**Užduotis:**

**Paskaičiuoti pelną pardavus visas prekes. Informacija apie kainąir kiekius yra pateikiama kaip žodynai (dictionary):d1 = {"cost\_price": 32.67,"sell\_price": 45.00,"inventory": 1200}d2 = {"cost\_price": 225.89,"sell\_price": 550.00,"inventory": 100}d3 = {"cost\_price": 2.77,"sell\_price": 7.95,"inventory": 8500}**

#Paskaičiuoti pelną pardavus visas prekes. Informacija apie kainą

#ir kiekius yra pateikiama kaip žodynai (dictionary):

d1 = {"cost\_price": 32.67,"sell\_price": 45.00,"inventory": 1200}

d2 = {"cost\_price": 225.89,"sell\_price": 550.00,"inventory": 100}

d3 = {"cost\_price": 2.77,"sell\_price": 7.95,"inventory": 8500}

# Funkcija pelnui apskaičiuoti

def calculate\_profit(item):

    cost\_price = item["cost\_price"]

    sell\_price = item["sell\_price"]

    inventory = item["inventory"]

    profit\_per\_item = sell\_price - cost\_price

    total\_profit = profit\_per\_item \* inventory

    return total\_profit

# Skaičiuojame pelną kiekvienam žodynui

profit\_d1 = calculate\_profit(d1)

profit\_d2 = calculate\_profit(d2)

profit\_d3 = calculate\_profit(d3)

# Bendras pelnas

total\_profit = profit\_d1 + profit\_d2 + profit\_d3

# Rezultatų spausdinimas

print(f"Pelnas pardavus visas d1 prekes: {profit\_d1:.2f} EUR")

print(f"Pelnas pardavus visas d2 prekes: {profit\_d2:.2f} EUR")

print(f"Pelnas pardavus visas d3 prekes: {profit\_d3:.2f} EUR")

print(f"Bendras pelnas pardavus visas prekes: {total\_profit:.2f} EUR")

**Užduotis:**

Parašyk programą Python kalba, kad paprašius vartotojo įvesti simbolį bei skaičių būtų Išvestas kvadratas, sudarytą iš įvesto simbolio ir kurio kraštinė būtų lygi įvestam skaičiui.

#Paprašykite vartotojo įvesti simbolį bei skaičių. Išveskite kvadratą, sudarytą iš įvesto simbolio ir kurio kraštinė būtų

# lygi įvestam skaičiui. Pvz.: įvesta '@' ir 5. Rezultatas:

def spausdinti\_kvadrata(simbolis, dydis):

    for \_ in range(dydis):

        print(simbolis \* dydis)

# Prašome vartotojo įvesti simbolį

simbolis = input("Įveskite simbolį: ")

# Prašome vartotojo įvesti skaičių

dydis = int(input("Įveskite skaičių,kvadrato kraštinės ilgį: "))

# Iškviečiame funkciją ir spausdiname kvadratą

spausdinti\_kvadrata(simbolis, dydis)

**Užduotis:**

# Duotas sąrašas miestų miestai = ['Vilnius', 'Kaunas', 'Alytus', 'Rokiškis', 'Ūla', 'Mažeikiai', 'Akmena'] #

Filtruojame ir spausdiname miestus, kurių pavadinimai ilgesni nei 6 simboliai

# Duotas sąrašas miestų

miestai = ['Vilnius', 'Kaunas', 'Alytus', 'Rokiškis', 'Ūla', 'Mažeikiai', 'Akmena']

# Filtruojame ir spausdiname miestus, kurių pavadinimai ilgesni nei 6 simboliai

for miestas in miestai:

    if len(miestas) > 6:

        print(miestas)

**Uzduotis**

Parašyti programą Python kalba, kuri paprašo vartotojo įvesti simbolį ir skaičių, o tada išveda eglutę, sudarytą iš įvesto simbolio, o jos dydis lygūs įvestam skaičiui:

# Funkcija, kuri sukuria ir atspausdina eglutę

def spausdinti\_eglute(simbolis, dydis):

    for i in range(1, dydis + 1):

        # Skaičiuojame tarpų kiekį

        tarpai = ' ' \* (dydis - i)

        # Skaičiuojame simbolių kiekį

        simboliai = simbolis \* (2 \* i - 1)

        # Spausdiname tarpų ir simbolių eilutę

        print(tarpai + simboliai)

# Prašome vartotojo įvesti simbolį

simbolis = input("Įveskite simbolį: ")

# Prašome vartotojo įvesti skaičių

dydis = int(input("Įveskite eglutės dydį: "))

# Iškviečiame funkciją ir spausdiname eglutę

spausdinti\_eglute(simbolis, dydis)

5

**Uzduotis**

Parašyti programą Python kalba, kuri paprašo vartotojo įvesti simbolį ir skaičių, o tada išveda pusinę eglutę, sudarytą iš įvesto simbolio, o jos dydis lygūs įvestam skaičiui:

# Funkcija, kuri sukuria ir atspausdina eglutę

def spausdinti\_eglute(simbolis, dydis):

    for i in range(1, dydis + 1):

        # Skaičiuojame tarpų kiekį

        tarpai = ' ' \* (dydis - i)

        # Skaičiuojame simbolių kiekį

        simboliai = simbolis \* ((2 \* i - 1)//2)

        # Spausdiname tarpų ir simbolių eilutę

        print(tarpai + simboliai)

# Prašome vartotojo įvesti simbolį

simbolis = input("Įveskite simbolį: ")

# Prašome vartotojo įvesti skaičių

dydis = int(input("Įveskite eglutės dydį: "))

# Iškviečiame funkciją ir spausdiname eglutę

spausdinti\_eglute(simbolis, dydis)

Atspausdins

^

^^

^^^

Jei

# Funkcija, kuri sukuria ir atspausdina pusinę eglutę

def spausdinti\_pusine\_eglute(simbolis, dydis):

    for i in range(1, dydis + 1):

        # Spausdiname simbolius

        print(simbolis \* i)

# Prašome vartotojo įvesti simbolį

simbolis = input("Įveskite simbolį: ")

# Prašome vartotojo įvesti skaičių

dydis = int(input("Įveskite pusinės eglutės dydį: "))

# Iškviečiame funkciją ir spausdiname pusinę eglutę

spausdinti\_pusine\_eglute(simbolis, dydis)

Atspausdins

@

@@

@@@

@@@@

#Paprašykite vartotojo įvesti simbolį bei skaičių. Išveskite

#kvadratą, sudarytą iš įvesto simbolio ir kurio kraštinė būtų

#lygi įvestam skaičiui.

