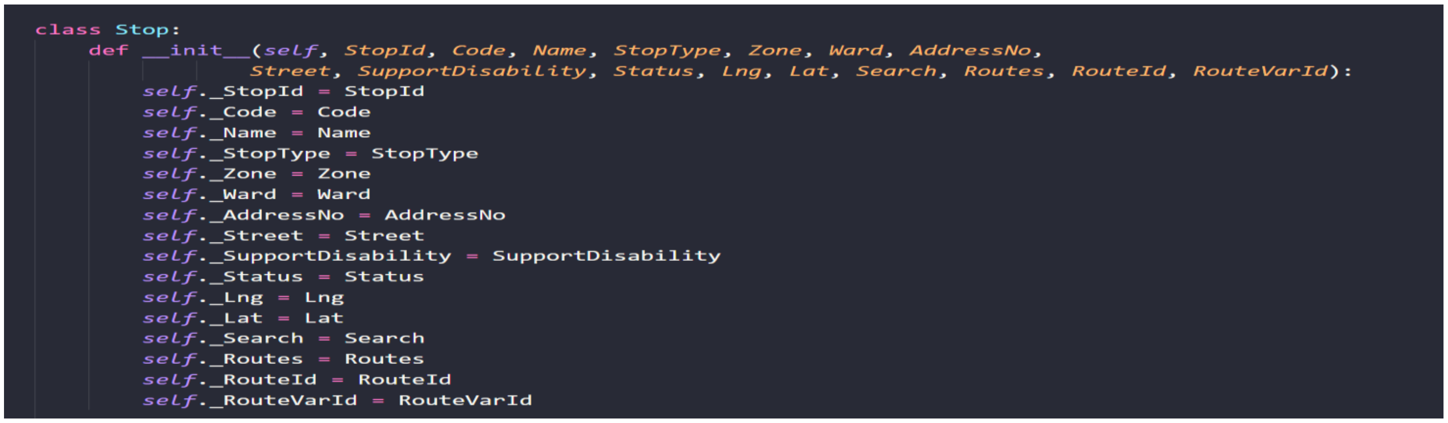
+ Still using to output CSV but by pandas library. We can make the data out in Excel format with numbers in rows and columns. However, it is not nice when implementing it in Python with this data so I don't put it here.

B) Stop and StopQuery

a) Stop

- Name class: Stop.

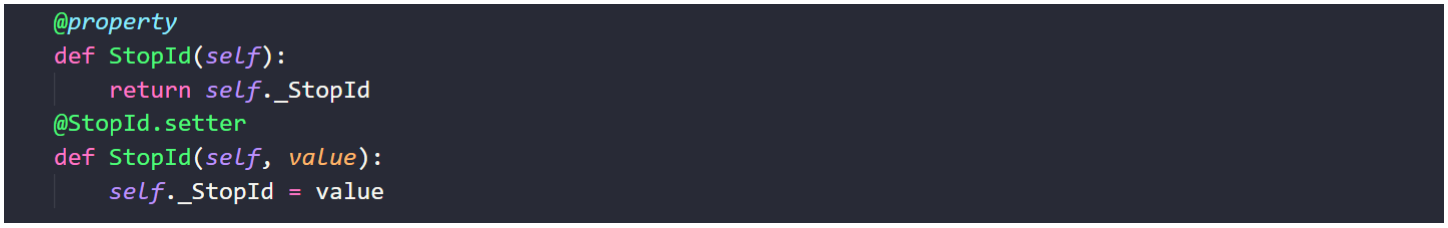
- Constructor:



```python
class Stop:
    def __init__(self, StopId, Code, Name, StopType, Zone, Ward, AddressNo,
                 Street, SupportDisability, Status, Lng, Lat, Search, Routes, RouteId, RouteVarId):
        self._StopId = StopId
        self._Code = Code
        self._Name = Name
        self._StopType = StopType
        self._Zone = Zone
        self._Ward = Ward
        self._AddressNo = AddressNo
        self._Street = Street
        self._SupportDisability = SupportDisability
        self._Status = Status
        self._Lng = Lng
        self._Lat = Lat
        self._Search = Search
        self._Routes = Routes
        self._RouteId = RouteId
        self._RouteVarId = RouteVarId
```

+ For each object Stop, these are the attributes they will have. As I mentioned above, all attributes will be protected so that only inside the class and its subclasses can be used.

- Using @property and get-set method:



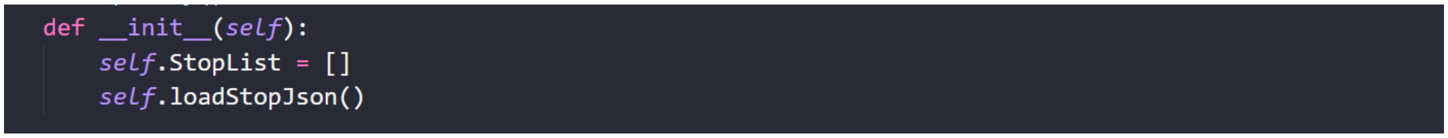
```
@property
def StopId(self):
    return self._StopId
@StopId.setter
def StopId(self, value):
    self._StopId = value
```

+ Make get and set method for all other attributes.

b) StopQuery

- Name: StopQuery()

- Constructor:

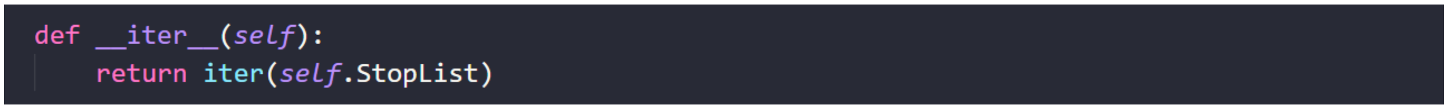


```
def __init__(self):
    self.StopList = []
    self.loadStopJson()
```

+ The same as RouteVarQuery, creating a StopList holding Stop objects.

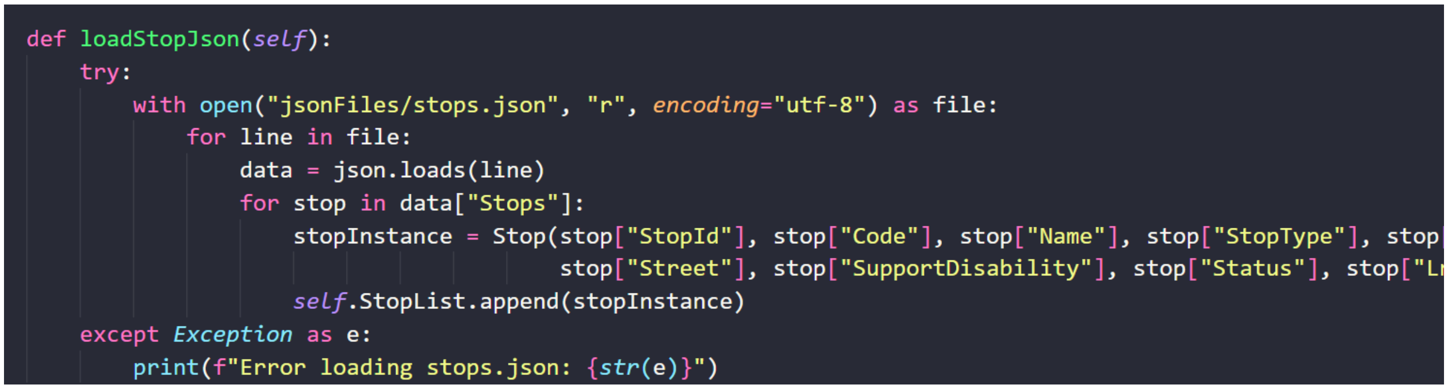
+ Load stops.json file initially.

