1.Να προτείνετε αλγόριθμο για την οπτική ανάγνωση της πινακίδας από φωτογραφία

Πρέπει να χρησιμοποιήσουμε έναν αλγόριθμο OCR(Optical Character Recognition)

Η τεχνογνωσία του Optical Character Recognition (OCR) αντιλαμβάνεται το έντυπο και το χειροκίνητο κείμενο σε έγγραφα με εικόνα.

Ένας αλγόριθμος OCR χρησιμοποιεί τα μοντέλα AI που παρέχονται από το Computer Vision.

Η αναγνώριση οπτικών χαρακτήρων (OCR) επιτρέπει να αναγνωριστεί έντυπο ή γραμμένο με το χέρι κείμενο από εικόνες, για παράδειγμα, φωτογραφίες οδικών πινακίδων και αντικειμένων, καθώς και από αρχεία — προσκλήσεις, λογαριασμούς, νομισματικές εκθέσεις, άρθρα και άλλα.

Αναφέρω σαν παράδειγμα τις Azure Cognitive Services της Microsoft οι οποίες δίνουν έτοιμα Computer Vision APIs για να τα καλέσεις στον κώδικά σου για αυτό το σκοπό..

Για παράδειγμα το Computer Vision API v3.0 το οποίο ανήκει στις Γνωστικές Υπηρεσίες της Azure.

Για Αγγλικά, Ισπανικά, Γερμανικά, Γαλλικά, Ιταλικά, Πορτογαλικά και Ολλανδικά, χρησιμοποιείται το νέο API "Read" της Azure.

Για τυχόν υπόλοιπες διαλέκτους, χρησιμοποιείται το OCR API.

Η τεχνολογία OCR διαχωρίζει κείμενο από έγγραφα εικόνας. Τα σχέδια των διατηρημένων αρχείων περιλαμβάνουν: .JPEG .JPG .PNG .BMP .GIF .TIFF

Προτείνω το API “Read” της Azure για τη δουλειά που θέλουμε με την αναγνώριση πινακίδων οχημάτων το οποίο είναι τόσο διαθέσιμο ως Cloud υπηρεσία όσο και ως υπηρεσία on premises.

Είναι βελτιωμένο για να εξάγει κείμενο από εικόνες με μεγάλο μέγεθος κειμένου και πολυσέλιδες εγγραφές PDF με ανάμεικτες διαλέκτους. Υποστηρίζει την εξαγωγή τόσο έντυπου όσο και μη γραμμένου κειμένου σε παρόμοια εικόνα ή εγγραφή.

Links: What is Optical character recognition? - Azure Cognitive Services | Microsoft Docs

OCR cognitive skill - Azure Cognitive Search | Microsoft Docs

Το καλό είναι ότι μπορούμε να χρησιμοποιήσουμε επίσης pretrained μοντέλα της Microsoft από ένα μεγάλο dataset εικόνων κάτι που για να το κάναμε εμείς και να πετυχαίναμε τέτοιο accuracy θα έπρεπε να έχουμε υπολογιστικούς πόρους αντίστοιχους της Microsoft για αυτό το σκοπό κάτι ασύμφορο.

Έταιρη λύση που θα μπορούσα να προτείνω για μεγαλύτερη ευελιξία και ανεξαρτησία από APIs τρίτων παρόχων όπως είναι η Microsoft είναι βιβλιοθήκες που υπάρχουν για OCI στο framework-βιβλιοθήκη OpenCV της Python.

Παραθέτω αντίστοιχο link εδώ:

OpenCV: Automatic License/Number Plate Recognition (ANPR) with Python - PyImageSearch

Εδώ μέσα υπάρχει και ο κώδικας για την υλοποίηση.

2.Να προτείνετε αλγόριθμο για την αναγνώριση του τύπου οχήματος.

Για την αναγνώριση του τύπου οχήματος πρέπει να χρησιμοποιήσουμε συνελικτικά νευρωνικά δίκτυα.

(Convolutional Neural Networks)

Για αυτό το σκοπό ουσιαστικά χρειαζόμαστε κάποιον αλγόριθμο image classification.

Για να γίνει αυτό χρειαζόμαστε τύπους οχημάτων από ένα σύνολο εικόνων που περιέχει διαφορετικούς τύπους οχημάτων.

Το image classification περιλαμβάνει την εκπαίδευση ενός μοντέλου τεχνητής νοημοσύνης για τον χαρακτηρισμό των εικόνων ενόψει των αντικειμένων τους. Για παράδειγμα, σε μια ρύθμιση ελέγχου του αδιεξόδου στις ώρες αιχμής, θα μπορούσαμε να χρησιμοποιήσουμε ένα μοντέλο image classification για να ταξινομήσουμε φωτογραφίες υπό το φως του είδους του οχήματος που περιέχουν, όπως καμπίνες, μεταφορικά, ποδηλάτες κ.λπ.