# Prašome vartotojo įvesti simbolį

simbolis = input("Įveskite simbolį: ")

# Prašome vartotojo įvesti skaičių

dydis = int(input("Įveskite skaičių, paišomo kvadrato kraštinės ilgį: "))

for  a in range(dydis):

        print(simbolis \* a)

# Iškviečiameą

#spausdinti\_kvadrata(simbolis, dydis)

**Užduotis:**

t = "Aš rytais mėgstu kavą su sumuštiniais ir arbatą"

• Išvestų po vieną simbolį iš teksto. Šalia nurodykite ir simbolio poziciją (indeksą) tekste.

# Pradinis tekstas

t = "Aš rytais mėgstu kavą su sumuštiniais ir arbatą"

# Išvesti po vieną simbolį iš teksto su jų pozicijomis

print("Simboliai su pozicijomis:")

for i, simbolis in enumerate(t):

    print(f"{i}: {simbolis}")

• Išvesdami simbolį, spausdinkite jį viršutiniame registre, jei jis yra lyginėje pozicijoje. Kitu atveju- žemutiniame registre.

# Pradinis tekstas

t = "Aš rytais mėgstu kavą"

# Išvesti po vieną simbolį iš teksto su jų pozicijomis ir formatuojant pagal poziciją

print("Simboliai su pozicijomis ir formatu:")

for i, simbolis in enumerate(t):

    if i % 2 == 0:

        print(f"{i}: {simbolis.upper()}")

    else:

        print(f"{i}: {simbolis.lower()}")

• Išveskite po vieną žodį.

# Pradinis tekstas

t = "Aš rytais mėgstu kavą"

print("\nPo vieną žodį:")

# Išvesti po vieną žodį

zodziai = t.split()

for zodis in zodziai:

    print(zodis)

• Išveskite tik tuos žodžius, kurie yra ilgesni nei nurodytas simbolių kiekis, ir savyje turi nurodytą tekstą. Ilgį ir tekstą nurodo vartotojas.

# Pradinis tekstas

t = "Aš rytais mėgstu kavą su sumuštiniais ir arbatą su pyragu"

# Prašome vartotojo įvesti ilgį ir tekstą

min\_ilgis = int(input("\n Įveskite minimalų žodžio ilgį: "))

tekstas = input("Įveskite tekstą, kurį turi turėti žodis: ")

zodziai=t.split()

print("\n Žodžiai, kurie yra ilgesni nei nurodytas simbolių kiekis ir turi nurodytą tekstą:")

# Išvesti tik tuos žodžius, kurie yra ilgesni nei nurodytas simbolių kiekis ir turi nurodytą tekstą

for zodis in zodziai:

    if len(zodis) > min\_ilgis and tekstas in zodis:

        print(zodis)

• Išveskite tekstą (visą vienu metu), kuriame būtų kas antras žodis parašytas didžiosiomis raidėmis.

# Pradinis tekstas

t = "Aš rytais mėgstu kavą su sumuštiniais ir arbatą su pyragu"

zodis=t.split()

naujas\_tekstas = " ".join([zodis.upper() if i % 2 != 0 else zodis for i, zodis in enumerate(zodziai)])

print(naujas\_tekstas)

• Išveskite tekstą su kas 𝑛-tuoju simboliu viršutiniame registre. 𝑛 įveda vartotojas\*

# Pradinis tekstas

t = "Aš rytais mėgstu kavą su sumuštiniais ir arbatą su pyragu"

zodis=t.split()

n=int(input('Ivesk registra'))

naujas\_tekstas\_su\_n = " ".join([simbolis.upper() if (i + 1) % n == 0 else simbolis for i, simbolis in enumerate(t)])

print(naujas\_tekstas\_su\_n)

**Užduotis:**

Duoti sąrašai:

a = [1,2,3,4,5]

b = [-7,-2,0,1,4]

• Suskaičiuokite:

c = a + b, d = a ⋅ b, e =a/b

 # Duoti sąrašai

a = [1, 2, 3, 4, 5]

b = [-7, -2, 0, 1, 4]

# Funkcija, kuri sudeda sąrašų elementus

def sudeti\_sarasus(a, b):

    return [x + y for x, y in zip(a, b)]

# Funkcija, kuri padaugina sąrašų elementus

def padauginti\_sarasus(a, b):

    return [x \* y for x, y in zip(a, b)]

# Funkcija, kuri padalina sąrašų elementus

def padalinti\_sarasus(a, b):

    # Apsauga nuo dalybos iš nulio

    return [x / y if y != 0 else 'N/A' for x, y in zip(a, b)]

# Atlikti operacijas

sudetis = sudeti\_sarasus(a, b)

daugyba = padauginti\_sarasus(a, b)

dalyba = padalinti\_sarasus(a, b)

# Spausdinti rezultatus

print(f"Sąrašų sudėtis: {sudetis}")

print(f"Sąrašų daugyba: {daugyba}")

print(f"Sąrašų dalyba: {dalyba}")

**Užduotis:**

.• Sugeneruokite skaičių sąrašą nuo n iki k, kas m. n, k, m įveda

vartotojas.

• Suraskite šio sąrašo skaičių sumą, aritmetinius ir geometrinius

vidurkius.

import math

# Funkcija sugeneruoti skaičių sąrašą

def generuoti\_sarasa(n, k, m):

    return list(range(n, k + 1, m))

# Funkcija apskaičiuoti skaičių sumą

def skaiciu\_suma(sarasas):

    return sum(sarasas)

# Funkcija apskaičiuoti aritmetinį vidurkį

def aritmetinis\_vidurkis(sarasas):

    return sum(sarasas) / len(sarasas) if sarasas else 0

# Funkcija apskaičiuoti geometrinį vidurkį

def geometrinis\_vidurkis(sarasas):

    if not sarasas:

        return 0

    sandauga = 1

    for sk in sarasas:

        sandauga \*= sk

    return math.pow(sandauga, 1 / len(sarasas))

# Prašome vartotojo įvesti n, k, m

n = int(input("Įveskite n reikšmę: "))

k = int(input("Įveskite k reikšmę: "))

m = int(input("Įveskite m reikšmę: "))

# Sugeneruojame skaičių sąrašą

sarasas = generuoti\_sarasa(n, k, m)

# Apskaičiuojame sumą, aritmetinį ir geometrinį vidurkius

suma = skaiciu\_suma(sarasas)

aritmetinis = aritmetinis\_vidurkis(sarasas)

geometrinis = geometrinis\_vidurkis(sarasas)