- Functions Descriptions:



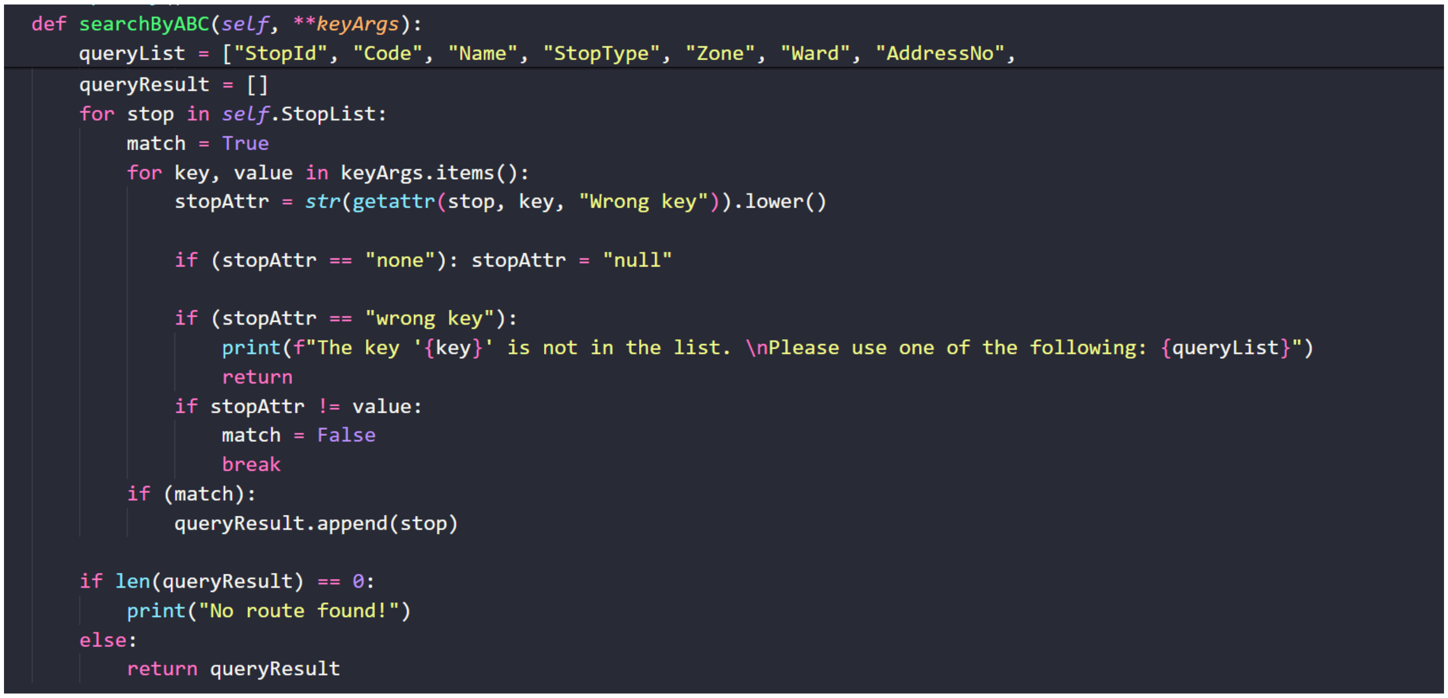
```
def __iter__(self):
    return iter(self.StopList)
```

+ Make the StopList can iterate.



```
def loadStopJson(self):
    try:
        with open("jsonFiles/stops.json", "r", encoding="utf-8") as file:
            for line in file:
                data = json.loads(line)
                for stop in data["Stops"]:
                    stopInstance = Stop(stop["StopId"], stop["Code"], stop["Name"], stop["StopType"], stop
                                        stop["Street"], stop["SupportDisability"], stop["Status"], stop["L
                    self.StopList.append(stopInstance)
    except Exception as e:
        print(f"Error loading stops.json: {str(e)}")
```

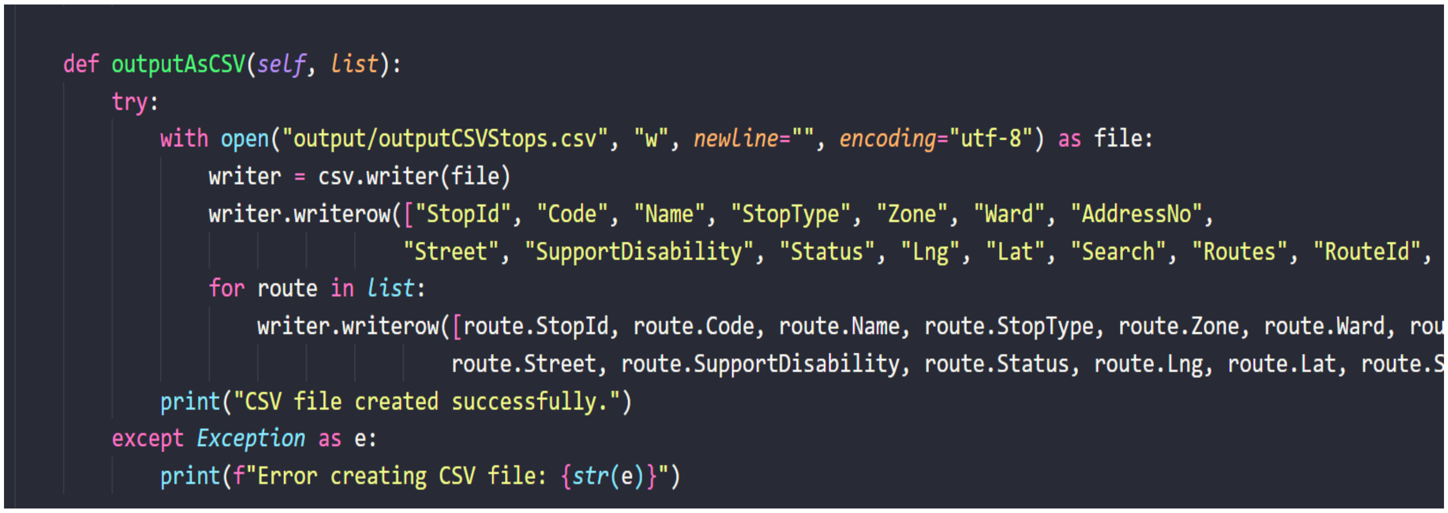
+ Load stops.json. It is special that we have to look for each attribute in the “Stops” of a RouteVar and RouteVarId.



```
def searchByABC(self, **keyArgs):
    queryList = ["StopId", "Code", "Name", "StopType", "Zone", "Ward", "AddressNo",
    queryResult = []
    for stop in self.StopList:
        match = True
        for key, value in keyArgs.items():
            stopAttr = str(getattr(stop, key, "Wrong key")).lower()

            if (stopAttr == "none"): stopAttr = "null"

            if (stopAttr == "wrong key"):
                print(f"The key '{key}' is not in the list. \nPlease use one of the following: {queryList}")
                return
            if stopAttr != value:
                match = False
                break
        if (match):
            queryResult.append(stop)

    if len(queryResult) == 0:
        print("No route found!")
    else:
        return queryResult
```

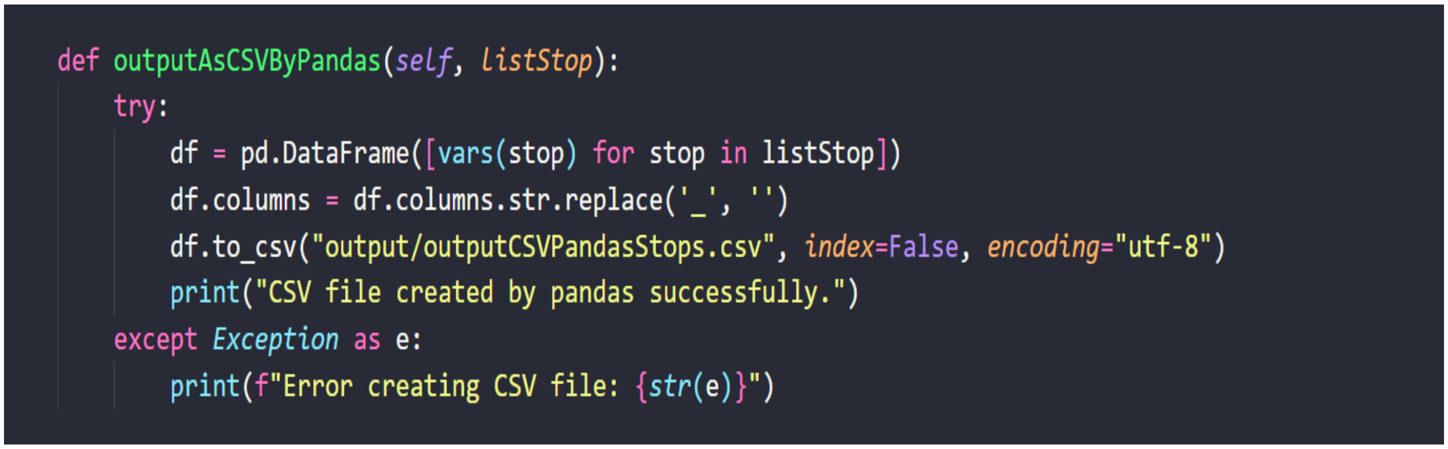
+ Quite the same as RouteVarQuery(). **keysArgs is used to unpack the list as key–value (queries that we input). If any stops are correct for all of the values that we input, we append it to the result list and return it.