Στην περίπτωσή μας με την αναγνώριση είδος οχήματος σε parking φυσικά μπορεί να χρησιμοποιηθεί.

Και εδώ θα μπορούσαμε να χρησιμοποιήσουμε επίσης pretrained μοντέλα της Microsoft από ένα μεγάλο dataset εικόνων κάτι που για να το κάναμε εμείς και να πετυχαίναμε τέτοιο accuracy θα έπρεπε να έχουμε υπολογιστικούς πόρους αντίστοιχους της Microsoft για αυτό το σκοπό κάτι ασύμφορο.

Επιπρόσθετα το image classification ίσως χρειαστεί ακόμα μεγαλύτερη υπολογιστική ισχύ.

Και εδώ προτείνω τη λύση της Microsoft για Image Classification.

Understand computer vision - Learn | Microsoft Docs

Εδώ μπορούμε να δούμε ένα παράδειγμα image classification με pretrained μοντέλα.

AI Demos (microsoft.com)

Εδώ δε μπορούμε να ακολουθήσουμε τα βήματα που προτείνει η Microsoft Azure για auto-train computer vision models με την Python.(Microsoft Azure Machine Learning Studio)

Set up AutoML for computer vision - Azure Machine Learning | Microsoft Docs

Αντίστοιχα μπορούμε να χρησιμοποιήσουμε την πλατφόρμα ή να κατεβάσουμε το API και να το καλέσουμε στον κώδικά μας Python με το API key του και έναν κωδικό.

‘Άλλη λύση αν θέλουμε έτοιμο κώδικα όχι από πάροχο χρησιμοποιώντας λογισμικά όπως το Keras και το Tensorflow για την εκπαίδευση τέτοιων μοντέλων μπορούμε να βρούμε σε αυτό το link:

PyImageSearch University: You can master Computer Vision, Deep Learning, and OpenCV

Εδώ υπάρχει και αντίστοιχος κώδικας για την υλοποίηση.

3.Με βάση τον παρακάτω τιμοκατάλογο, να κατασκευάσετε την εφαρμογή που θα υπολογίζει τέλη διέλευσης και στάθμευσης στην πόλη. Η εφαρμογή θα τρέχει σε ημερήσια βάση και θα παράγει μια αναφορά με τα αυτοκίνητα που είχαν είσοδο και έξοδο, το αριθμό της πινακίδας και το τέλος που θα πρέπει να πληρώσουν.

Διάρκεια	Τιμή για Επιβατηγό	Τιμή για Φορτηγό	Τιμή για Λεωφορείο
0-30	0	0	0
30-60	3	5	7
60-120	5	9	16
120- 240	7	11	19
240 +	Για κάθε ώρα 1 ευρώ	Για κάθε ώρα 2 ευρώ	Για κάθε ώρα 3 ευρώ

Εδώ παραθέτω τον κώδικα Python που έγραψα και θα τον επισυνάψω και ξεχωριστά όπως και ένα αρχείο Excel με τους πίνακες που τελικά κατέληξα να χρησιμοποιήσω για τη λύση μου.

Όπως θα δείτε είναι απλός και απέριττος με απλές εντολές και λίγες βιβλιοθήκες.

Ήθελα να χρησιμοποιήσω pandas ή Numpy αλλά τελικά θεώρησα ότι αυτά είναι πιο χρήσιμα όταν έχεις ένα έτοιμο dataset και θες να το σπάσεις σε κομμάτια και θες να εξάγεις από εκεί πίνακες.

Συνεπώς χρησιμοποίησα απλά δυσδιάστατους πίνακες που είναι ουσιαστικά λίστες με δυναμική δέσμευση μνήμης.

Έχω έναν ατέρμων βρόχο με while που τελειώνει όταν το επιλέξει ο χρήστης με την αντίστοιχη επιλογή.

Επίσης ένα while μέσα που ελέγχει μόνο την ημέρα και αν αλλάξει βγαίνει.

Είναι μέσα όσο η μέρα δεν αλλάζει..

Χρησιμοποίησα την συνάρτηση datetime από την οποία πήρα μόνο τη μέρα για τη σύγκριση στο βρόχο.

Επίσης όπως θα δείτε εξάγω στο τέλος με print την πινακίδα το κόστος και τη διάρκεια παραμονής σε λεπτά στο parking και για τους 3 τύπους οχημάτων(επιβατικό, φορτηγό ή λεωφορείο) μόνο όμως αυτά που μπαίνουν και βγαίνουν την ίδια μέρα γιατί εγώ αυτό κατάλαβα από την εκφώνηση ότι ζητείται.