# Spausdiname rezultatus

print(f"Sugeneruotas sąrašas: {sarasas}")

print(f"Skaičių suma: {suma}")

print(f"Aritmetinis vidurkis: {aritmetinis}")

print(f"Geometrinis vidurkis: {geometrinis}")

**Užduotis:**

[('BMW', 'M5', '253 000 km', '2005-12', '19 500'), ('Mitsubishi', 'Outlander', '195 405 km', '2006', '2 700'), ('Seat', 'Arona', '129 800 km', '2019-11', '13 490'), ('Skoda', 'Octavia Scout', '183 000 km', '2015-06', '14 650'), ('Volvo', 'XC90', '127 896 km', '2018-08', '38 000'), ('Audi', '80', '194 000 km', '1987-07', '1 200'), ('Dacia', 'Lodgy', '102 122 km', '2013-03', '8 000'), ('Skoda', 'Octavia', '227 937 km', '2017-11', '11 990'), ('BMW', '320', '158 486 km', '2017-07', '16 190'), ('Volkswagen', 'Passat', '150 869 km', '2019-01', '16 490'), ('BMW', '520', '74 455 km', '2018-03', '28 900'), ('Mercedes-Benz', 'GLE Coupe 350', '194 000 km', '2016-11', '42 500'), ('Ford', 'C-MAX', '276 260 km', '2005-06', '2 000'), ('BMW', 'X3', '290 000 km', '2004-11', '3 390'), ('BMW', '530', '400 000 km', '2004', '3 750')]

#Suraskite vidutinę kainą, amžių ir rida

from datetime import datetime

# Automobilių duomenys

cars = ['''

    ('BMW', 'M5', '253 000 km', '2005-12', '19 500'),

    ('Mitsubishi', 'Outlander', '195 405 km', '2006', '2 700'),

    ('Seat', 'Arona', '129 800 km', '2019-11', '13 490'),

    ('Skoda', 'Octavia Scout', '183 000 km', '2015-06', '14 650'),

    ('Volvo', 'XC90', '127 896 km', '2018-08', '38 000'),

    ('Audi', '80', '194 000 km', '1987-07', '1 200'),

    ('Dacia', 'Lodgy', '102 122 km', '2013-03', '8 000'),

    ('Skoda', 'Octavia', '227 937 km', '2017-11', '11 990'),

    ('BMW', '320', '158 486 km', '2017-07', '16 190'),

    ('Volkswagen', 'Passat', '150 869 km', '2019-01', '16 490'),

    ('BMW', '520', '74 455 km', '2018-03', '28 900'),

    ('Mercedes-Benz', 'GLE Coupe 350', '194 000 km', '2016-11', '42 500'),

    ('Ford', 'C-MAX', '276 260 km', '2005-06', '2 000'),

    ('BMW', 'X3', '290 000 km', '2004-11', '3 390'),

    ('BMW', '530', '400 000 km', '2004', '3 750')'''

]

# Funkcija konvertuoti kainą į skaičių

def convert\_price(price\_str):

    return float(price\_str.replace(' ', '').replace('EUR', ''))

# Funkcija konvertuoti ridą į skaičių

def convert\_mileage(mileage\_str):

    return int(mileage\_str.replace(' km', '').replace(' ', ''))

# Funkcija gauti automobilio amžių

def calculate\_age(date\_str):

    current\_year = datetime.now().year

    if '-' in date\_str:

        year, month = map(int, date\_str.split('-'))

        age = current\_year - year - (datetime.now().month < month)

    else:

        age = current\_year - int(date\_str)

    return age

# Skaičiuojame vidutinę kainą, ridą ir amžių

total\_price = sum(convert\_price(car[4]) for car in cars)

total\_mileage = sum(convert\_mileage(car[2]) for car in cars)

total\_age = sum(calculate\_age(car[3]) for car in cars)

average\_price = total\_price / len(cars)

average\_mileage = total\_mileage / len(cars)

average\_age = total\_age / len(cars)

# Rezultatų spausdinimas

print(f"Vidutinė kaina: {average\_price:.2f} EUR")

print(f"Vidutinis amžius: {average\_age:.2f} metai")

print(f"Vidutinė rida: {average\_mileage:.2f} km")

**Užduotis:**

1. a) susikurti 3 tuščius sąrašusb) parašyti ciklą, kuris:eitų per duotąjį sąrašą, paimtų ridą, kainą, metus, konvertuotų į skaičiussu append() konvertuotą skaičių pridėkite prie atitinkamo tuščio sąrašoc) kiekvienam sąrašui suraskite vidurkįtai ir bus ats.

auto = [('BMW', 'M5', '253 000 km', '2005-12', '19 500'), ('Mitsubishi', 'Outlander', '195 405 km', '2006', '2 700'), ('Seat', 'Arona', '129 800 km', '2019-11', '13 490'), ('Skoda', 'Octavia Scout', '183 000 km', '2015-06', '14 650'), ('Volvo', 'XC90', '127 896 km', '2018-08', '38 000'), ('Audi', '80', '194 000 km', '1987-07', '1 200'), ('Dacia', 'Lodgy', '102 122 km', '2013-03', '8 000'), ('Skoda', 'Octavia', '227 937 km', '2017-11', '11 990'), ('BMW', '320', '158 486 km', '2017-07', '16 190'), ('Volkswagen', 'Passat', '150 869 km', '2019-01', '16 490'), ('BMW', '520', '74 455 km', '2018-03', '28 900'), ('Mercedes-Benz', 'GLE Coupe 350', '194 000 km', '2016-11', '42 500'), ('Ford', 'C-MAX', '276 260 km', '2005-06', '2 000'), ('BMW', 'X3', '290 000 km', '2004-11', '3 390'), ('BMW', '530', '400 000 km', '2004', '3 750')]

ridos = []

kainos = []

metai = []

for eilute in auto:

    rida = int(eilute[2].replace(' ','').replace('km',''))

    amzius = 2024- int(eilute[3][0:4])

    kaina = int(eilute[-1].replace(' ',''))

    ridos.append(rida)

    kainos.append(kaina)

    metai.append(amzius)

print(sum(ridos)/len(ridos), sum(kainos)/len(kainos), sum(metai)/len(metai), sep='\n')

