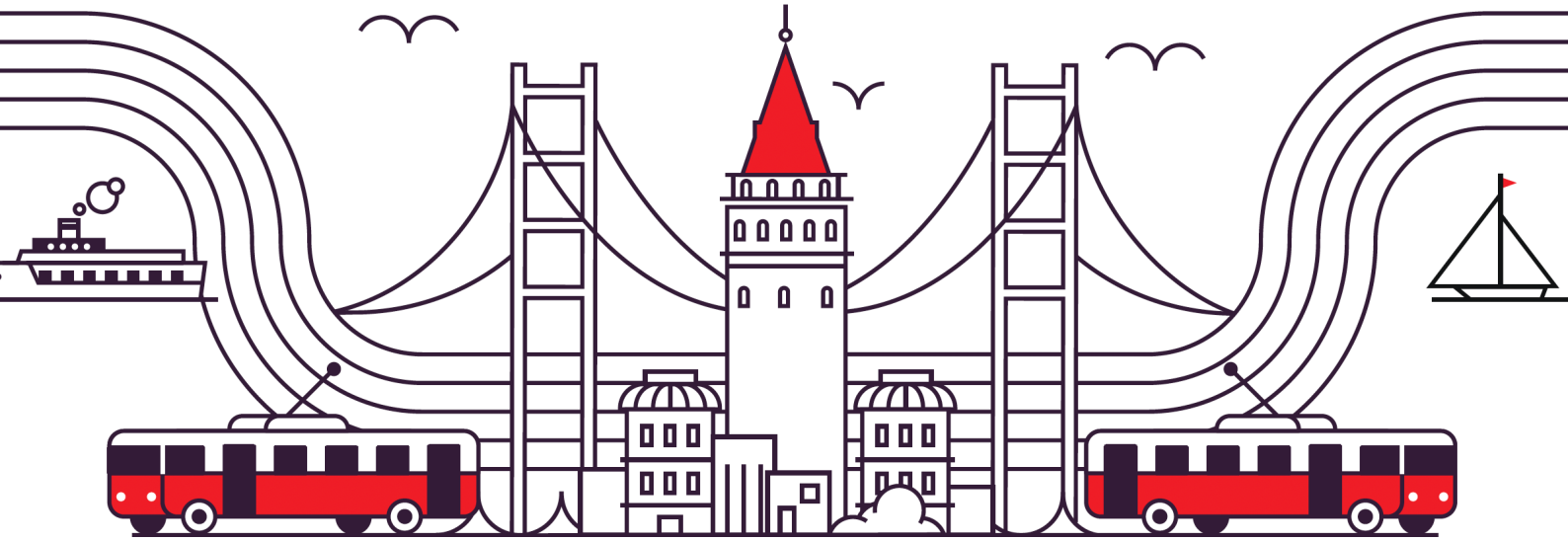
ACIBADEM UNIVERSITY

SPRING 2024

CSE 102 – PROGRAMMING PRACTICE

Assignment 1: Where is my bus?

(Assigned 15 April 2024 15:00, Due: 28 April 2024 23:59)



Objective: In this assignment, we will create a bus tracking and information application by using API and web services and processing data in XML and JSON data types. The API services used in this assignment are provided by IMM Open Data and IETT. To learn more about the Open Data platform, you can visit https://data.ibb.gov.tr/en/. All the information required for this assignment is available in this paper and in the documentation shared with you (en_iett-web-service-usage-document-v.1.4.pdf).

Before you start:
Just fill in the name, surname and student_id values in the solution.py extension file in the assignment folder you downloaded. All values you fill in are strings, do not change their type!

```
############ Do not change the assignment code value ############
assignment_code = 140110201
name = ""
surname = ""
student_id = ""
### Do not change the variable names above, just fill them in ###
```

Attention: After that please update the solution.py file to be your studentID_solution.py. For example, if your school number is 210402005, your file name should be 210402005_solution.py.