Δηλαδή αν και ο αλγόριθμος όπως θα δείτε δουλεύει ακόμα και αν μπουν και βγουν οσεσδήποτε φορές οχήματα από το parking.(εκτυπώνονται αυτά που ζητούνται για το κάθε όχημα ξεχωριστά όπως και ο τύπος του) και επίσης αφήνω την επιλογή μία μέρα να μπει μόνο ή να βγει μόνο ένα όχημα εγώ εκτυπώνω μόνο αυτά που ζητούνται για οχήματα που μπαίνουν και βγαίνουν την ίδια μέρα.

Επίσης μιας και αυτά με ενδιαφέρουν μόνο σε κάθε λούπα του μέσα while όπου είναι για κάθε ημέρα ξεχωριστά αδειάζονται όλοι οι πίνακες που χρησιμοποιώ μιας και δεν με ενδιαφέρει αν είχαν μπει προηγούμενες μέρες οχήματα και βγήκαν σήμερα ούτε αν μπήκαν σήμερα οχήματα και βγουν τις επόμενες μέρες.

Οπότε καθημερινά γεμίζουν οι πίνακες με τα αμάξια που μπήκαν, με τα αμάξια που βγήκαν και με τα αμάξια που μπήκαν και βγήκαν και εγώ εξάγω πινακίδα, κόστος και διάρκεια παραμονής ανά επιβατικό με τη σειρά, ανά φορτηγό με τη σειρά και ανά λεωφορείο με τη σειρά **μόνο** για αυτά τα οχήματα που μπήκαν και βγήκαν την ίδια μέρα.

Για το κόστος σαφώς χρησιμοποιήθηκε ο πίνακας που μας προτείνεται με τα κόστη.

Επίσης έχω στρογγυλοποιήσει τα δευτερόλεπτα στη διαμονή και υπολογίζω τον χρόνο διαμονής σε λεπτά.

Μιας και λεπτά έχει και ο πίνακας.

Βεβαίως αυτό λειτουργεί ανάλογα με τη διάρκεια που παίρνει από την βιβλιοθήκη datetime που μετατρέπω σε string οπότε θέλει κάποια ώρα για να μεγαλώσει το κόστος πάνω από 0.

Τέλος για το μέσα while-λούπα όπως είπα βλέπω μόνο τη μέρα από την datetime ενώ μέσα στην μέσα while-λούπα παίρνω μόνο την ώρα και βλέπω έτσι ποιο αμάξι μπήκε πρώτο ποιο δεύτερο κλπ και ποιο βγήκε αντίστοιχα πρώτο ποιο δεύτερο κλπ μιας και θα πρόκειται για την ίδια μέρα, μήνα, έτος οπότε αρκεί να συγκρίνω μόνο τις ώρες, τα λεπτά και τα δευτερόλεπτα για να υπολογίσω ώρα εξόδου - ώρα εισόδου ώστε να υπολογίσω και το κόστος.

Τέλος να πω ότι σε κανονικές συνθήκες θα τραβούσα από κάποιο API για κάθε αμάξι τις πινακίδες και το είδος με βάση αυτά που είπα στα ερωτήματα 1 και 2 ή θα χρησιμοποιούσα κάποια βιβλιοθήκη σαν αυτές που ανέφερα παραπάνω με κάποια scripts τα οποία θα τα καλούσα για κάθε αμάξι ώστε να έχω πινακίδα-αριθμός κυκλοφορίας, είδος αμαξιού-τύπος οχήματος, ώρα εισόδου και ώρα εξόδου.

Στο παράδειγμα που υλοποίησα όμως θεωρώ ότι τα εισάγει ο χρήστης με input που του ζητείται μιας και δεν κάλεσα κάποιο API πχ ώστε να πάρω αυτά τα δεδομένα για κάθε αμάξι και να τα χρησιμοποιήσω στον αλγόριθμό μου.

Έχω 7 πίνακες με 2 στήλες και l γραμμές για το κάθε είδος οχήματος επιβατικό, φορτηγό ή λεωφορείο.

Οπότε είναι 7*3=21 πίνακες συνολικά.

Car_plate_entry:στην πρώτη στήλη είναι ένας δείκτης με τη σειρά εισόδου του οχήματος, κοινός σε όλους τους πίνακες εισόδων ανά τύπο αμαξιού και στην δεύτερη στήλη η πινακίδα του

Car_date_entry: στην πρώτη στήλη είναι ένας δείκτης με τη σειρά εισόδου του οχήματος, κοινός σε όλους τους πίνακες εισόδων ανά τύπο αμαξιού και στην δεύτερη στήλη η ημερομηνία εισόδου του που τελικά όπως είπα κράτησα την ώρα εισόδου του αφού η λούπα μέσα while είναι ανά ημέρα για λόγους απλότητας