**Užduotis:**

Parašyti programą, kuri isrenka, kurio automobilio rida yra didžiaisia

cl = [{'Modelis':'WV', 'rida':25000},{'Modelis':'Audi', 'rida':36000},{'Modelis':'Ford','rida':42000},{'Modelis':'Toyota', 'rida':31000},{'Modelis':'Nissan', 'rida':75000}]**Kolegos programa**

b=<0f=cl[0]['rida']for a in cl: a['rida'] print(a['rida']) if int(a['rida']) > b: b=int(a['rida']) c=a['Modelis'] if int(a['rida']) < f: f=int(a['rida']) d=a['Modelis'] print('daugiausiai nuvaziavo', c)print('maziausiai nuvaziavo', d)

C

#Parašyti programą, kuri isrenka, kurio automobilio rida yra didžiaisia

cl =[{'Modelis':'WV', 'rida':25000},

     {'Modelis':'Audi', 'rida':36000},

     {'Modelis':'Ford', 'rida':42000},

     {'Modelis':'Toyota', 'rida':31000},

     {'Modelis':'Nissan', 'rida':75000}]

max\_rida = max(cl, key=lambda x: x['rida'])

print(f"Automobilis su didžiausia rida: {max\_rida['Modelis']}, rida: {max\_rida['rida']}")

Be max f-jos: BLOGAI. KAI YRA DU TOKIE PATYS DYDZIAI SPAUSDINA TIK VIENA>

#Parašyti programą, kuri isrenka, kurio automobilio rida yra didžiaisia

cl=[{'Modelis':'WV', 'rida':25000},

     {'Modelis':'Audi', 'rida':36000},

     {'Modelis':'Ford', 'rida':42000},

     {'Modelis':'Toyota', 'rida':31000},

     {'Modelis':'Nissan', 'rida':75000}]

 # Inicializuoti kintamuosius didžiausiai ridai ir atitinkamam automobiliui saugoti

max\_rida = -1

max\_rida\_auto = None

# Iteruoti per automobilių sąrašą ir rasti automobilį su didžiausia rida

for auto in cl:

    if auto['rida'] > max\_rida:

        max\_rida = auto['rida']

        max\_rida\_auto = auto

# Atspausdinti rezultatą

print(f"Automobilis su didžiausia rida: {max\_rida\_auto['Modelis']}, rida: {max\_rida\_auto['rida']}")

Užduotis:

Parašyti programą, kuri išrinktų Auto su mažiausia rida.

#Parašyti programą, kuri isrenka, kurio automobilio rida yra maziausia

cl=[{'Modelis':'WV', 'rida':25000},

     {'Modelis':'Audi', 'rida':36000},

     {'Modelis':'Ford', 'rida':42000},

     {'Modelis':'Toyota', 'rida':31000},

     {'Modelis':'Nissan', 'rida':75000}]

max\_rida\_auto = max(cl, key=lambda x: x['rida'])

min\_rida\_auto = None

min\_rida\_aoto=max\_rida\_auto

for auto in cl:

    if auto['rida'] < max\_rida:

        min\_rida = auto['rida']

        min\_rida\_auto = auto

print(f"Automobilis su maziausia rida yra {min\_rida\_auto['Modelis']}, jo rida= {min\_rida\_auto['rida']}",'km')

**Uzduotis**

Parašykite kodą, kuris paklaustų vartotojo skaičių A ir B, matematinio veiksmo (+, −, /, ∗), atliktų veiksmą, rezultatą išspausdintų. Pritaikykite try bloką, kad išvengtumėte tokiųsituacijų kaip:• dalyba iš 0;• vietoj skaičiaus tekstas;Pritaikykite visas try bloko šakas.

try:

    A = float(input('A?'))

    B = float(input('B?'))

    m = input('+-/\*?')

    if m == '+':

        print(A+B)

    if m == '/':

        try:

            print(A/B)

        except ZeroDivisionError as ex:

            print(ex, 'Dalyba iš nulio negalima')

except ValueError as ex:

    print(ex, 'Įvesti neteisingi duomenys')

else:

    print('Matematika suveikė')

finally:

    print('Demo pabaiga')

Mano, pataisytas ChatGPT

A = input('ivesk skaiciu A: ')

B = input('ivesk kitas skaiciu B: ')

m = input('ivest matematinio veiksmo zenkla: ')

print(f'A={A}', f'B={B}', f'm yra {m}')

try:

    A = float(A)

    B = float(B)

    if m == '+':

        print('A+B=', A + B)

    elif m == '-':

        print('A-B=', A - B)

    elif m == '\*':

        print('A\*B=',A \* B)

    elif m == '/':

        try:

            print('A/B=', A / B, 'B/A=', B/A)

        except ZeroDivisionError as ex:

            print(ex, 'Dalyba iš nulio negalima')

    else:

        print('Neatpažintas matematinis veiksmas')

except ValueError as ex:

    print(ex, 'Įvesti neteisingi duomenys')

finally:

    print('sveikinu, suveike')

**Uzduotis:**

Parašykite funkciją PVM skaičiavimui. Pateikiami argumentai - kaina be pvm, tarifas. funkcija turi gražinti PVM kiekį

def pvm\_skaiciavimas(kaina\_be\_pvm, tarifas):

    try:

# Paverčiame tarifą procentais, jei jis pateiktas kaip sveikasis skaičius

        tarifas = tarifas / 100 if tarifas > 1 else tarifas

        pvm\_kiekis = kaina\_be\_pvm \* tarifas

        return pvm\_kiekis

    except TypeError:

  return "Įvesties duomenys turi būti skaičiai."

# Pavyzdys naudojimo:

kaina\_be\_pvm = 100.0  # Pavyzdžiui, 100 eurų

tarifas = 21  # Pavyzdžiui, 21% PVM tarifas

pvm\_kiekis = pvm\_skaiciavimas(kaina\_be\_pvm, tarifas)

print(f"PVM kiekis: {pvm\_kiekis:.2f} EUR")

1. Parašyti funkciją, kuri priima sąrašą ir grąžina naują sąrašą supaskutiniu ir pirmu elementais iš pirmojo sąrašo. PVZ:[1,2,3,4,5,6,7,8,9] ats: [9,1]

def pirmas\_paskutinis (sarasas):

    naujas\_sarasas = [sarasas[0], sarasas[-1]]

    return naujas\_sarasas

sarasas = [1, 2, 3, 4, 5, 6, 7, 8, 9]

ats= pirmas\_paskutinis(sarasas)

print(f"Naujas sąrašas: {ats}")

2. Parašyti funkciją greičiui tikrinti: Funkcija paima argumentą -automobilio greitį. Jei greitis 50 ar mažesnis, funkcija grąžina”Ok”, jei greitis didesnis nei 50 - už kiekvieną 5km/h greičioviršijimą duodamas baudos taškas (pvz: greitis 70, atsakymas- 4) Jei surenkami 8 taškai ir daugiau - funkcija ne tik grąžinataškų sumą, bet ir priduria, jog vairuotojo teisės atimamos.