Glossary

line code: It is the unique code of the bus line. For example, the line code of "Yenidoğan / Sarıgazi – Kadıköy" line is 19S.

direction: The way of the bus. (arrival: "D"(dönüş)), (departure: "G"(gidiş))

IMM: Istanbul Metropolitan Municipality (İstanbul Büyükşehir Belediyesi - İBB)

Ready to start
Prototypes of all the functions you will write have been added to the solution.py file. Write only the functions without changing the function names. You are free to add other functions and call them within existing functions, but remember, only prototyped functions will be checked.

Attention: All functions you write should ***return*** values, not print them.

Attention: The web service you need to use for each function is stated in the description of the function. You can find the link of the relevant web service in the en_iett-web-service-usage-document-v.1.4.pdf file.

How to send a request to SOAP API ?

IETT uses SOAP API in this API service. There are different libraries to access SOAP APIs in Python. We will access the SOAP API using the "**zeep**" library.

First, we need to install the **zeep** library:

```
pip install zeep
```

Next, we need to know the WSDL (Web Services Description Language) file of the SOAP API we are targeting. This file contains information on how to use the API. In this assignment you must give the relevant url to the wsdl value.

Next, we create a client using the zeep.Client class. This client allows us to connect to the SOAP API based on the specified WSDL file.

We use the service feature on the client we created to use API methods. This includes all methods defined on the server side. For example, we can call a method like client.service.MethodName. Method names and if parameter names are available in the documentation.

Finally, we can get the return value of the called method and process it.

Example:

```
import zeep
import json

url = "https://api.ibb.gov.tr/iett/AracAnaVeri/AracOzellik.asmx?wsdl"

client = zeep.Client(wsdl = url)
fuel = client.service.GetAkarYakitToplamLitre_json(Yil = 2024, Ay = 4)
fuel = json.loads(fuel)

print(fuel)
```

`output:`

```
[{'ToplamAkarYakit': 405709.74, 'Gun': 1, 'Ay': 4, 'Yil': 2024}, {'ToplamAkarYakit': 418488.57, 'Gun': 2, 'Ay': 4, 'Yil': 2024}, {'ToplamAkar
Yakit': 414934.33, 'Gun': 3, 'Ay': 4, 'Yil': 2024}, {'ToplamAkarYakit': 414874.6, 'Gun': 4, 'Ay': 4, 'Yil': 2024}, {'ToplamAkarYakit': 38841
3.18, 'Gun': 5, 'Ay': 4, 'Yil': 2024}, {'ToplamAkarYakit': 153858.27, 'Gun': 6, 'Ay': 4, 'Yil': 2024}, {'ToplamAkarYakit': 108196.84, 'Gun':
7, 'Ay': 4, 'Yil': 2024}, {'ToplamAkarYakit': 161577.42, 'Gun': 8, 'Ay': 4, 'Yil': 2024}, {'ToplamAkarYakit': 157164.08, 'Gun': 9, 'Ay': 4, '
Yil': 2024}, {'ToplamAkarYakit': 170278.31, 'Gun': 10, 'Ay': 4, 'Yil': 2024}, {'ToplamAkarYakit': 167975.68, 'Gun': 11, 'Ay': 4, 'Yil': 202
4}, {'ToplamAkarYakit': 172864.42, 'Gun': 12, 'Ay': 4, 'Yil': 2024}, {'ToplamAkarYakit': 15682.27, 'Gun': 13, 'Ay': 4, 'Yil': 2024}]
```

Attention: You are free to use whatever you want in the APIs that return both data types: XML and JSON. Just make sure that the value you return is the same as the output type in the outputs.

1. Announcements

Write a function that takes the line code, calculates how many announcements there are on the line, and returns the announcements as a list. The "***MESAJ***" value in the response you receive from the relevant API is the announcements. You can use *TRANSPORTATION DYNAMIC DATA WEB SERVICES* for this function.