Car_plate_exit: στην πρώτη στήλη είναι ένας δείκτης με τη σειρά εξόδου του οχήματος, κοινός σε όλους τους πίνακες εξόδων ανά τύπο αμαξιού και στην δεύτερη στήλη η πινακίδα του

Cat_date_exit: στην πρώτη στήλη είναι ένας δείκτης με τη σειρά εξόδου του οχήματος, κοινός σε όλους τους πίνακες εξόδων ανά τύπο αμαξιού και στην δεύτερη στήλη η η ημερομηνία εξόδου του που τελικά όπως είπα κράτησα την ώρα εξόδου του αφού η λούπα μέσα while είναι ανά ημέρα για λόγους απλότητας

Car_duration: στην πρώτη στήλη είναι ένας δείκτης με τη σειρά εξόδου του οχήματος, κοινός σε όλους τους πίνακες εξόδων ανά τύπο αμαξιού και στην δεύτερη στήλη η διάρκεια παραμονής του κάθε οχήματος βγήκε από το parking μέσα στη μέρα σε λεπτά

Car_cost: στην πρώτη στήλη είναι ένας δείκτης με τη σειρά εξόδου του οχήματος, κοινός σε όλους τους πίνακες εξόδων ανά τύπο αμαξιού και στην δεύτερη στήλη το κόστος παραμονής του κάθε οχήματος στο parking σε ευρώ

Car_plate_entry_exit: στην πρώτη στήλη είναι ένας δείκτης με τη σειρά εξόδου του οχήματος, κοινός σε όλους τους πίνακες εξόδων ανά τύπο αμαξιού και στην δεύτερη στήλη η πινακίδα του κάθε οχήματος που μπήκε και βγήκε την ίδια ημέρα στο parking που αυτά τα οχήματα τα στοιχεία τους ψάχνουμε κιόλας

Τρέχω μια λούπα σε κάθε ημέρα και βλέπω αν η πινακίδα εξόδου είναι ίση με την πινακίδα εξόδου και η ημερομηνία εξόδου είναι μεγαλύτερη από την ημερομηνία εισόδου(σε περίπτωση που ένα αμάξι μπήκε και βγήκε πάνω από 1 φορές) τότε καταχωρώ σε έναν πίνακα τα αμάξια που μπήκαν και βγήκαν, τη διάρκεια παραμονής αφαιρώντας ώρα εξόδου-ώρα εισόδου για κάθε όχημα ξεχωριστά και το κόστος.

Τέλος αυτό το κάνω 3 φορές για τα 3 είδη οχήματος.