#Parašyti funkciją greičiui tikrinti:

# Funkcija paima argumentą - automobilio greitį.

# Jei greitis 50 ar mažesnis, funkcija grąžina ”Ok”,

# jei greitis didesnis nei 50 - už kiekvieną 5km/h greičio viršijimą duodamas baudos taškas

# (pvz: greitis 70, atsakymas - 4) Jei surenkami 8 taškai ir daugiau - funkcija ne tik grąžina taškų sumą,

# bet ir priduria, jog vairuotojo teisės atimamos.

def greicio\_virsijimas(greitis):

    if greitis<=50:

        print("Dekui uz saugu vairavima")

    else:

     if greitis>50:

         baudos\_taskai = (greitis - 50) // 5

         print(baudos\_taskai)

         if baudos\_taskai >= 8:

             return f"{baudos\_taskai} baudos taškai. Vairuotojo teisės atimamos."

         else:

            return f"{baudos\_taskai} baudos taškai."

greitis = 190

rezultatas = greicio\_virsijimas(greitis)

print(f"Greitis: {greitis} km/h - Rezultatas: {rezultatas}")

3. Parašyti funkciją, kuri priima tekstą ir grąžina atgal tekstą suapkeistom pirma ir paskutine raidėmis. PVZ: Rytas sytaR.

#Parašyti funkciją, kuri priima tekstą ir grąžina atgal tekstą su

#apkeistom pirma ir paskutine raidėmis. PVZ: Rytas sytaR.

def apkeisti\_pirma\_ir\_paskutine\_raide(tekstas):

        naujas\_tekstas = tekstas[-1] + tekstas[1:-1] + tekstas[0]

        return naujas\_tekstas

tekstas = "Rytas"

rezultatas = apkeisti\_pirma\_ir\_paskutine\_raide(tekstas)

print(f"Originalus tekstas: {tekstas}")

print(f"Tekstas su apkeistom pirma ir paskutine raidėmis: {rezultatas}")

#1.0.30 Parašyti paprastą konverterį, kaip funkciją, tarp skirtingų temperatūros

#skalių.Vartotojas įveda temperatūrą tokiu formatu ’xxC’, ’yyF’, tarp temperatūros dimensijos

# \*ir skaičių gali būti tarpas. Programa atpažįsta, ar įvesta Celsijaus laipsniai, ar Farenheito, ir pagal

#tai atlieka konvertavimą į °F arba į °C.

def convert\_temperature(temp):

    temp = temp.strip()  # Remove any leading/trailing spaces

    value = float(temp[:-1].strip())  # Get the numerical part

    scale = temp[-1].upper()  # Get the last character (scale)

    if scale == 'C':

        t\_F = (9/5) \* value + 32

        return f"{t\_F:.2f}F"

    elif scale == 'F':

        t\_C = (value - 32) \* 5/9

        return f"{t\_C:.2f}C"0

    else:

        return "Neteisingas formatas"

# Example usage:

temp = input("Įveskite temperatūrą laipsniais Celsijaus ar Farenheito (pvz., '25C' arba '77F'): ")

result = convert\_temperature(temp)

print(result)\*

Uzduotei – parasyti zodi atbuline tvarka. Trumpiausia programa.

def text(x): a = '' for i in x: a = i + a return atext('Dangus')

def kitapuse(zod):

#return zod[::-1]

i=len(zod)-1

doz=''

while i >= 0:

doz=doz+zod[i]

i=i-1

return doz

print(kitapuse('Labas 1234'))

Lukas 6:20 PM

def reverse\_string(string): return "".join(reversed(string))reverse\_string("Labas Vakaras!")

Sandauga skai2iu, kai nezinai, kiek juu bus ivest

def sandauga(\*pos3):s = 1for i in pos3:s = s\*ireturn ssandauga(4,5,6,7,8,9)

#8)

#duomenys:[('BMW', '253 000 km', '19 500', '4999 cm³, 507 AG (373kW)', 'Benzinas'), ('Mitsubishi', '195 405 km', '2 700', '2378 cm³, 160 AG (118kW)', 'Benzinas'), ('Seat', '129 800 km', '13 490', '1598 cm³, 95 AG (70kW)', 'Dyzelinas'), ('Skoda', '183 000 km', '14 650', '1798 cm³, 179 AG (132kW)', 'Benzinas'), ('Volvo', '127 896 km', '38 000', '1969 cm³, 235 AG (173kW)', 'Dyzelinas')]

#parašyti funkciją, kuri juos spausdintų  lenteleje Auto, rida, sk, turis, AG, KW,benzinas

duomenys = [

    ('BMW', '253 000 km', '19 500', '4999 cm³', '507 AG (373kW)', 'Benzinas'),

    ('Mitsubishi', '195 405 km', '2 700', '2378 cm³', '160 AG (118kW)', 'Benzinas'),

    ('Seat', '129 800 km', '13 490', '1598 cm³', '95 AG (70kW)', 'Dyzelinas'),

    ('Skoda', '183 000 km', '14 650', '1798 cm³', '179 AG (132kW)', 'Benzinas'),

    ('Volvo', '127 896 km', '38 000', '1969 cm³', '235 AG (173kW)', 'Dyzelinas')

]

# Funkcija spausdinanti lentelę

def spausdinti\_lentele(duomenys):

    # Lentelės antraštės

    print(f"{'Auto':<12} {'Rida':<12} {'Kaina':<10} {'Tūris':<10} {'AG':<15} {'KWh':<10} {'Kuras':<10}")

    print("="\*80)

    # Spausdina kiekvieną įrašą

    for auto in duomenys:

        auto\_pavadinimas, rida, kaina, turis, ag\_kwh, kuras = auto

        ag, kwh = ag\_kwh.split(' (')

        kwh = kwh.rstrip(')')

        print(f"| {auto\_pavadinimas:<10} | {rida:<10} | {kaina:<8} | {turis:<8} | {ag:<8} | {kwh:<8} | {kuras:<10} |")

# Iškviečiame funkciją

spausdinti\_lentele(duomenys)

#5. Parašyti funkciją, kurioje sugeneruojamos dvi skaičių sekos

#(jos gali būti skirtingo dydžio), programa turi grąžinti sąrašą

#skaičių, kurie sutampa tarp dviejų sugeneruotu sąrašų,

#galutiniame sąraše neturi būti skaičių dublikatų. PVZ: x =[1, 1, 2, 3, 1] y = [1, 3, 5, 6] atsakymas = [1,3]

import random

def sugeneruoti\_sarasus():