Clue: To return two values at the same time in a function, simply write them side by side with a comma between them. Like the example below.

```
return number_of_announcements, announcement_messages
```

Prototype:

```
def announcements(line_code):
```

Sample printouts:

input1:

```
print(announcements("11H"))
```

output1:

```
(2, ['TEPEÜSTÜ dan Saat 19:24 de hareket etmesi planlanan seferimiz  çesitli nedenlerle yapilamayacaktir.', 'TEPEÜSTÜ dan Saat 20:36 de harek
et etmesi planlanan seferimiz  çesitli nedenlerle yapilamayacaktir.'])
```

input2:

```
print(announcements("10"))
```

output2:

```
(0, [])
```

Note: Since this data is dynamic, your output results may change when you run it.

2. Stopping Buses

Write a function that returns the door numbers of all currently stopping buses as a list. You can use *FLEET MANAGEMENT DATA WEB SERVICES* for this function.

Prototype:

```
def stopping_buses():
```

Sample printouts:

input1:

```
print(stopping_buses())
```

output1:

```
['A-001', 'A-004', 'A-006', 'A-007', 'A-008', 'A-010', 'A-014', 'A-016', 'A-017', 'A-018', 'A-019', 'A-023', 'A-025', 'A-027', 'A-028', 'A-03
0', 'A-032', 'A-036', 'A-037', 'A-040', 'A-041', 'A-042', 'A-044', 'A-046', 'A-048', 'A-049', 'A-050', 'A-051', 'A-052', 'A-054', 'A-059', 'A
-060', 'A-062', 'A-063', 'A-064', 'A-065', 'A-066', 'A-068', 'A-070', 'A-071', 'A-072', 'A-073', 'A-074', 'A-076', 'A-077', 'A-078', 'A-082',
'A-083', 'A-084', 'A-086', 'A-087', 'A-088', 'A-089', 'A-091', 'A-092', 'A-093', 'A-094', 'A-095', 'A-096', 'A-097', 'A-098', 'A-100', 'A-100
1', 'A-1003', 'A-1005', 'A-1006', 'A-1007', 'A-1008', 'A-102', 'A-1021', 'A-1022', 'A-1026', 'A-1028', 'A-1029', 'A-103', 'A-1032', 'A-1044',
'A-1045', 'A-1047', 'A-1048', 'A-1049', 'A-105', 'A-1050', 'A-1051', 'A-1052', 'A-1053', 'A-1054', 'A-1055', 'A-1056', 'A-1057', 'A-1058', 'A
-1059', 'A-106', 'A-1061', 'A-1062', 'A-1063', 'A-1064', 'A-1065', 'A-1066', 'A-1067', 'A-1068', 'A-1069', 'A-107', 'A-1070', 'A-1078', 'A-10
8', 'A-1081', 'A-1082', 'A-1084', 'A-1085', 'A-110', 'A-1101', 'A-1105', 'A-1106', 'A-1107', 'A-1108', 'A-1113', 'A-112', 'A-1130', 'A-1131',
'A-114', 'A-1141', 'A-116', 'A-1160', 'A-1163', 'A-1164', 'A-1165', 'A-1167', 'A-1168', 'A-1180', 'A-1182', 'A-1183', 'A-1185', 'A-1186', 'A-
120', 'A-1200', 'A-1201', 'A-1202', 'A-1204', 'A-1205', 'A-121', 'A-1215', 'A-1216', 'A-1217', 'A-1218', 'A-122', 'A-1222', 'A-1225', 'A-122
```

3. The Fastest

Write a function that returns information about the top 3 buses with the highest speed at the moment. You can use *FLEET MANAGEMENT DATA WEB SERVICES* for this function.

Clue: You can use the sorted() method.