Παρατίθεται ο κώδικας και εδώ:

```
#1 Importing the libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math
from datetime import datetime

# getting the timestamp
#ts = datetime.timestamp(dt)
```

```
#print("Date and time is:", dt)
#print("Timestamp is:", ts)

o=1
while o==1:
    # Getting the current date and time
    dt = datetime.now()
    #str_date_time = dt.strftime("%d-%m-%Y, %H:%M:%S")
    str_day = dt.strftime("%d")
    #day = int(str_date_time[0:2])
    day=int(str_day)
    index_car_entry = 0
    index_car_exit = 0
    index_truck_entry = 0
    index_truck_exit = 0
    index_bus_entry = 0
    index_bus_exit = 0
    car_plate_entry = []
    car_date_entry=[]
    car_plate_exit=[]
    car_date_exit=[]
    car_duration=[]
    car_cost=[]
    car_plate_entry_exit=[]
    truck_plate_entry = []
    truck_date_entry = []
    truck_plate_exit = []
    truck_date_exit = []
    truck_duration = []
    truck_cost = []
    truck_plate_entry_exit=[]
    bus_plate_entry = []
    bus_date_entry = []
    bus_plate_exit = []
    bus_date_exit = []
    bus_duration = []
    bus_cost = []
    bus_plate_entry_exit=[]
    flag_car_entry_exit=0
    flag_bus_entry_exit=0
    flag_truck_entry_exit=0
    dt8 = datetime.now()
    str_day2 = dt8.strftime("%d")
    day2 = int(str_day2)
    while day2==day:
        flag_car_entry=input("Enter 1 if a car enters else enter 0:")
        if flag_car_entry=='1':
            index_car_entry+=1
            car_plate_entr=input("Enter the car plate of the car it enters:")
            col = []
            for i in range(0,2):
                if i==0:
                  col.append(index_car_entry)
                else:
                  col.append(car_plate_entr)
            car_plate_entry.append(col)
```

```
            col1=[]
            for i in range(0,2):
                if i == 0:
                    col1.append(index_car_entry)
                else:
                    dt1 = datetime.now()
                    entry_time = dt1.strftime("%H:%M:%S")
                    col1.append(entry_time)
            car_date_entry.append(col1)
        flag_car_exit = input("Enter 1 if a car exits else enter 0:")
        if flag_car_exit == '1':
            index_car_exit += 1
            car_plate_exi = input("Enter the car plate of the car it exits:")
            col2=[]
            for i in range(0,2):
                if i==0:
                  col2.append(index_car_exit)
                else:
                  col2.append(car_plate_exi)
            car_plate_exit.append(col2)
            col3=[]
            for i in range(0,2):
                if i == 0:
                    col3.append(index_car_exit)
                else:
                    dt2 = datetime.now()
                    exit_time = dt2.strftime("%H:%M:%S")
                    col3.append(exit_time)
            car_date_exit.append(col3)
        flag_truck_entry = input("Enter 1 if a truck enters else enter 0:")
        if flag_truck_entry == '1':
            index_truck_entry += 1
            truck_plate_entr = input("Enter the truck plate of the truck it
enters:")
            col4 = []
            for i in range(0, 2):
                if i == 0:
                    col4.append(index_truck_entry)
                else:
                    col4.append(truck_plate_entr)
            truck_plate_entry.append(col4)
            col5 = []
            for i in range(0, 2):
                if i == 0:
                    col5.append(index_truck_entry)
                else:
                    dt3 = datetime.now()
                    entry_time1 = dt3.strftime("%H:%M:%S")
                    col5.append(entry_time1)
            truck_date_entry.append(col5)
        flag_truck_exit = input("Enter 1 if a truck exits else enter 0:")
        if flag_truck_exit == '1':
            index_truck_exit += 1
            truck_plate_exi = input("Enter the truck plate of the truck it
exits:")
            col6 = []
            for i in range(0, 2):
```

```
                if i == 0:
                    col6.append(index_truck_exit)
                else:
                    col6.append(truck_plate_exi)
            truck_plate_exit.append(col6)
            col7 = []
            for i in range(0, 2):
                if i == 0:
                    col7.append(index_truck_exit)
                else:
                    dt4 = datetime.now()
                    exit_time1 = dt4.strftime("%H:%M:%S")
                    col7.append(exit_time1)
            truck_date_exit.append(col7)
        flag_bus_entry = input("Enter 1 if a bus enters else enter 0:")
        if flag_bus_entry == '1':
            index_bus_entry += 1
            bus_plate_entr = input("Enter the bus plate of the bus it
enters:")
            col8 = []
            for i in range(0, 2):
                if i == 0:
                    col8.append(index_bus_entry)
                else:
                    col8.append(bus_plate_entr)
            bus_plate_entry.append(col8)
            col9 = []
            for i in range(0, 2):
                if i == 0:
                    col9.append(index_bus_entry)
                else:
                    dt5 = datetime.now()
                    entry_time2 = dt5.strftime("%H:%M:%S")
                    col9.append(entry_time2)
            bus_date_entry.append(col9)
        flag_bus_exit = input("Enter 1 if a bus exits else enter 0:")
        if flag_bus_exit == '1':
            index_bus_exit += 1
            bus_plate_exi = input("Enter the bus plate of the bus it exits:")
            col10 = []
            for i in range(0, 2):
                if i == 0:
                    col10.append(index_bus_exit)
                else:
                    col10.append(bus_plate_exi)
            bus_plate_exit.append(col10)
            col11 = []
            for i in range(0, 2):
                if i == 0:
                    col11.append(index_bus_exit)
                else:
                    dt6 = datetime.now()
                    exit_time2 = dt6.strftime("%H:%M:%S")
                    col11.append(exit_time2)
            bus_date_exit.append(col11)

        car_plate_entry_exit = []
```

```
        car_duration = []
        car_cost = []
        for i in range(0, index_car_exit):
            for j in range(0, index_car_entry):
                if car_plate_exit[i][1]==car_plate_entry[j][1] and
car_date_exit[i][1]>car_date_entry[j][1]:
                    flag_car_entry_exit=1
                    col12=[]
                    col13=[]
                    col14=[]
                    for k in range(0, 2):
                        if k == 0:
                            col12.append(index_car_exit)
                            col13.append(index_car_exit)
                            col14.append(index_car_exit)
                        else:
                            duration_car_min=((int(car_date_exit[i][1][0:2])-
int(car_date_entry[j][1][0:2]))*60)+(int(car_date_exit[i][1][3:5])-
int(car_date_entry[j][1][3:5]))+((int(car_date_exit[i][1][6:8])-
int(car_date_entry[j][1][6:8]))/60)