+ If there are no suitable stops. We will print “No Stops Found” or “No Routes Found”.



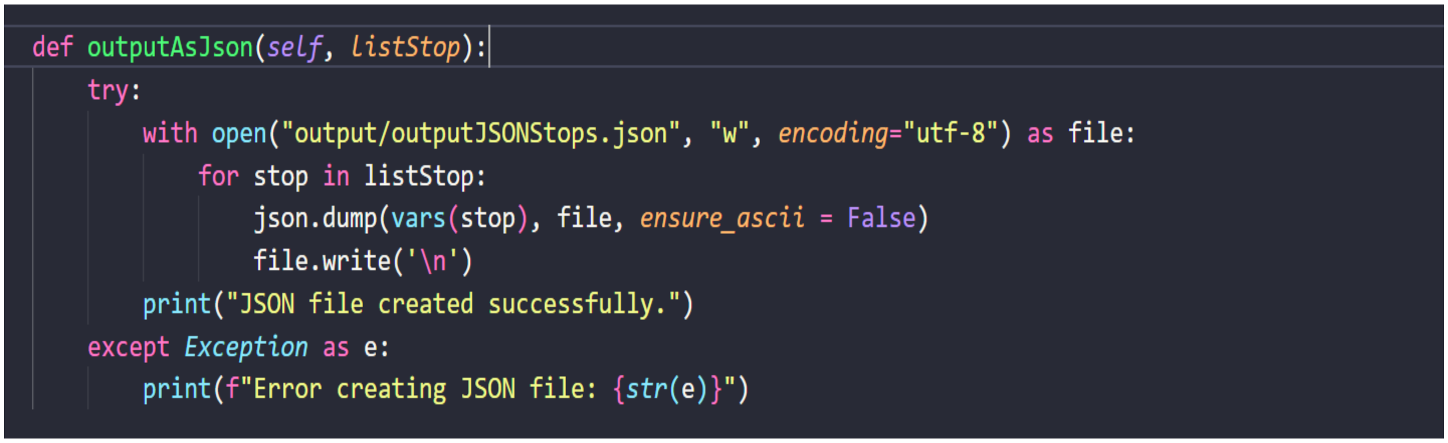
```
def outputAsCSV(self, list):
    try:
        with open("output/outputCSVStops.csv", "w", newline="", encoding="utf-8") as file:
            writer = csv.writer(file)
            writer.writerow(["StopId", "Code", "Name", "StopType", "Zone", "Ward", "AddressNo",
                            "Street", "SupportDisability", "Status", "Lng", "Lat", "Search", "Routes", "RouteId",
            for route in list:
                writer.writerow([route.StopId, route.Code, route.Name, route.StopType, route.Zone, route.Ward, rou
                                route.Street, route.SupportDisability, route.Status, route.Lng, route.Lat, route.S
        print("CSV file created successfully.")
    except Exception as e:
        print(f"Error creating CSV file: {str(e)}")
```

+ Output the result list as CSV.



```
def outputAsCSVByPandas(self, listStop):
    try:
        df = pd.DataFrame([vars(stop) for stop in listStop])
        df.columns = df.columns.str.replace('_', '')
        df.to_csv("output/outputCSVPandasStops.csv", index=False, encoding="utf-8")
        print("CSV file created by pandas successfully.")
    except Exception as e:
        print(f"Error creating CSV file: {str(e)}")
```

+ Output the result list as CSV but by pandas library.



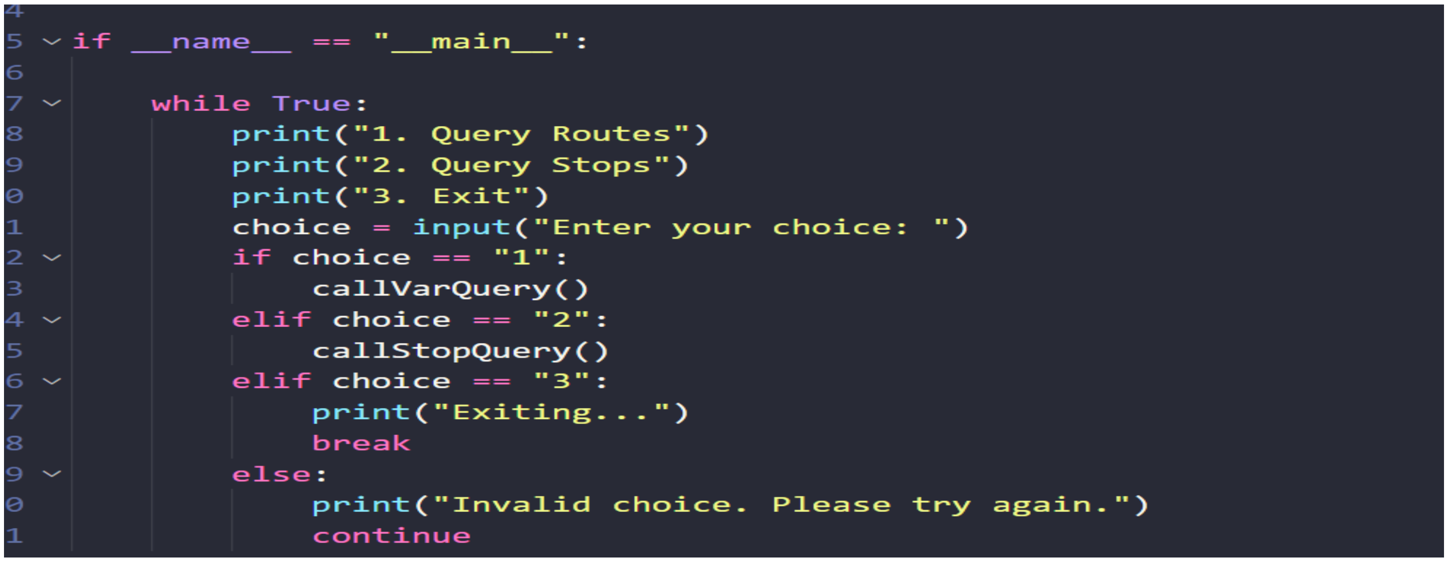
```
def outputAsJson(self, listStop):
    try:
        with open("output/outputJSONStops.json", "w", encoding="utf-8") as file:
            for stop in listStop:
                json.dump(vars(stop), file, ensure_ascii = False)
                file.write('\n')
        print("JSON file created successfully.")
    except Exception as e:
        print(f"Error creating JSON file: {str(e)}")
```

+ Output the result list as a JSON file.

+ When the print is finished, print “JSON file created successfully”.