Prototype:

```
def max_speeds():
```

Sample printouts:

input1:

```
print(max_speeds())
```

output1:


```
[{'Operator': 'İETT', 'Garaj': 'IKITELLIISLETTIRMEGARAJI2', 'KapiNo': 'M2517', 'Saat': '09:00:26', 'Boylam': '28.7370158333333', 'Enlem': '4
1.0434316666666', 'Hiz': '127', 'Plaka': '34TN1784'}, {'Operator': 'Yeni İstanbul Özel Halk Otobüsleri Tic.A.Ş', 'Garaj': None, 'KapiNo': 'B-
190', 'Saat': '12:25:34', 'Boylam': '28.8143753333333', 'Enlem': '41.0050596666667', 'Hiz': '105', 'Plaka': '34 HO 2305'}, {'Operator': 'GÜNA
YDIN-ÇİMEN TUR', 'Garaj': None, 'KapiNo': 'E-025', 'Saat': '12:26:36', 'Boylam': '29.1586293333333', 'Enlem': '41.050757', 'Hiz': '103', 'Pla
ka': '34 CHK 755'}]
```


4. Stops of the Line

Write a function that takes the line code and direction parameters and returns the list of the stops of that line in that direction respectively. For the direction parameter in the function, take 'D' for arrival (dönüş) and 'G' for departure (gidiş). The direction information of the line is specified in the 'YON' parameter in the relevant data. You can use *Ibb Web Service* for this function.

Prototype:

```
def show_line_stops(line_code, direction):
```

Sample printouts:

input1:

```
print(show_line_stops("19T", "G"))
```

output1:


```
['ATATÜRK BULVARI', 'SEYİD NİZAM CADDESİ', 'AKDENİZ CADDESİ', 'NECMETTİN ERBAKAN PARKI', 'FERHATPAŞA MAHALLE MUHTARLIĞI', 'KILIÇ REİS', 'YEDİ
TEPE CADDESİ', '10. SOKAK', 'FERHATPAŞA İSMEK', 'YENİ ÇAMLICA', 'YENİ ÇAMLICA MERKEZ CAMİİ', 'DURSUNBEY CADDESİ', 'ACARLAR SİTESİ', 'MEVLANA
MESLEKİ VE TEKNİK ANADOLU LİSESİ', 'TÜRK İŞ 2.KISIM', 'TÜRK İŞ 1.KISIM', 'MEHMET USLU CAMİİ', 'NECİP FAZIL BULVARI', 'YENİŞEHİR PAZAR YERİ',
'CEMİLE BESLER İLKOKULU', 'UYDUKENT', 'CAHİT ZARİFOĞLU ANADOLU İMAM HATİP LİSESİ', 'GÜMRÜK BÖLGE MÜDÜRLÜĞÜ', 'KAYIŞDAĞI DUDULLU CADDESİ', 'İÇ
ERENKÖY METRO-KÜÇÜKBAKKALKÖY MERKEZ CAMİ', 'ATAŞEHİR ANADOLU LİSESİ', 'FEVZİPAŞA', 'MESKENLER', 'KÜÇÜKBAKKALKÖY', 'ACIBADEM ÜNİVERSİTESİ', 'K
ARAMAN ÇİFTLİK YOLU', 'ATAŞEHİR HAL', 'ÇAYIR CADDESİ ', 'KOZYATAĞI METRO', 'YÜZBEŞEVLER', 'YENİSAHRA', 'HASTANE ', 'GÖZTEPE KÖPRÜSÜ', 'İSTANB
UL MEDENİYET ÜNİVERSİTESİ', 'UZUNÇAYIR METROBÜS', 'ACIBADEM METRO  ', 'KOŞUYOLU KÖPRÜSÜ', 'İSFALT', 'SAĞLIK BİLİMLERİ ÜNİVERSİTESİ ECZACILIK
FAKÜLTESİ', 'HAYDARPAŞA NUMUNE HASTANESİ', 'HAYDARPAŞA MESLEKİ VE TEKNİK ANADOLU LİSESİ', 'KADIKÖY']
```


input2:

```
print(show_line_stops("122C", "D"))
```

output2:


```
['MECİDİYEKÖY VİYADÜK', 'ALİ SAMİ YEN', 'GAYRETTEPE', 'ESENTEPE', 'ZİNCİRLİKUYU-ÜNİVERSİTE', 'LEVENT', 'FABRİKALAR', 'KAVACIK KÖPRÜSÜ', 'BALK
AN CADDESİ', 'TEPEÜSTÜ']
```