                            col12.append(round(duration_car_min))
                            if round(duration_car_min)>=0 and
round(duration_car_min)<=30:
                                car_cost_euros=0
                            elif round(duration_car_min)>30 and
round(duration_car_min)<=60:
                                car_cost_euros=3
                            elif round(duration_car_min)>60 and
round(duration_car_min)<=120:
                                car_cost_euros=5
                            elif round(duration_car_min)>120 and
round(duration_car_min)<=240:
                                car_cost_euros=7
                            else:
                                car_cost_euros=(round(duration_car_min)/60)*1
                            col13.append(car_cost_euros)
                            col14.append(car_plate_exit[i][1])
                    car_duration.append(col12)
                    car_cost.append(col13)
                    car_plate_entry_exit.append(col14)

        truck_plate_entry_exit = []
        truck_duration = []
        truck_cost = []
        for i in range(0, index_truck_exit):
            for j in range(0, index_truck_entry):
                if truck_plate_exit[i][1]==truck_plate_entry[j][1] and
truck_date_exit[i][1]>truck_date_entry[j][1]:
                    flag_truck_entry_exit=1
                    col15=[]
                    col16=[]
                    col17=[]
                    for k in range(0, 2):
                        if k == 0:
                            col15.append(index_truck_exit)
                            col16.append(index_truck_exit)
                            col17.append(index_truck_exit)
```

```
                        else:

duration_truck_min=((int(truck_date_exit[i][1][0:2])-
int(truck_date_entry[j][1][0:2]))*60)+(int(truck_date_exit[i][1][3:5])-
int(truck_date_entry[j][1][3:5]))+((int(truck_date_exit[i][1][6:8])-
int(truck_date_entry[j][1][6:8]))/60)
                            col15.append(round(duration_truck_min))
                            if round(duration_truck_min)>=0 and
round(duration_truck_min)<=30:
                                truck_cost_euros=0
                            elif round(duration_truck_min)>30 and
round(duration_truck_min)<=60:
                                truck_cost_euros=5
                            elif round(duration_truck_min)>60 and
round(duration_truck_min)<=120:
                                truck_cost_euros=9
                            elif round(duration_truck_min)>120 and
round(duration_truck_min)<=240:
                                truck_cost_euros=11
                            else:

truck_cost_euros=(round(duration_truck_min)/60)*2
                            col16.append(truck_cost_euros)
                            col17.append(truck_plate_exit[i][1])
                    truck_duration.append(col15)
                    truck_cost.append(col16)
                    truck_plate_entry_exit.append(col17)

        bus_plate_entry_exit=[]
        bus_duration=[]
        bus_cost=[]
        for i in range(0, index_bus_exit):
            for j in range(0, index_bus_entry):
                if bus_plate_exit[i][1]==bus_plate_entry[j][1] and
bus_date_exit[i][1]>bus_date_entry[j][1]:
                    flag_bus_entry_exit=1
                    col18=[]
                    col19=[]
                    col20=[]
                    for k in range(0, 2):
                        if k == 0:
                            col18.append(index_bus_exit)
                            col19.append(index_bus_exit)
                            col20.append(index_bus_exit)
                        else:
                            duration_bus_min=((int(bus_date_exit[i][1][0:2])-
int(bus_date_entry[j][1][0:2]))*60)+(int(bus_date_exit[i][1][3:5])-
int(bus_date_entry[j][1][3:5]))+((int(bus_date_exit[i][1][6:8])-
int(bus_date_entry[j][1][6:8]))/60)
                            col18.append(round(duration_bus_min))
                            if round(duration_bus_min)>=0 and
round(duration_bus_min)<=30:
                                bus_cost_euros=0
                            elif round(duration_bus_min)>30 and
round(duration_bus_min)<=60:
                                bus_cost_euros=7
                            elif round(duration_bus_min)>60 and
```

```
round(duration_bus_min)<=120:
                                bus_cost_euros=16
                            elif round(duration_bus_min)>120 and
round(duration_bus_min)<=240:
                                bus_cost_euros=19
                            else:
                                bus_cost_euros=(round(duration_bus_min)/60)*3
                            col19.append(bus_cost_euros)
                            col20.append(bus_plate_exit[i][1])
                    bus_duration.append(col18)
                    bus_cost.append(col19)
                    bus_plate_entry_exit.append(col20)




        dt8 = datetime.now()
        str_day2 = dt8.strftime("%d")
        day2 = int(str_day2)
        if (day2-day)==1:
            break
        end1 = input("If you want the program to stop running the day type
number 5:")
        if end1 == '5':
            break
    if flag_car_entry_exit==1:
     for i in range(0, index_car_exit):
        print('The plate of the ',i+1,'st car that had first entry and then
exit today was:',car_plate_entry_exit[i][1])
        print('The costs of the ',i+1,'st car that had first entry and then
exit today was:',car_cost[i][1])
        print('The duration in minutes of the ',i+1,'st car that had first
entry and then exit today was:',car_duration[i][1])
    print('The number of cars that had entry first and exited then today
were', index_car_exit)
    if flag_truck_entry_exit==1:
     for i in range(0, index_truck_exit):
        print('The plate of the ',i+1,'st truck that had first entry and then
exit today was:',truck_plate_entry_exit[i][1])
        print('The costs of the ',i+1,'st truck that had first entry and then
exit today was:',truck_cost[i][1])
        print('The duration in minutes of the ',i+1,'st truck that had first
entry and then exit today was:',truck_duration[i][1])
     print('The number of trucks that had entry first and exited then today
were', index_truck_exit)
    if flag_bus_entry_exit==1:
     for i in range(0, index_bus_exit):
        print('The plate of the ',i+1,'st bus that had first entry and then
exit today was:',bus_plate_entry_exit[i][1])
        print('The costs of the ',i+1,'st bus that had first entry and then
exit today was:',bus_cost[i][1])
        print('The duration in minutes of the ',i+1,'st bus that had first
entry and then exit today was:',bus_duration[i][1])
     print('The number of buses that had entry first and exited then today
were', index_bus_exit)
    total_vehicles=index_car_exit + index_truck_exit + index_bus_exit
```

```
    print('The total number of cars,trucks and buses that had entry first and
exited then today were',total_vehicles)
    end = input("If you want the program to stop running type number 5:")
    if end=='5':
        break
```