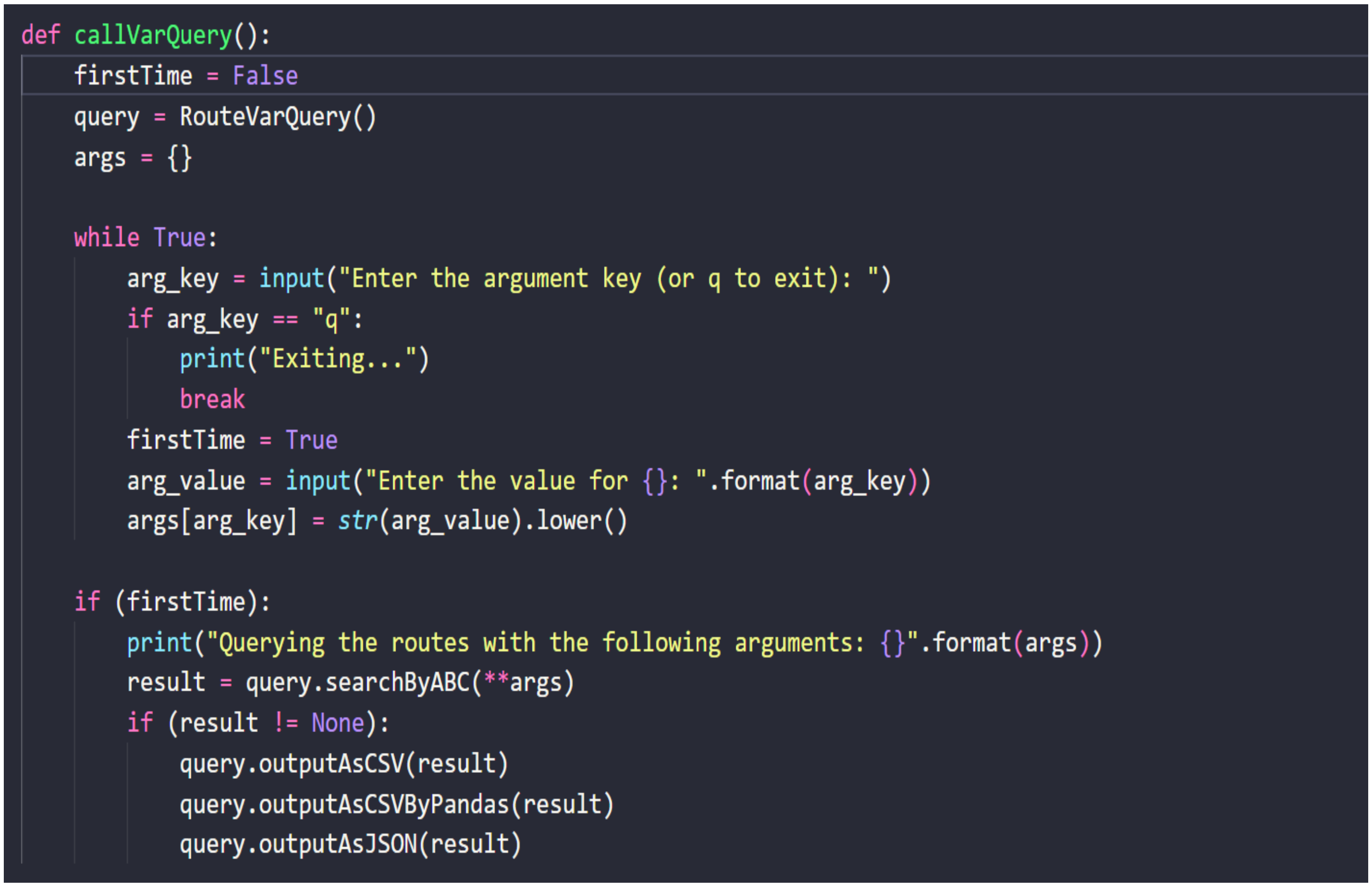
IV) User interface in the terminal and main handling

- In main.py, this is where people will see the UI.

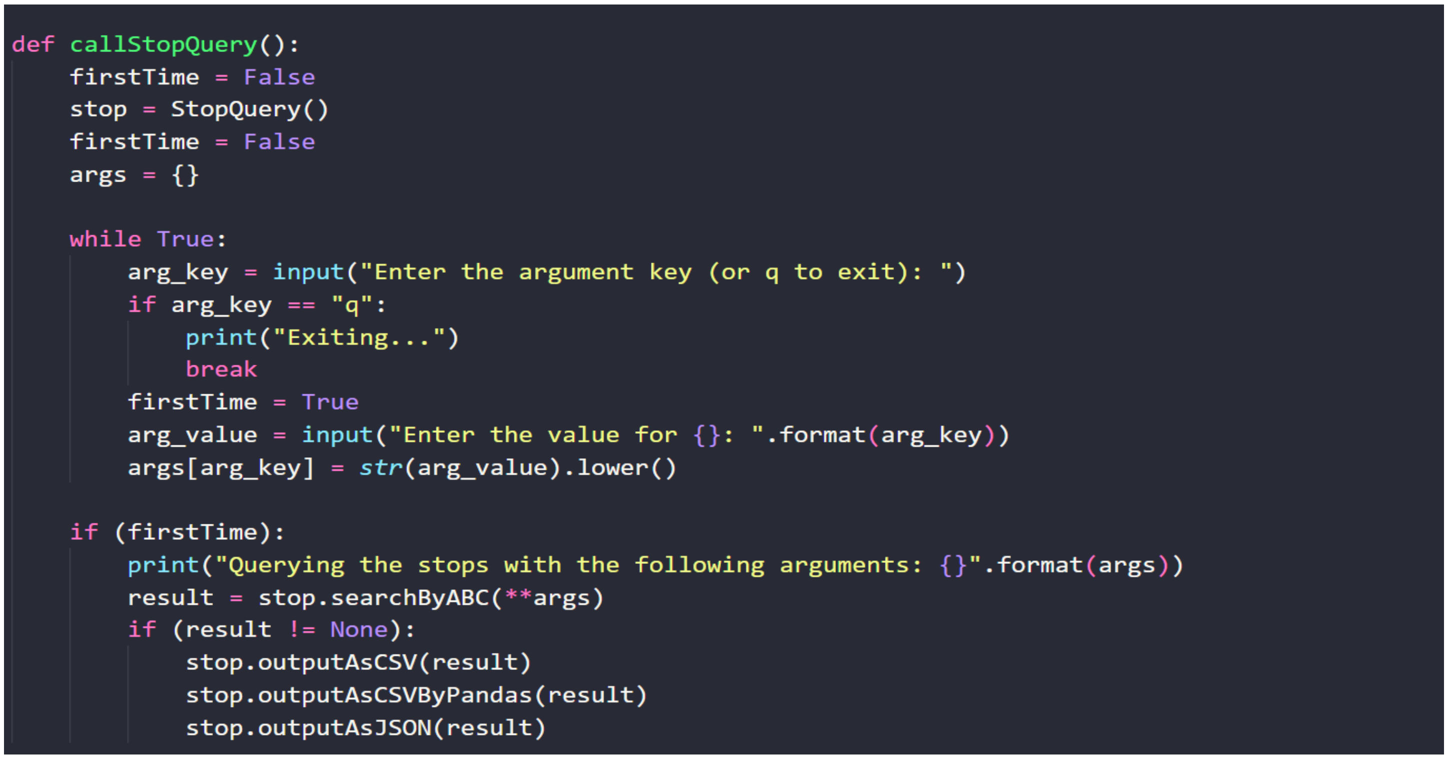


```
if __name__ == "__main__":

    while True:
        print("1. Query Routes")
        print("2. Query Stops")
        print("3. Exit")
        choice = input("Enter your choice: ")
        if choice == "1":
            callVarQuery()
        elif choice == "2":
            callStopQuery()
        elif choice == "3":
            print("Exiting...")
            break
        else:
            print("Invalid choice. Please try again.")
            continue
```

- In function.py, there are some functions used to input the queries of RouteVarId and StopQuery

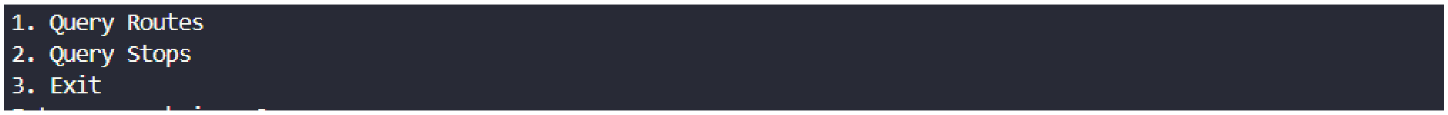


```python
def callVarQuery():
    firstTime = False
    query = RouteVarQuery()
    args = {}

    while True:
        arg_key = input("Enter the argument key (or q to exit): ")
        if arg_key == "q":
            print("Exiting...")
            break
        firstTime = True
        arg_value = input("Enter the value for {}: ".format(arg_key))
        args[arg_key] = str(arg_value).lower()

    if (firstTime):
        print("Querying the routes with the following arguments: {}".format(args))
        result = query.searchByABC(**args)
        if (result != None):
            query.outputAsCSV(result)
            query.outputAsCSVByPandas(result)
            query.outputAsJSON(result)
```