5. Live Tracking

Write the function that saves the stops of the selected line and direction and the data of the buses currently operating on that line to the where.js file in the nested list type specified below. In this way, when you open the where_is_my_bus.html file in the assignment folder, you will be able to see the current locations of the stops and buses of the selected line. After saving these two lists to the where.js file, the function also should return the values of stops and buses respectively as the return value. You will need two different web services in this function. While using the *Ibb Web Service* for stops, you can use the *FLEET MANAGEMENT DATA WEB SERVICES* for instant bus locations.

Clue: To return two values at the same time in a function, simply write them side by side with a comma between them. Like the example below.

```
return stops, buses
```

For the stops of the line, the name of your list should be "**stops**". The order of the data in it should be as follows.

```
[['DURAKADI', 'YKOORDINATI', 'XKOORDINATI'], ['DURAKADI', 'YKOORDINATI', 'XKOORDINATI'], ...]
```

Note: The words are given according to the data returned from the relevant API.

If the selected line was 122C and its direction was departure (gidiş, "G"), the stop list should be as follows.

```
[['TEPEÜSTÜ', 41.022228, 29.129686], ['KAVACIK KÖPRÜSÜ', 41.087486, 29.093445], ['FABRİKALAR', 41.080142, 29.011466], ['LEVENT', 41.075935, 29.014347], ['ZİNCİRLİKUYU-ÜNİVERSİTE', 41.069947, 29.013611], ['ESENTEPE', 41.068372, 29.008536], ['ALİ SAMİ YEN', 41.066874, 29.001944], ['MECİDİYEKÖY VİYADÜK', 41.066853, 28.997007]]
```

The name of the select nested list for current bus locations should be "**buses**". The order of the data in it should be as follows.

```
[['kapino', 'enlem', 'boylam'], ['kapino', 'enlem', 'boylam'], ...]
```

Note: The words are given according to the data returned from the relevant API.

If the selected line is 122C and its direction is departure (gidiş, "G"), the list of stops may be as follows.

```
[['C-095', 41.0287905, 29.1301188333333], ['C-094', 41.0682383333333, 29.0081708333333]]
```

Attention: The door numbers, latitude and longitude information of the buses may differ from yours because these data are dynamic.

Note: Pay attention to the direction when getting data for buses. You can check the data specified with "route code" in the data received from the relevant API.

For example, in the data received from line 19S, the values may be like '19S_G_D0' or '19S_D_D0'. If the value between two underlines is "G", it means departure (gidiş), and "D" means arrival (dönüş).

Prototype:

```
def live_tracking(line_code, direction):
```

Sample printouts:

input1:

```
print(live_tracking("122C", "G"))
```

output1:

```
(([['TEPEÜSTÜ', 41.022228, 29.129686], ['KAVACIK KÖPRÜSÜ', 41.087486, 29.093445], ['FABRİKALAR', 41.080142, 29.011466], ['LEVENT', 41.075935, 29.014347], ['ZİNCİRLİKUYU-ÜNİVERSİTE', 41.069947, 29.013611], ['ESENTEPE', 41.068372, 29.008536], ['ALİ SAMİ YEN', 41.066874, 29.001944], ['MECİDİYEKÖY VİYADÜK', 41.066853, 28.997007]], [['C-095', 41.0287905, 29.1301188333333], ['C-094', 41.0682383333333, 29.0081708333333]])
```