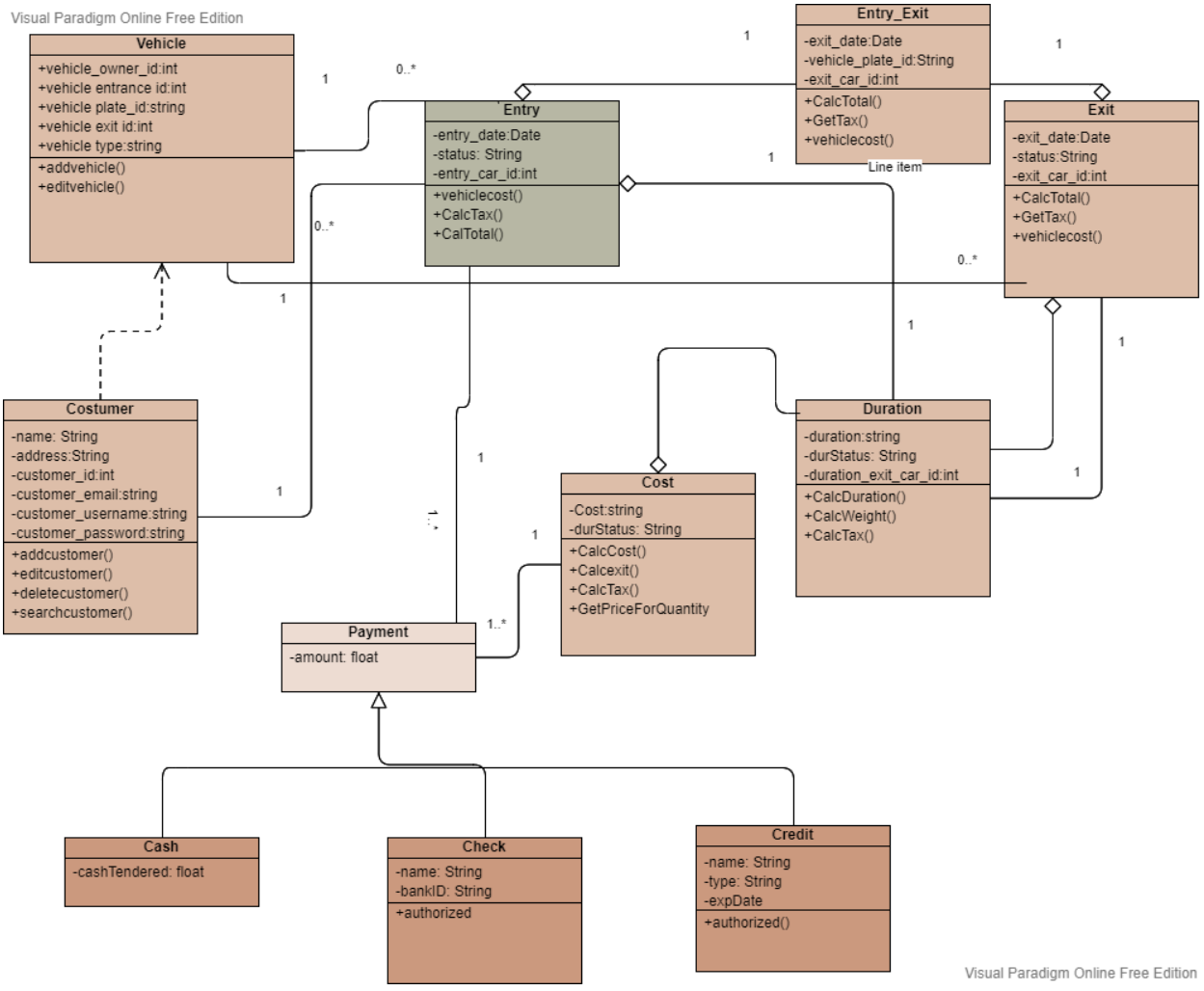
Τέλος να πω ότι χρησιμοποίησα Pycharm Editor 2020.2 για να τρέξω τον κώδικα και χρησιμοποίησα αντίστοιχα κάποιον interpreter της Python.