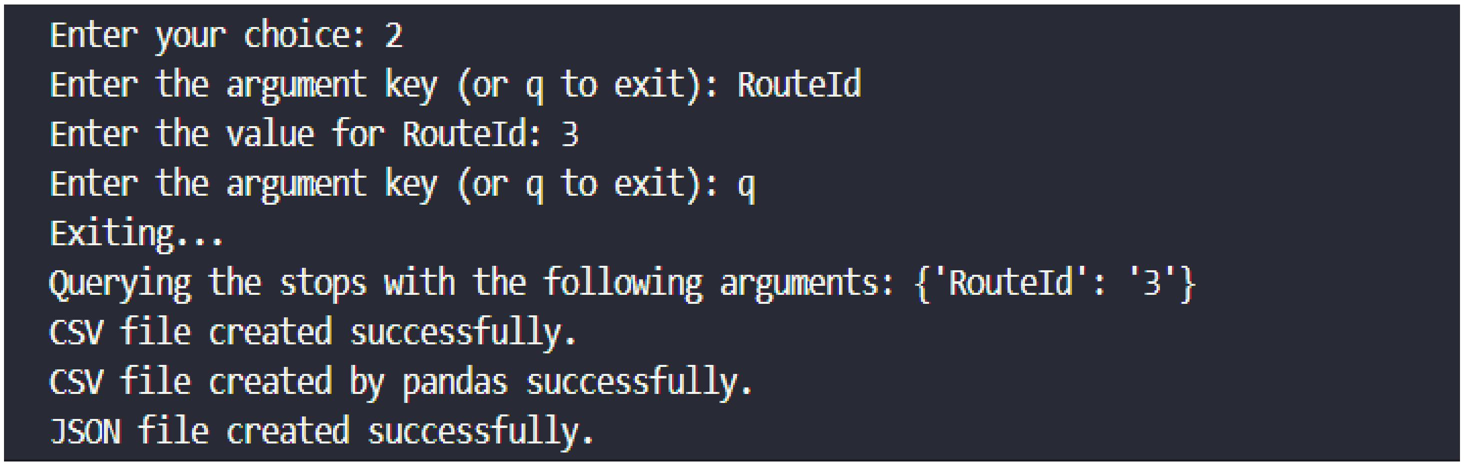


```python
def callStopQuery():
    firstTime = False
    stop = StopQuery()
    firstTime = False
    args = {}

    while True:
        arg_key = input("Enter the argument key (or q to exit): ")
        if arg_key == "q":
            print("Exiting...")
            break
        firstTime = True
        arg_value = input("Enter the value for {}: ".format(arg_key))
        args[arg_key] = str(arg_value).lower()

    if (firstTime):
        print("Querying the stops with the following arguments: {}".format(args))
        result = stop.searchByABC(**args)
        if (result != None):
            stop.outputAsCSV(result)
            stop.outputAsCSVByPandas(result)
            stop.outputAsJSON(result)
```

- How to use it?



```
1. Query Routes
2. Query Stops
3. Exit
```

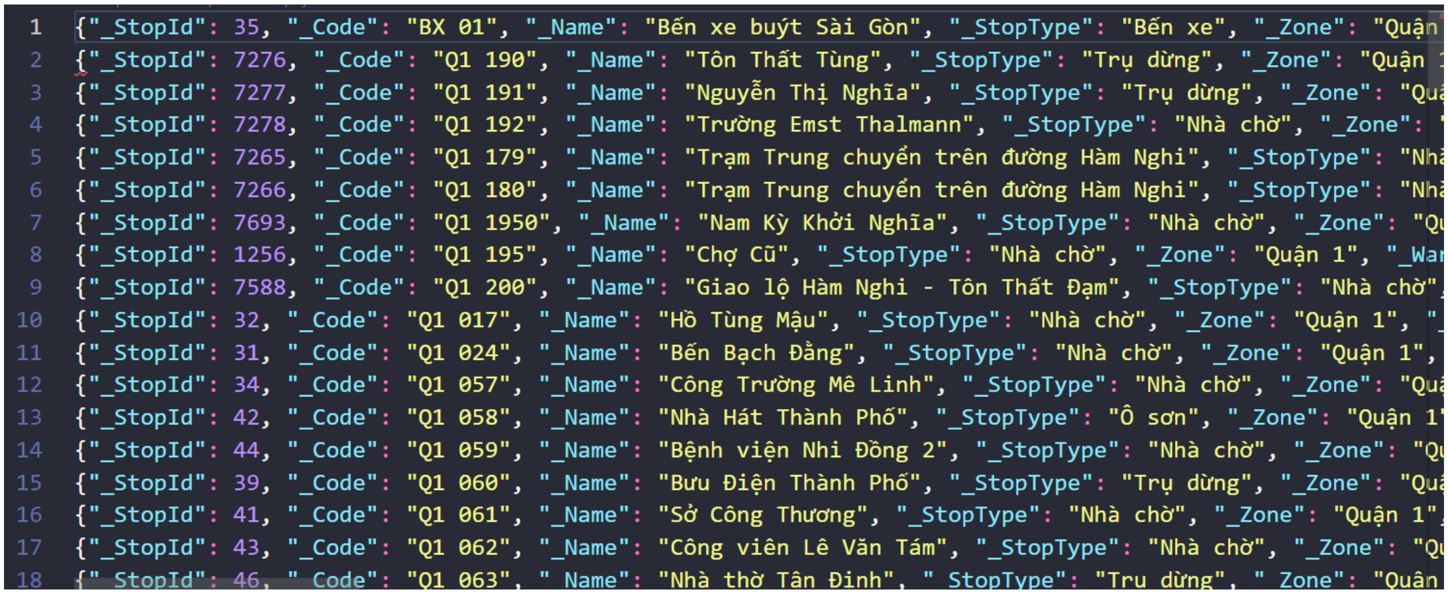


```
Enter your choice: 2
Enter the argument key (or q to exit): RouteId
Enter the value for RouteId: 3
Enter the argument key (or q to exit): q
Exiting...
Querying the stops with the following arguments: {'RouteId': '3'}
CSV file created successfully.
CSV file created by pandas successfully.
JSON file created successfully.
```

+ You can input as many queries as you want as long as those are corrected attributes of stops/vars.

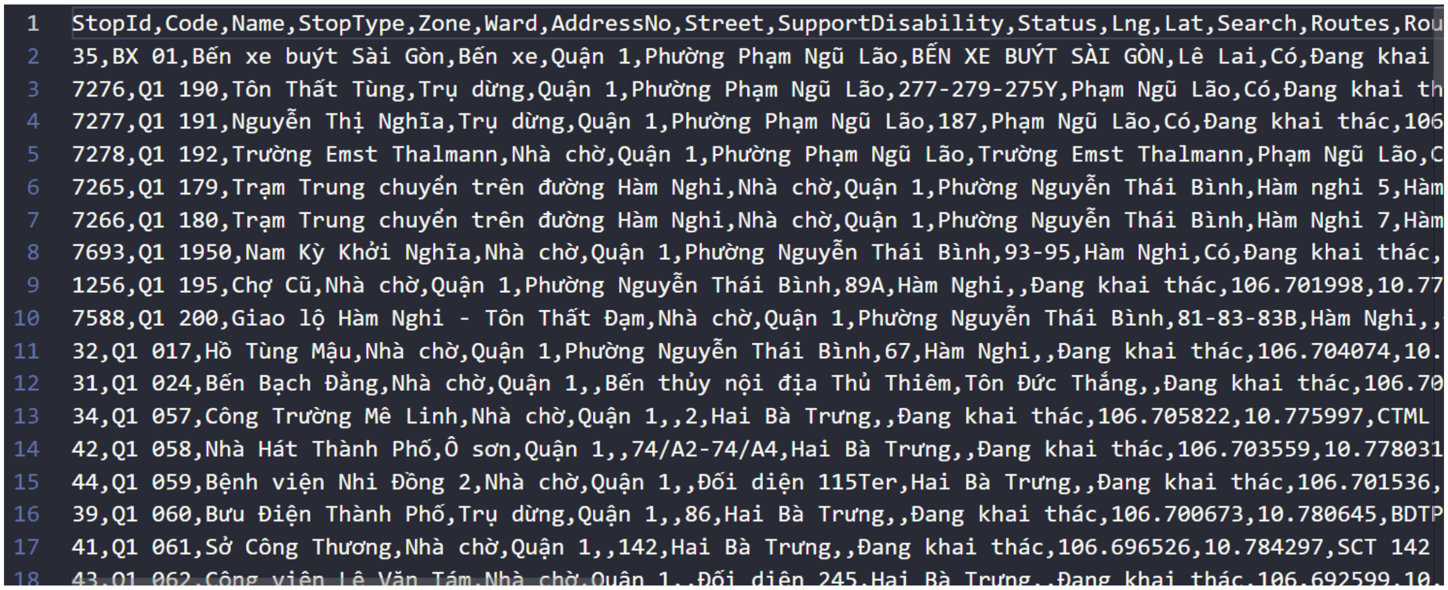
- Result:

+ JSON file:



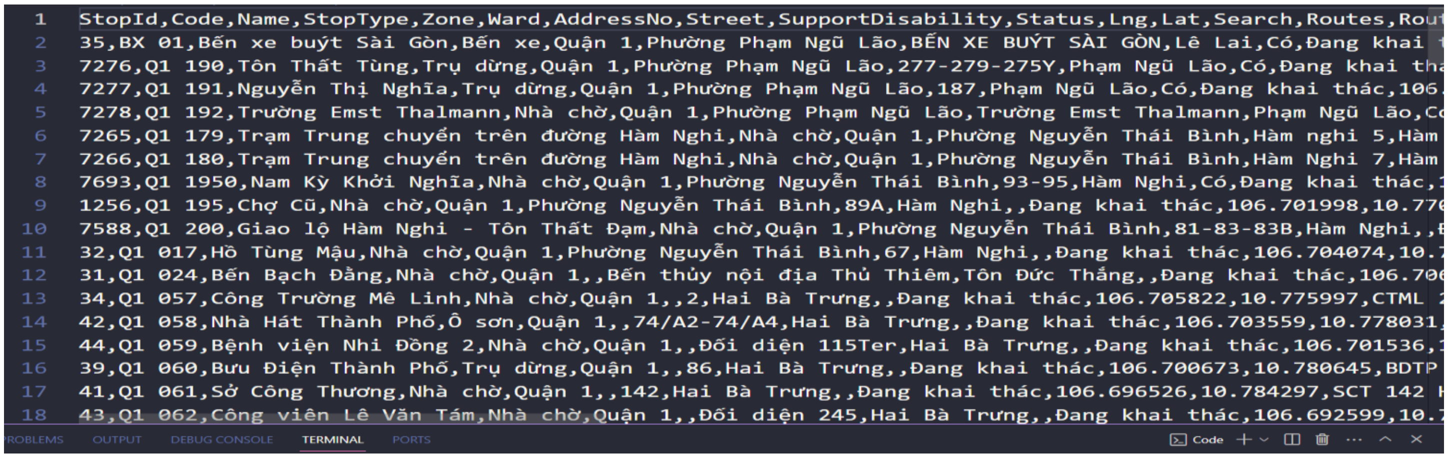
```
{"_StopId": 35, "_Code": "BX 01", "_Name": "Bến xe buýt Sài Gòn", "_StopType": "Bến xe", "_Zone": "Quận
{"_StopId": 7276, "_Code": "Q1 190", "_Name": "Tôn Thất Tùng", "_StopType": "Trụ dừng", "_Zone": "Quận 
{"_StopId": 7277, "_Code": "Q1 191", "_Name": "Nguyễn Thị Nghĩa", "_StopType": "Trụ dừng", "_Zone": "Qu
{"_StopId": 7278, "_Code": "Q1 192", "_Name": "Trường Emst Thalmann", "_StopType": "Nhà chờ", "_Zone": 
{"_StopId": 7265, "_Code": "Q1 179", "_Name": "Trạm Trung chuyển trên đường Hàm Nghi", "_StopType": "Nh
{"_StopId": 7266, "_Code": "Q1 180", "_Name": "Trạm Trung chuyển trên đường Hàm Nghi", "_StopType": "Nh
{"_StopId": 7693, "_Code": "Q1 1950", "_Name": "Nam Kỳ Khởi Nghĩa", "_StopType": "Nhà chờ", "_Zone": "Q
{"_StopId": 1256, "_Code": "Q1 195", "_Name": "Chợ Cũ", "_StopType": "Nhà chờ", "_Zone": "Quận 1", "_Wa
{"_StopId": 7588, "_Code": "Q1 200", "_Name": "Giao lộ Hàm Nghi - Tôn Thất Đạm", "_StopType": "Nhà chờ"
{"_StopId": 32, "_Code": "Q1 017", "_Name": "Hồ Tùng Mậu", "_StopType": "Nhà chờ", "_Zone": "Quận 1", "
{"_StopId": 31, "_Code": "Q1 024", "_Name": "Bến Bạch Đằng", "_StopType": "Nhà chờ", "_Zone": "Quận 1",
{"_StopId": 34, "_Code": "Q1 057", "_Name": "Công Trường Mê Linh", "_StopType": "Nhà chờ", "_Zone": "Qu
{"_StopId": 42, "_Code": "Q1 058", "_Name": "Nhà Hát Thành Phố", "_StopType": "Ô sơn", "_Zone": "Quận 1
{"_StopId": 44, "_Code": "Q1 059", "_Name": "Bệnh viện Nhi Đồng 2", "_StopType": "Nhà chờ", "_Zone": "Q
{"_StopId": 39, "_Code": "Q1 060", "_Name": "Bưu Điện Thành Phố", "_StopType": "Trụ dừng", "_Zone": "Qu
{"_StopId": 41, "_Code": "Q1 061", "_Name": "Sở Công Thương", "_StopType": "Nhà chờ", "_Zone": "Quận 1"
{"_StopId": 43, "_Code": "Q1 062", "_Name": "Công viên Lê Văn Tám", "_StopType": "Nhà chờ", "_Zone": "Q
{"_StopId": 46, "_Code": "Q1 063", "_Name": "Nhà thờ Tân Định", "_StopType": "Trụ dừng", "_Zone": "Quận
```

+ CSV file:



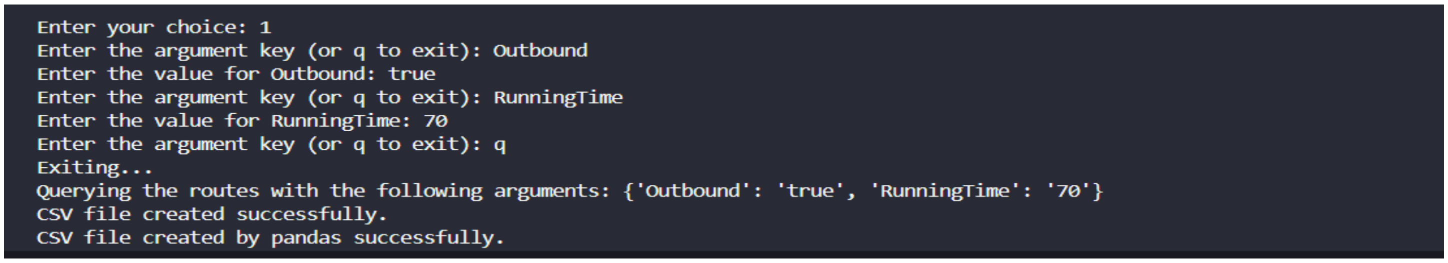
```
StopId,Code,Name,StopType,Zone,Ward,AddressNo,Street,SupportDisability,Status,Lng,Lat,Search,Routes,Rou
35,BX 01,Bến xe buýt Sài Gòn,Bến xe,Quận 1,Phường Phạm Ngũ Lão,BẾN XE BUÝT SÀI GÒN,Lê Lai,Có,Đang khai
7276,Q1 190,Tôn Thất Tùng,Trụ dừng,Quận 1,Phường Phạm Ngũ Lão,277-279-275Y,Phạm Ngũ Lão,Có,Đang khai th
7277,Q1 191,Nguyễn Thị Nghĩa,Trụ dừng,Quận 1,Phường Phạm Ngũ Lão,187,Phạm Ngũ Lão,Có,Đang khai thác,106
7278,Q1 192,Trường Emst Thalmann,Nhà chờ,Quận 1,Phường Phạm Ngũ Lão,Trường Emst Thalmann,Phạm Ngũ Lão,C
7265,Q1 179,Trạm Trung chuyển trên đường Hàm Nghi,Nhà chờ,Quận 1,Phường Nguyễn Thái Bình,Hàm nghi 5,Hàm
7266,Q1 180,Trạm Trung chuyển trên đường Hàm Nghi,Nhà chờ,Quận 1,Phường Nguyễn Thái Bình,Hàm Nghi 7,Hàm
7693,Q1 1950,Nam Kỳ Khởi Nghĩa,Nhà chờ,Quận 1,Phường Nguyễn Thái Bình,93-95,Hàm Nghi,Có,Đang khai thác,
1256,Q1 195,Chợ Cũ,Nhà chờ,Quận 1,Phường Nguyễn Thái Bình,89A,Hàm Nghi,,Đang khai thác,106.701998,10.77
7588,Q1 200,Giao lộ Hàm Nghi - Tôn Thất Đạm,Nhà chờ,Quận 1,Phường Nguyễn Thái Bình,81-83-83B,Hàm Nghi,,
32,Q1 017,Hồ Tùng Mậu,Nhà chờ,Quận 1,Phường Nguyễn Thái Bình,67,Hàm Nghi,,Đang khai thác,106.704074,10.
31,Q1 024,Bến Bạch Đằng,Nhà chờ,Quận 1,,Bến thủy nội địa Thủ Thiêm,Tôn Đức Thắng,,Đang khai thác,106.70
34,Q1 057,Công Trường Mê Linh,Nhà chờ,Quận 1,,2,Hai Bà Trưng,,Đang khai thác,106.705822,10.775997,CTML
42,Q1 058,Nhà Hát Thành Phố,Ô sơn,Quận 1,,74/A2-74/A4,Hai Bà Trưng,,Đang khai thác,106.703559,10.778031
44,Q1 059,Bệnh viện Nhi Đồng 2,Nhà chờ,Quận 1,,Đối diện 115Ter,Hai Bà Trưng,,Đang khai thác,106.701536,
39,Q1 060,Bưu Điện Thành Phố,Trụ dừng,Quận 1,,86,Hai Bà Trưng,,Đang khai thác,106.700673,10.780645,BDTP
41,Q1 061,Sở Công Thương,Nhà chờ,Quận 1,,142,Hai Bà Trưng,,Đang khai thác,106.696526,10.784297,SCT 142
43,Q1 062,Công viên Lê Văn Tám,Nhà chờ,Quận 1,,Đối diện 245,Hai Bà Trưng,,Đang khai thác,106.692599,10.
```