    # Sugeneruojame dvi skaičių sekas su skirtingais dydžiais

    x = [random.randint(1,10) for i in range(random.randint(5,10))]

    y = [random.randint(1,10) for a in range(random.randint(5,10))]

    return x, y

def rasti\_sutampancius(x, y):

    # Randa skaičius, kurie sutampa tarp dviejų sąrašų

    sutampantys = list(set(x) & set(y))

    return sutampantys

# Pagrindinė programa

x, y = sugeneruoti\_sarasus()

print(f"x = {x}")

print(f"y = {y}")

sutampantys = rasti\_sutampancius(x, y)

print(f"atsakymas = {sutampantys}")

# Parašykite kodą, kuris sukurtų tekstinį failą iš trijų stulpelių.

# Pirmąjame stulpelyje skaičiai nuo 1 iki 25-kių,

# antrame - atsitiktiniai skaičiau - ribos 10 - 1000,

# trečiame stulpelyje - atsitiktiniai skaičiai tarp 1 ir 10,

# tačiau - float tipo, turintys 4turis ženklus po kablelio

# stulpeliai atskirti viens nuo kito ;

# Pavyzdys, kas maždaug turi gautis:1;25;1.45672;42;7.5213

# ir maždaug taip toliau

import random

# Funkcija sugeneruojanti atsitiktinius skaičius ir rašanti į failą

def generuoti\_faila(filename):

    file\_numbers=open(filename, mode='w', encoding="utf-8")

    for i in range(1, 26):

            antras\_stulpelis = random.randint(10, 1000)

            trecias\_stulpelis = round(random.uniform(1, 10), 4)

            file\_numbers.write(f"{i};{antras\_stulpelis};{trecias\_stulpelis:.4f}\n")

# Sukuriame failą

generuoti\_faila('skaiciai.txt')

# Parašykite kodą, kuris sukurtų tekstinį failą iš trijų stulpelių.

# Pirmąjame stulpelyje skaičiai nuo 1 iki 25-kių,

# antrame - atsitiktiniai skaičiau - ribos 10 - 1000,

# trečiame stulpelyje - atsitiktiniai skaičiai tarp 1 ir 10,

# tačiau - float tipo, turintys 4turis ženklus po kablelio

# stulpeliai atskirti viens nuo kito ;

# Pavyzdys, kas maždaug turi gautis:1; 25; 1.45672; 42; 7.5213

# ir maždaug taip toliau

import random

# Funkcija sugeneruojanti atsitiktinius skaičius ir rašanti į failą

def generuoti\_faila(filename):

    file\_numbers=open(filename, mode='w', encoding="utf-8")

    for i in range(1, 26):

            antras\_stulpelis = random.randint(10, 1000)

            trecias\_stulpelis = round(random.uniform(1, 10), 4)

            eilute = f"{i}; {antras\_stulpelis}; {trecias\_stulpelis:.4f}"

            file\_numbers.write(eilute + "\n")

            print(eilute)

# Sukuriame failą

generuoti\_faila('skaiciai.txt')

Parašyti funkciją, kuri patikrintų, ar duotas tekstas

# yra palindromas (palindròmas (gr. palindromos – sugrįžtantis), žodis,

# žodžių grupė, sakinys, posakis ar eilėraštis,

# kuris skaitomas iš galo reiškia tą patį arba bent yra prasmingas.)

# sula - alus, ėmė, savas, sėdėk užu kėdės, level, Hanah.

# Ignoruoti raidžių registrą.

# Turi veikti ir su skaičiais (pvz 12321, 666, 85258, etc.)

# Vieno skaitmens skaičiai neturi būti laikomi palindromais.

# Minuso ženklą - kaip sugalvosite. Mokėti pagrįsti savo sprendimą.

import string

def is\_palindrome(text):

    # Pašaliname tarpus, skyrybos ženklus ir minusą

    cleaned\_text = ''.join(char for char in text if char.isalnum())

    # Jei tekstas tuščias arba vieno ilgio, jis nėra palindromas

    if len(cleaned\_text) <= 1:

        return False

    # Ignoruojame raidžių registrą

    cleaned\_text = cleaned\_text.lower()

    # Tikriname, ar tekstas yra palindromas

    return cleaned\_text == cleaned\_text[::-1]

# Testavimas

print(is\_palindrome("sula - alus"))        # True

su split f-ja.

def is\_palindrome(text):

    # Paverčiame tekstą į sąrašą, pašalinant nepageidaujamus simbolius

    unwanted\_chars = string.whitespace + string.punctuation + '-'

    text\_list = list(text)

    cleaned\_list = [char for char in text\_list if char not in unwanted\_chars]

    cleaned\_text = ''.join(cleaned\_list)

    # Jei tekstas tuščias arba vieno ilgio, jis nėra palindromas

    if len(cleaned\_text) <= 1:

        return False

    # Ignoruojame raidžių registrą

    cleaned\_text = cleaned\_text.lower()

    # Tikriname, ar tekstas yra palindromas

    return cleaned\_text == cleaned\_text[::-1]

print(is\_palindrome("mesa nevalgoma"))

Kolegos sprendimas uzdavinio:

# Parašykite kodą, kuris sukurtų tekstinį failą iš trijų stulpelių.

# Pirmąjame stulpelyje skaičiai nuo 1 iki 25-kių,

# antrame - atsitiktiniai skaičiau - ribos 10 - 1000,

# trečiame stulpelyje - atsitiktiniai skaičiai tarp 1 ir 10,

# tačiau - float tipo, turintys 4turis ženklus po kablelio

# stulpeliai atskirti viens nuo kito ;

# Pavyzdys, kas maždaug turi gautis:1; 25; 1.45672; 42; 7.5213

# ir maždaug taip toliau

#PADARYTA< KAD ATSAKYMAS BUTU FAILE IR ATSPAUSDINTAS EKRANE \_ PATOGIAU TIKRINTI.

f\_name = 'text08-06.txt'f = open(f\_name, mode='w', encoding='utf-8')for i in range(1,26): f.write(str(i)+';') f.write(str(random.randint(10,1000))+';') x = float(random.random()) f.write(f'{x:.4f}') f.write('\n') f.close()

Kolegu sprendimas  
#5. Parašyti funkciją, kurioje sugeneruojamos dvi skaičių sekos

