Oyster Card System

This notebook is consist of a lean solution(fewer than 50 lines of code) including instruction to run the code lines for Oyster Card System to load and update balance of commuters after every journey either by tube or bus. This solution is simple, completely scalable and can be applied to large data sets with all possible stations(with in zone of 1,2 and 3) and all journey combinations for London's Oyster Card System. I also attached a notepad consisting these code block to run offline in a python environment/IDE like Jupyter or Spyder. ¶

- Instruction: This is a PDF and HTML of original Jupyter notebook. To run this codes refer the notepad(.py file) and copy paste it in a Jupyter notebook and after every code block enter shift+enter to execute the code block. You will get the same output after execution every code block.

Assumption: London tube rail, as a radial network system, I assume that any station in zone 3 from any station in zone 1 will be travelled only by crossing zone 2. So eventually the journey will come under the fare category of 'Any three zones'(3.2£) not under the category of 'Any two zones including zone 1'(3.00£).

First let's import all the necessary packages for this solution and read two tables-

1) Fare Chart(table to lookup fare for any journey)

2) Station-zone(table to lookup zone details for source and destination for any journey) - I created a dummy zone(0) for Chelsea to calculate bus fare for Earl's Court to Chelsea.

Instruction to run: copy paste it in a Jupyter notebook and enter shift+enter to execute the code block.

In [25]:
```
#Necessary packages

import os
import numpy as np
import pandas as pd


# Get fare and Station-zone data

os.chdir('D:/data_science')
fare_chart = pd.read_csv('alef_fare.csv')
staton_zone = pd.read_csv('station_zone.csv')
print(fare_chart)
staton_zone.head()
```

```
                          Journey  Fare
0                 Anywhere zone 1  2.50
1     Any one zone outside zone 1  2.00
2  Any two zones including zone 1  3.00
3  Any two zones excluding zone 1  2.25
4                 Any three zones  3.20
5                 Any bus journey  1.80
```

Out[25]:

	Station	Zone
0	Holborn	1
1	Earl's Court	1 2
2	Wimbledon	3
3	Hammersmith	2
4	Chelsea	0

.

Function block to load Oyster Card with any amount

In [26]:
```
def load_card(amount):
    initial_balance = amount
    return initial_balance
```

Function block to get zones for source station and destination station from Station-zone look up table. Also System gets information about mode of travel(e.g. Tube legitimate jouney or Bus or through the inward barrier at the station)

Instruction to run: copy paste it in a Jupyter notebook and enter shift+enter to execute the code block.

In [27]:
```
#Function to get source_zone and destination zone

source_zone = []
dest_zone = []
def get_zone(source_station,dest_station,mode):
    return staton_zone.loc[staton_zone.Station == source_station, 'Zone'],stat
on_zone.loc[staton_zone.Station == dest_station, 'Zone'],mode
```

In [28]:
```
#testing funtion

source_zone,dest_zone,mode =  get_zone(source_station="Holborn",dest_station =
 "Earl's Court",mode='tube_legitimate')
print('source zone is:', source_zone)
print('destination zone is:', dest_zone)
print('transportation mode is:', mode)
```

```
source zone is: 0    1
Name: Zone, dtype: object
destination zone is: 1    1 2
Name: Zone, dtype: object
transportation mode is: tube_legitimate
```

.

Function block to get fare from Fare look up table after inputting source station zone, destination station zone and mode of travel. System favour the customer in case of more than one possible fare(in case a station comes under two zones, e.g. Holborn to Earl's Court fare would be 2.5 Pounds not 3 Pounds)

Instruction to run: copy paste it in a Jupyter notebook and enter shift+enter to execute the code block.

In [29]:
```
#Fare calculation function

fare = []

def fare_calculation(source_zone,dest_zone):
    if mode == 'tube_legitimate':
        if '1' in str(list(source_zone)) and '1' in str(list(dest_zone)):
            fare = 2.50
        elif '1' in str(list(source_zone)) and '3' in str(list(dest_zone)):

            fare = 3.20
        elif ('2' in str(list(source_zone)) and '2' in str(list(dest_zone))) o
r ('3' in str(list(source_zone)) and '3' in str(list(dest_zone))):
            fare = 2.00
        elif ('2' in str(list(source_zone)) and '3' in str(list(dest_zone))) o
r ('3' in str(list(source_zone)) and '2' in str(list(dest_zone))):
            fare = 2.25
        elif ('1' in str(list(source_zone)) and '2' in str(list(dest_zone))) o
r ('2' in str(list(source_zone)) and '1' in str(list(dest_zone))):
            fare = 3.00
        else:
            fare = 3.20

    elif mode == 'bus':
        fare = 1.80

#Will take care of passes through the inward barrier at the station
    else:
        fare = 3.20
    return fare
```

In [30]:
```
#testing of fare calculation function

print('fare for Holborn to Earl Court by tube is:' + str(fare_calculation(sour
ce_zone, dest_zone)) + ' Pounds')
```

```
fare for Holborn to Earl Court by tube is:2.5 Pounds
```

.

Testing in a Real Scenario:

Testing the system for demonstrate loading of card for a user with 30 £ and update balance after following three journey:

- Tube Holborn to Earl's Court

- 328 bus from Earl's Court to Chelsea

- Tube Earl's court to Hammersmith

Instruction to run: copy paste it in a Jupyter notebook and enter shift+enter to execute the code

In [32]:
```
#Calculation fare and updating balance

#Loading card with 30 Pounds
initial_bal = load_card(30)
print('card loaded with:' + str(initial_bal) + ' Pounds')

#Update after Tube Holborn to Earl’s Court
source_zone,dest_zone,mode =  get_zone(source_station="Holborn",dest_station =
 "Earl's Court",mode='tube_legitimate')
balance = initial_bal - fare_calculation(source_zone=source_zone, dest_zone=de
st_zone)
print('fare for Holborn to Earl Court by tube:' + str(fare_calculation(source_
zone, dest_zone)) + ' Pounds')
print('balance after this trip:' + str(balance) + ' Pounds')

#328 bus from Earl’s Court to Chelsea- (To run the code correctly I put a dumm
y zone for Chelsea)
source_zone,dest_zone,mode =  get_zone(source_station="Earl's Court",dest_stat
ion = "Chelsea",mode='bus')
balance = balance - fare_calculation(source_zone=source_zone, dest_zone=dest_z
one)
print('fare for Earl Court to Chelsea by bus:' + str(fare_calculation(source_z
one, dest_zone)) + ' Pounds')
print('balance after this trip:' + str(balance) + ' Pounds')

#Tube Earl’s court to Hammersmith
source_zone,dest_zone,mode =  get_zone(source_station="Earl's Court",dest_stat
ion = "Hammersmith",mode='tube_legitimate')
balance = balance - fare_calculation(source_zone=source_zone, dest_zone=dest_z
one)
print('fare for Earl Court to Hammersmith by bus:' + str(fare_calculation(sour
ce_zone, dest_zone)) + ' Pounds')
print('balance after this trip:' + str(balance) + ' Pounds')
```

```
card loaded with:30 Pounds
fare for Holborn to Earl Court by tube:2.5 Pounds
balance after this trip:27.5 Pounds
fare for Earl Court to Chelsea by bus:1.8 Pounds
balance after this trip:25.7 Pounds
fare for Earl Court to Hammersmith by bus:2.0 Pounds
balance after this trip:23.7 Pounds
```

Thank You