+ CSV file by pandas:



```
StopId,Code,Name,StopType,Zone,Ward,AddressNo,Street,SupportDisability,Status,Lng,Lat,Search,Routes,Rou
35,BX 01,Bến xe buýt Sài Gòn,Bến xe,Quận 1,Phường Phạm Ngũ Lão,BẾN XE BUÝT SÀI GÒN,Lê Lai,Có,Đang khai
7276,Q1 190,Tôn Thất Tùng,Trụ dừng,Quận 1,Phường Phạm Ngũ Lão,277-279-275Y,Phạm Ngũ Lão,Có,Đang khai th
7277,Q1 191,Nguyễn Thị Nghĩa,Trụ dừng,Quận 1,Phường Phạm Ngũ Lão,187,Phạm Ngũ Lão,Có,Đang khai thác,106.
7278,Q1 192,Trường Emst Thalmann,Nhà chờ,Quận 1,Phường Phạm Ngũ Lão,Trường Emst Thalmann,Phạm Ngũ Lão,C
7265,Q1 179,Trạm Trung chuyển trên đường Hàm Nghi,Nhà chờ,Quận 1,Phường Nguyễn Thái Bình,Hàm nghi 5,Hàm
7266,Q1 180,Trạm Trung chuyển trên đường Hàm Nghi,Nhà chờ,Quận 1,Phường Nguyễn Thái Bình,Hàm Nghi 7,Hàm
7693,Q1 1950,Nam Kỳ Khởi Nghĩa,Nhà chờ,Quận 1,Phường Nguyễn Thái Bình,93-95,Hàm Nghi,Có,Đang khai thác,
1256,Q1 195,Chợ Cũ,Nhà chờ,Quận 1,Phường Nguyễn Thái Bình,89A,Hàm Nghi,,Đang khai thác,106.701998,10.77
7588,Q1 200,Giao lộ Hàm Nghi - Tôn Thất Đạm,Nhà chờ,Quận 1,Phường Nguyễn Thái Bình,81-83-83B,Hàm Nghi,,
32,Q1 017,Hồ Tùng Mậu,Nhà chờ,Quận 1,Phường Nguyễn Thái Bình,67,Hàm Nghi,,Đang khai thác,106.704074,10.
31,Q1 024,Bến Bạch Đằng,Nhà chờ,Quận 1,,Bến thủy nội địa Thủ Thiêm,Tôn Đức Thắng,,Đang khai thác,106.70
34,Q1 057,Công Trường Mê Linh,Nhà chờ,Quận 1,,2,Hai Bà Trưng,,Đang khai thác,106.705822,10.775997,CTML
42,Q1 058,Nhà Hát Thành Phố,Ô sơn,Quận 1,,74/A2-74/A4,Hai Bà Trưng,,Đang khai thác,106.703559,10.778031
44,Q1 059,Bệnh viện Nhi Đồng 2,Nhà chờ,Quận 1,,Đối diện 115Ter,Hai Bà Trưng,,Đang khai thác,106.701536,
39,Q1 060,Bưu Điện Thành Phố,Trụ dừng,Quận 1,,86,Hai Bà Trưng,,Đang khai thác,106.700673,10.780645,BDTP
41,Q1 061,Sở Công Thương,Nhà chờ,Quận 1,,142,Hai Bà Trưng,,Đang khai thác,106.696526,10.784297,SCT 142
43,Q1 062,Công viên Lê Văn Tám,Nhà chờ,Quận 1,,Đối diện 245,Hai Bà Trưng,,Đang khai thác,106.692599,10.
```

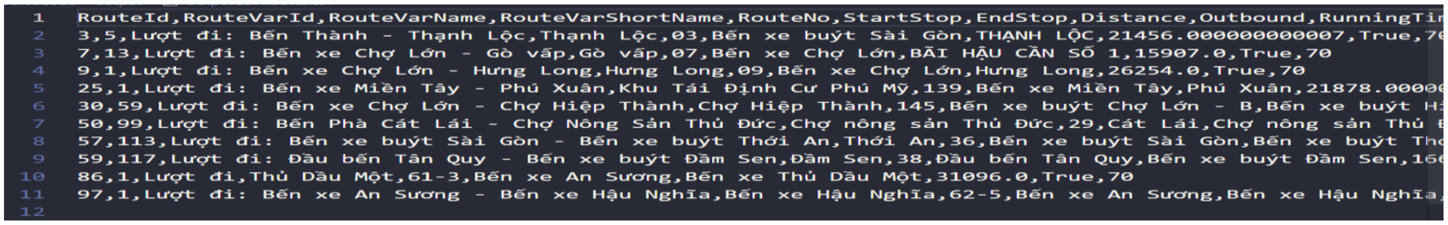
- The same for RouteVarQuery:



```
Enter your choice: 1
Enter the argument key (or q to exit): Outbound
Enter the value for Outbound: true
Enter the argument key (or q to exit): RunningTime
Enter the value for RunningTime: 70
Enter the argument key (or q to exit): q
Exiting...
Querying the routes with the following arguments: {'Outbound': 'true', 'RunningTime': '70'}
CSV file created successfully.
CSV file created by pandas successfully.
```

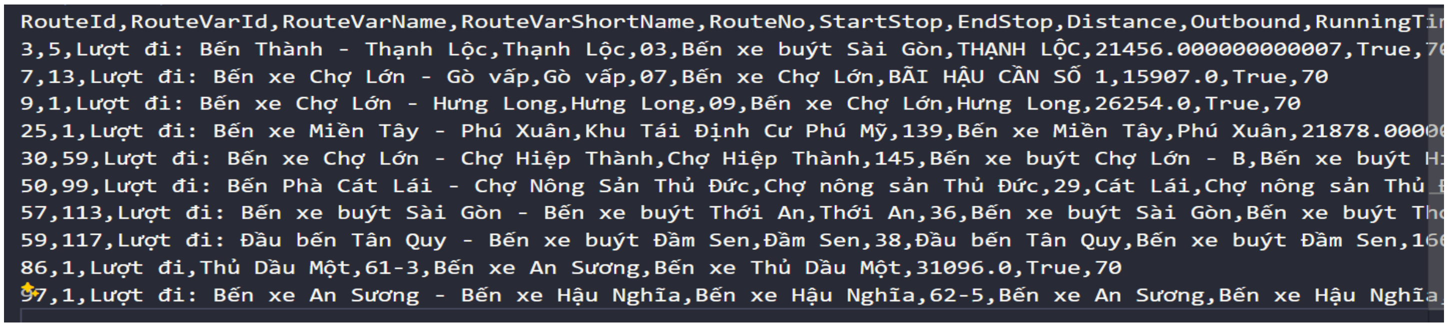
+ Notice that this time I use 2 conditions for a query.