#(jos gali būti skirtingo dydžio), programa turi grąžinti sąrašą

#skaičių, kurie sutampa tarp dviejų sugeneruotu sąrašų,

#galutiniame sąraše neturi būti skaičių dublikatų. PVZ: x =[1, 1, 2, 3, 1] y = [1, 3, 5, 6] atsakymas = [1,3]

def lst\_intersection(): lst1 = [] lst2 = [] i = 0 j = 0 a = random.randint(1, 10) b = random.randint(1, 10) while i != a: lst1.append(random.randint(1, 30)) i+= 1 while j != b: lst2.append(random.randint(1, 30)) j+= 1 return list(set(lst1).intersection(lst2))

KITAS TO PATIES UZDAVINIO SPRENDIMAS  
l1=[]l2=[]bendri=[]for i in range(0, 10): l1.append(random.randint(1, 10))for i in range(0, 30): l2.append(random.randint(1, 10))if len(l1)>=len(l2): for i in range(0, len(l1)): if (l1[i] in l2) and (l1[i] not in bendri) : bendri.append(l1[i])else: for i in range(0, len(l2)): if (l2[i] in l1) and (l2[i] not in bendri): bendri.append(l2[i])#ats=list(set(bendri))print(l1)print(l2)print(bendri)3

# Prisikelkite į darbinį aplanką įvairių failų : paveikslėlių,. csv, tekstinių, docx, pdf failų.

# Paklauskite vartotojo aplanko pavadinimo

# Sukurkite jį, jame sukurkite kitus atlankus, kurių pavadinimai būtų

# 'Tekstiniai failai', 'Paveikslėliai', 'Wordai', 'pdfai'

# ir tada surūšiuokite savo susirinktus failus į naujojo aplanko atitinkamus subfolderius.

# Jūs turite aplanką DATA su įvairiais failais. parašykite Python kodą, kuris padarytų šio aplanko kopiją,

# prie kiekvieno failo pavadinimo pradžios pridėdamas po tekstą "09p\_"

5

# Parašykite kodą, kuris ištrintų aplanką, kurio pavadinimą (arba kelią iki jo)

# įveda vartotojas.

# Kartu parašykite, kiek failų ištrynė jūsų kodas

# Parašykite kodą, kuris sukurtų jums tekstinį failą, kurio pavadinimą jūs įvedate.

# Prieš kurdami, patikrinkite, ar toks failas neegzistuoja.

# jei egzistuoa - praneškite, jog 'failas jau yra tokiu pavadinimu'.

# kiek čia buvo vyrų, moterų, koks jų buvo vidutinis amžius, kiek kaltininkų pasišalino?

data = [

    [{'dalyvisId': 47756477, 'kategorija': 'Automobilio vairuotojas', 'dalyvioTipas': 'Kitas asmuo', 'pasisalino': 'Ne', 'asmuoNezinomas': 'Ne', 'lytis': 'Moteris', 'pilietybe': None, 'amzius': 19, 'neigalumas': 'Ne', 'saugosDirzas': 'Užsegtas', 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': 'Nesuveikė', 'bukle': 'Nenukentėjo', 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': 'Atitinka', 'vairavimoStazas': 0, 'busena': 'Blaivus', 'detaliBusena': None, 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Taip', 'girtumasPromilemis': None, 'dalyvioBusena': 'KALTININKAS', 'teisenosStadijaBusena': 'ADMINISTRACINIS', 'dalyvioKetPazeidimai': ['Kiti Kelių eismo taisyklių pažeidimai'], 'tpId': 32912177}],

[{'dalyvisId': 47759080, 'kategorija': 'Automobilio vairuotojas', 'dalyvioTipas': None, 'pasisalino': 'Ne', 'asmuoNezinomas': 'Taip', 'lytis': None, 'pilietybe': None, 'amzius': None, 'neigalumas': None, 'saugosDirzas': None, 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': None, 'bukle': None, 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': None, 'vairavimoStazas': None, 'busena': None, 'detaliBusena': None, 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Ne', 'girtumasPromilemis': None, 'dalyvioBusena': 'NEPAZEIDE', 'teisenosStadijaBusena': None, 'dalyvioKetPazeidimai': [], 'tpId': 32913457}, {'dalyvisId': 47759079, 'kategorija': 'Automobilio vairuotojas', 'dalyvioTipas': 'Kitas asmuo', 'pasisalino': 'Ne', 'asmuoNezinomas': 'Ne', 'lytis': 'Vyras', 'pilietybe': None, 'amzius': 18, 'neigalumas': 'Ne', 'saugosDirzas': 'Užsegtas', 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': 'Nesuveikė', 'bukle': 'Nenukentėjo', 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': 'Atitinka', 'vairavimoStazas': 0, 'busena': 'Blaivus', 'detaliBusena': None, 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Taip', 'girtumasPromilemis': None, 'dalyvioBusena': 'KALTININKAS', 'teisenosStadijaBusena': 'ADMINISTRACINIS', 'dalyvioKetPazeidimai': ['Saugaus atstumo nesilaikymas'], 'tpId': 32913456}],

[{'dalyvisId': 47756490, 'kategorija': 'Automobilio vairuotojas', 'dalyvioTipas': 'Kitas asmuo', 'pasisalino': 'Ne', 'asmuoNezinomas': 'Ne', 'lytis': 'Moteris', 'pilietybe': None, 'amzius': 39, 'neigalumas': 'Ne', 'saugosDirzas': 'Užsegtas', 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': 'Nesuveikė', 'bukle': 'Nenukentėjo', 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': 'Atitinka', 'vairavimoStazas': 13, 'busena': 'Blaivus', 'detaliBusena': None, 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Ne', 'girtumasPromilemis': None, 'dalyvioBusena': 'NEPAZEIDE', 'teisenosStadijaBusena': None, 'dalyvioKetPazeidimai': [], 'tpId': 32912184}],

[{'dalyvisId': 48447053, 'kategorija': 'Automobilio vairuotojas', 'dalyvioTipas': 'Kitas asmuo', 'pasisalino': 'Ne', 'asmuoNezinomas': 'Ne', 'lytis': 'Moteris', 'pilietybe': None, 'amzius': 48, 'neigalumas': 'Ne', 'saugosDirzas': 'Užsegtas', 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': 'Nesuveikė', 'bukle': 'Nenukentėjo', 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': 'Atitinka', 'vairavimoStazas': 25, 'busena': 'Blaivus', 'detaliBusena': None, 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Taip', 'girtumasPromilemis': None, 'dalyvioBusena': 'KALTININKAS', 'teisenosStadijaBusena': 'ADMINISTRACINIS', 'dalyvioKetPazeidimai': ['Saugaus greičio nepasirinkimas'], 'tpId': 33386413}],