Μιας και εδώ δεν φαίνεται καλά ο κώδικας και για σκοπούς run θα σας στείλω όπως είπα ξεχωριστά το Python αρχείο.

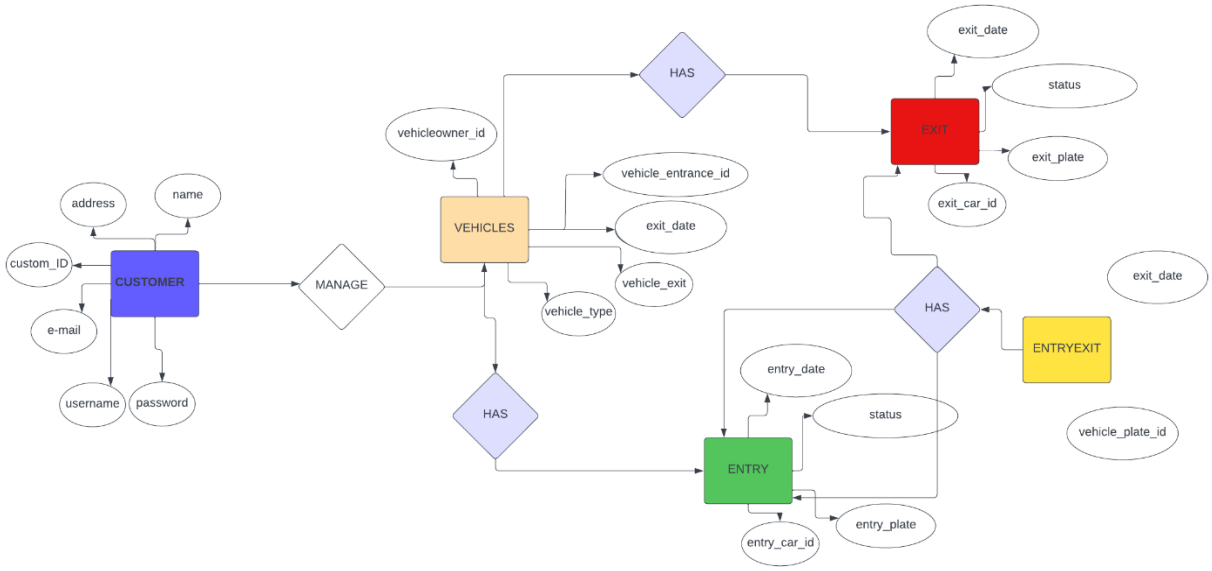
Στον κώδικά μου δεν χρησιμοποίησα ούτε κλάσεις ούτε αντικείμενα. Απλά και μόνο πίνακες(δυσδιάστατες λίστες) με δυναμική δέσμευση μνήμης.

4.Παρακαλούμε σχεδιάστε σε UML ή σε σχηματική αναπαράσταση τις κλάσεις τις οποίες θα χρησιμοποιήσετε για να σχεδιάσετε το εν λόγω σύστημα. Για κάθε κλάση μπορείτε να σημειώσετε ορισμένα τυπικά χαρακτηριστικά της/ πεδία που αυτή περιέχει. Επίσης θα πρέπει να αποτυπώσετε τις σχέσεις μεταξύ των κλάσεων.

Αν και στον κώδικά μου δεν χρησιμοποίησα ούτε κλάσεις ούτε αντικείμενα. Απλά και μόνο πίνακες(δυσδιάστατες λίστες) με δυναμική δέσμευση μνήμης.Θα δείξω πώς θα ήταν ένα αντίστοιχο σύστημα ενός Class Diagram ενός parking.Χρησιμοποίησα **https://online.visual-paradigm.com/app/diagrams**



5. Σχεδιάστε το σχήμα δεδομένων (ER διάγραμμα ή DML εντολές για την δημιουργία ενός σχήματος σχεσιακής βάσης δεδομένων).



Εδώ βλέπουμε τη σχεσιακή βάση δεδομένων που θα δοθεί και σε ξεχωριστό αρχείο..

Λόγω εργαλείου free trial δεν σχεδίασα την οντότητα **ΚΟΣΤΟΣ** και την οντότητα **ΠΛΗΡΩΜΗ**

Χρησιμοποίησα **https://lucid.app/lucidchart**