+ CSV file:



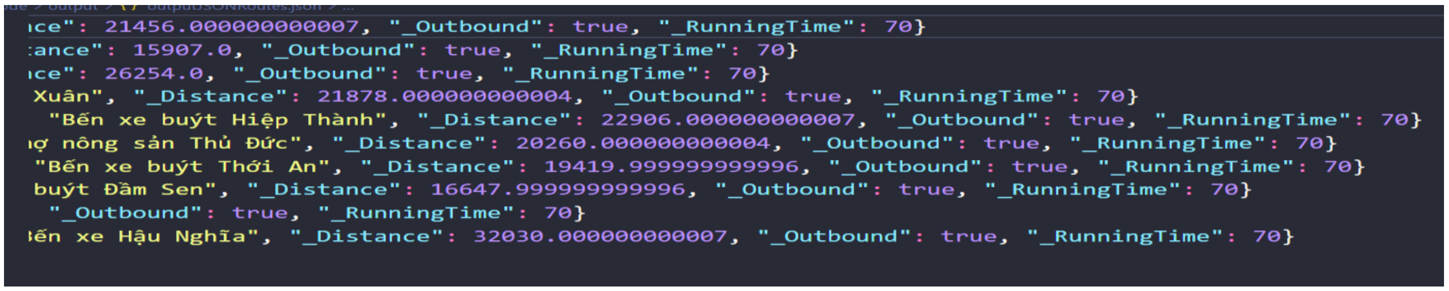
```
RouteId,RouteVarId,RouteVarName,RouteVarShortName,RouteNo,StartStop,EndStop,Distance,Outbound,RunningTi
3,5,Lượt đi: Bến Thành - Thạnh Lộc,Thạnh Lộc,03,Bến xe buýt Sài Gòn,THẠNH LỘC,21456.000000000007,True,7
7,13,Lượt đi: Bến xe Chợ Lớn - Gò vấp,Gò vấp,07,Bến xe Chợ Lớn,BÃI HẬU CẦN SỐ 1,15907.0,True,70
9,1,Lượt đi: Bến xe Chợ Lớn - Hưng Long,Hưng Long,09,Bến xe Chợ Lớn,Hưng Long,26254.0,True,70
25,1,Lượt đi: Bến xe Miền Tây - Phú Xuân,Khu Tái Định Cư Phú Mỹ,139,Bến xe Miền Tây,Phú Xuân,21878.0000
30,59,Lượt đi: Bến xe Chợ Lớn - Chợ Hiệp Thành,Chợ Hiệp Thành,145,Bến xe buýt Chợ Lớn - B,Bến xe buýt H
50,99,Lượt đi: Bến Phà Cát Lái - Chợ Nông Sản Thủ Đức,Chợ nông sản Thủ Đức,29,Cát Lái,Chợ nông sản Thủ
57,113,Lượt đi: Bến xe buýt Sài Gòn - Bến xe buýt Thới An,Thới An,36,Bến xe buýt Sài Gòn,Bến xe buýt Th
59,117,Lượt đi: Đầu bến Tân Quy - Bến xe buýt Đầm Sen,Đầm Sen,38,Đầu bến Tân Quy,Bến xe buýt Đầm Sen,16
86,1,Lượt đi,Thủ Dầu Một,61-3,Bến xe An Sương,Bến xe Thủ Dầu Một,31096.0,True,70
97,1,Lượt đi: Bến xe An Sương - Bến xe Hậu Nghĩa,Bến xe Hậu Nghĩa,62-5,Bến xe An Sương,Bến xe Hậu Nghĩa,

```

+ CSV file by pandas:



```
RouteId,RouteVarId,RouteVarName,RouteVarShortName,RouteNo,StartStop,EndStop,Distance,Outbound,RunningTi
3,5,Lượt đi: Bến Thành - Thạnh Lộc,Thạnh Lộc,03,Bến xe buýt Sài Gòn,THẠNH LỘC,21456.000000000007,True,7
7,13,Lượt đi: Bến xe Chợ Lớn - Gò vấp,Gò vấp,07,Bến xe Chợ Lớn,BÃI HẬU CẦN SỐ 1,15907.0,True,70
9,1,Lượt đi: Bến xe Chợ Lớn - Hưng Long,Hưng Long,09,Bến xe Chợ Lớn,Hưng Long,26254.0,True,70
25,1,Lượt đi: Bến xe Miền Tây - Phú Xuân,Khu Tái Định Cư Phú Mỹ,139,Bến xe Miền Tây,Phú Xuân,21878.0000
30,59,Lượt đi: Bến xe Chợ Lớn - Chợ Hiệp Thành,Chợ Hiệp Thành,145,Bến xe buýt Chợ Lớn - B,Bến xe buýt H
50,99,Lượt đi: Bến Phà Cát Lái - Chợ Nông Sản Thủ Đức,Chợ nông sản Thủ Đức,29,Cát Lái,Chợ nông sản Thủ
57,113,Lượt đi: Bến xe buýt Sài Gòn - Bến xe buýt Thới An,Thới An,36,Bến xe buýt Sài Gòn,Bến xe buýt Th
59,117,Lượt đi: Đầu bến Tân Quy - Bến xe buýt Đầm Sen,Đầm Sen,38,Đầu bến Tân Quy,Bến xe buýt Đầm Sen,16
86,1,Lượt đi,Thủ Dầu Một,61-3,Bến xe An Sương,Bến xe Thủ Dầu Một,31096.0,True,70
97,1,Lượt đi: Bến xe An Sương - Bến xe Hậu Nghĩa,Bến xe Hậu Nghĩa,62-5,Bến xe An Sương,Bến xe Hậu Nghĩa
```

+ JSON file:



```
ice": 21456.000000000007, "_Outbound": true, "_RunningTime": 70}
:ance": 15907.0, "_Outbound": true, "_RunningTime": 70}
ice": 26254.0, "_Outbound": true, "_RunningTime": 70}
 Xuân", "_Distance": 21878.000000000004, "_Outbound": true, "_RunningTime": 70}
  "Bến xe buýt Hiệp Thành", "_Distance": 22906.000000000007, "_Outbound": true, "_RunningTime": 70}
ıợ nông sản Thủ Đức", "_Distance": 20260.000000000004, "_Outbound": true, "_RunningTime": 70}
 "Bến xe buýt Thới An", "_Distance": 19419.999999999996, "_Outbound": true, "_RunningTime": 70}
 buýt Đầm Sen", "_Distance": 16647.999999999996, "_Outbound": true, "_RunningTime": 70}
  "_Outbound": true, "_RunningTime": 70}
ıến xe Hậu Nghĩa", "_Distance": 32030.000000000007, "_Outbound": true, "_RunningTime": 70}
```

V) Conclusion

In this technical report, I, Phan Tuan Kiet from Class 23APCS2, have successfully developed a program with a well-structured architecture and efficient functionality. Here are the key achievements and optimizations:

1) Structure and Components:

+ Organized the program into four main ".py" files: function.py, main.py, var.py, and drawio.png.

+ Utilized JSON files (stops.json and vars.json) to store and read data.

+ Established an output folder containing both CSV and JSON files to visualize results and resolve queries.

+ Created a Class Diagram (drawio.png) depicting essential classes like RouteVar, RouteVarQuery, Stop, and StopQuery.

+ Employed libraries such as JSON, CSV, and pandas for efficient reading and writing operations.

2) Comments and Initial Approach:

+ Thoroughly analyzed the structure and attributes of vars.json and stops.json.

+ Addressed differences between JSON boolean values and Python boolean types.

+ Flexibly handled special cases like null values (None in Python) in stops.json.

3) Data Handling and Class Design:

- Developed classes RouteVar and RouteVarQuery for managing route variables:

+ Utilized @property and get-set methods for attribute access.

+ Ensured encapsulation by making attributes protected.

- Implemented classes Stop and StopQuery for handling stop data and queries:

+ Applied similar encapsulation and access methods as in RouteVar classes.

+ Provided robust functions for loading data and querying stops.

4) User Interface and Main Handling:

+ Constructed a user-friendly interface in main.py.

+ Incorporated functions in function.py for inputting queries.

+ Allowed users to input multiple queries for detailed information retrieval.

5) Optimization:

+ Streamlined code structure and class design for readability and maintainability.

+ Utilized efficient libraries like pandas for CSV handling.

+ Ensured comprehensive error handling and informative outputs for better user experience.

+ Enhanced performance through optimized data querying and processing algorithms.

→ Overall, this project demonstrates a comprehensive approach to data handling and querying, coupled with a user-friendly interface. The optimized design and efficient functionality ensure smooth operation and reliable results. Further enhancements could include scalability for handling larger datasets and additional features for more advanced querying options.