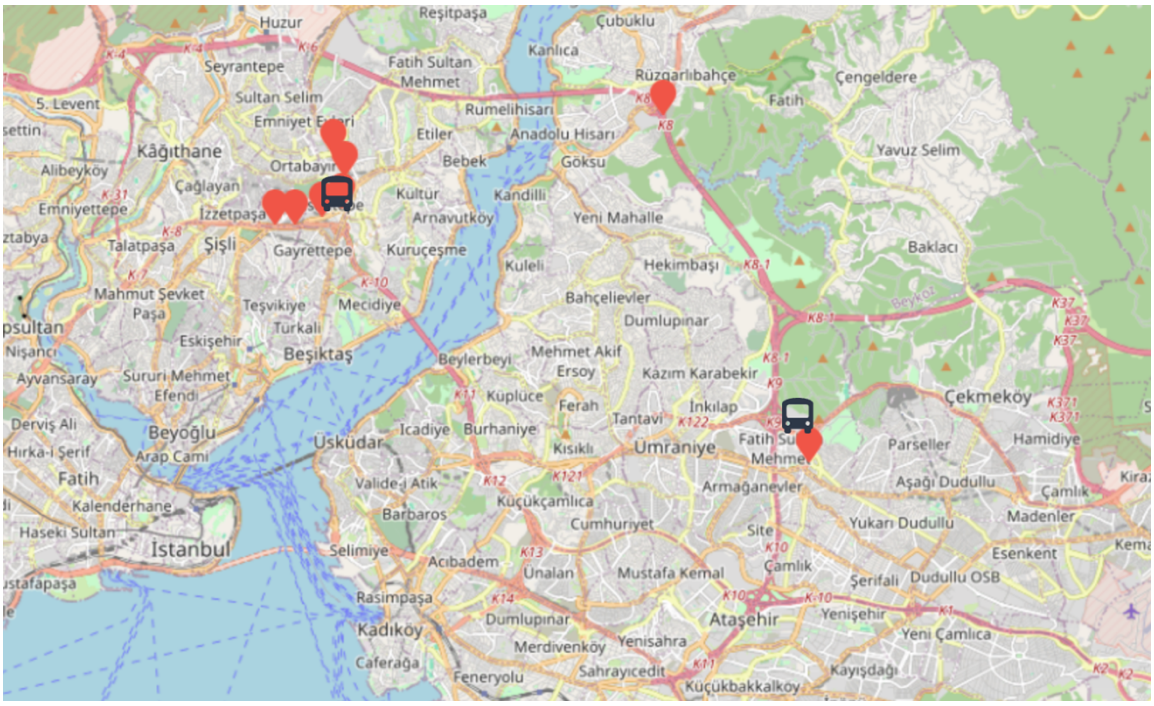
example where.js:

```
stops = [['TEPEÜSTÜ', 41.022228, 29.129686], ['KAVACIK KÖPRÜSÜ', 41.087486, 29.093445], ['FABRİKALAR', 41.080142, 29.011466], ['LEVENT', 41.075935, 29.014347], ['ZİNCİRLİKUYU-ÜNİVERSİTE', 41.069947, 29.013611], ['ESENTEPE', 41.068372, 29.008536], ['ALİ SAMİ YEN', 41.066874, 29.001944], ['MECİDİYEKÖY VİYADÜK', 41.066853, 28.997007]]

buses = [['C-095', 41.0277285, 29.1267245], ['C-094', 41.0693491666667, 29.0121541666667]]
```

Note: You do not need to make any changes to the where_is_my_bus.html file. After printing the where.js file correctly, when you open the html file, it will look like the following, depending on the line and direction you chose.

where_is_my_bus.html:



Before you send:

Create a new folder and name it your student ID. Then, put only the solution.py file, which contains only the answers, into this folder. Do not put any other files in the folder. Then format the folder as ZIP or RAR. For example, if your school number is 210402005, the name of your folder should be 210402005 and there should only be 210402005_solution.py file in it. The folder should be sent in ZIP or RAR format only.

Submission:

Upload the ZIP or RAR formatted folder within the time specified on the LMS (EYS).