[{'dalyvisId': 47756497, 'kategorija': 'Automobilio vairuotojas', 'dalyvioTipas': 'Kitas asmuo', 'pasisalino': 'Taip', 'asmuoNezinomas': 'Ne', 'lytis': 'Vyras', 'pilietybe': None, 'amzius': 31, 'neigalumas': 'Ne', 'saugosDirzas': 'Užsegtas', 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': 'Nesuveikė', 'bukle': 'Nenukentėjo', 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': 'Atitinka', 'vairavimoStazas': 5, 'busena': 'Neblaivus', 'detaliBusena': 'Lengvas neblaivumo (girtumo) laipsnis', 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Taip', 'girtumasPromilemis': 1.48, 'dalyvioBusena': 'KALTININKAS', 'teisenosStadijaBusena': 'ADMINISTRACINIS', 'dalyvioKetPazeidimai': ['Saugaus greičio nepasirinkimas'], 'tpId': 32912187}],

[{'dalyvisId': 47759516, 'kategorija': 'Automobilio vairuotojas', 'dalyvioTipas': 'Kitas asmuo', 'pasisalino': 'Ne', 'asmuoNezinomas': 'Ne', 'lytis': 'Vyras', 'pilietybe': None, 'amzius': 18, 'neigalumas': 'Ne', 'saugosDirzas': 'Užsegtas', 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': 'Nesuveikė', 'bukle': 'Nenukentėjo', 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': 'Neturi teisės vairuoti', 'vairavimoStazas': 0, 'busena': 'Neblaivus', 'detaliBusena': 'Lengvas neblaivumo (girtumo) laipsnis', 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Taip', 'girtumasPromilemis': 0.78, 'dalyvioBusena': 'KALTININKAS', 'teisenosStadijaBusena': 'ADMINISTRACINIS', 'dalyvioKetPazeidimai': ['Kiti Kelių eismo taisyklių pažeidimai'], 'tpId': 32913682}],

[{'dalyvisId': 47808609, 'kategorija': 'Automobilio vairuotojas', 'dalyvioTipas': 'Kitas asmuo', 'pasisalino': 'Ne', 'asmuoNezinomas': 'Ne', 'lytis': 'Vyras', 'pilietybe': None, 'amzius': 53, 'neigalumas': 'Ne', 'saugosDirzas': 'Užsegtas', 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': 'Neįrengta', 'bukle': 'Sužeistas', 'detaliBukle': 'Gydomas ambulatoriškai', 'suzalojimoMastas': 'Atsisakyta nustatyti sveikatos sutrikdymo mastą', 'vairuotojoKvalifikacija': 'Atitinka', 'vairavimoStazas': 18, 'busena': 'Blaivus', 'detaliBusena': None, 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Taip', 'girtumasPromilemis': None, 'dalyvioBusena': 'KALTININKAS', 'teisenosStadijaBusena': 'ADMINISTRACINIS', 'dalyvioKetPazeidimai': ['Saugaus greičio nepasirinkimas'], 'tpId': 32939779}, {'dalyvisId': 47808610, 'kategorija': 'Keleivis', 'dalyvioTipas': 'Kitas asmuo', 'pasisalino': 'Ne', 'asmuoNezinomas': 'Ne', 'lytis': 'Vyras', 'pilietybe': None, 'amzius': 34, 'neigalumas': 'Ne', 'saugosDirzas': 'Užsegtas', 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': 'Neįrengta', 'bukle': 'Sužeistas', 'detaliBukle': 'Gydomas ambulatoriškai', 'suzalojimoMastas': 'Nežymus sveikatos sutrikdymas', 'vairuotojoKvalifikacija': None, 'vairavimoStazas': None, 'busena': 'Blaivus', 'detaliBusena': None, 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Ne', 'girtumasPromilemis': None, 'dalyvioBusena': 'NEPAZEIDE', 'teisenosStadijaBusena': None, 'dalyvioKetPazeidimai': [], 'tpId': 32939779}],

[{'dalyvisId': 47756539, 'kategorija': 'Automobilio vairuotojas', 'dalyvioTipas': 'Kitas asmuo', 'pasisalino': 'Ne', 'asmuoNezinomas': 'Ne', 'lytis': 'Vyras', 'pilietybe': None, 'amzius': 27, 'neigalumas': 'Ne', 'saugosDirzas': 'Užsegtas', 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': 'Nesuveikė', 'bukle': 'Nenukentėjo', 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': 'Atitinka', 'vairavimoStazas': 0, 'busena': 'Neblaivus', 'detaliBusena': 'Vidutinis neblaivumo (girtumo) laipsnis', 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Taip', 'girtumasPromilemis': 2, 'dalyvioBusena': 'KALTININKAS', 'teisenosStadijaBusena': 'ADMINISTRACINIS', 'dalyvioKetPazeidimai': ['Saugaus atstumo nesilaikymas'], 'tpId': 32912212}],

[{'dalyvisId': 47870990, 'kategorija': 'Automobilio vairuotojas', 'dalyvioTipas': 'Kitas asmuo', 'pasisalino': 'Taip', 'asmuoNezinomas': 'Ne', 'lytis': 'Vyras', 'pilietybe': None, 'amzius': 29, 'neigalumas': 'Ne', 'saugosDirzas': None, 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': None, 'bukle': 'Nenukentėjo', 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': 'Atitinka', 'vairavimoStazas': 11, 'busena': 'Blaivus', 'detaliBusena': None, 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Taip', 'girtumasPromilemis': None, 'dalyvioBusena': 'KALTININKAS', 'teisenosStadijaBusena': 'ADMINISTRACINIS', 'dalyvioKetPazeidimai': ['Saugaus greičio nepasirinkimas'], 'tpId': 32972622}],

[{'dalyvisId': 47767352, 'kategorija': 'Automobilio vairuotojas', 'dalyvioTipas': 'Kitas asmuo', 'pasisalino': 'Ne', 'asmuoNezinomas': 'Ne', 'lytis': 'Vyras', 'pilietybe': None, 'amzius': 38, 'neigalumas': 'Ne', 'saugosDirzas': 'Neužsegtas', 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': 'Nesuveikė', 'bukle': 'Nenukentėjo', 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': 'Atitinka', 'vairavimoStazas': 18, 'busena': 'Blaivus', 'detaliBusena': None, 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Ne', 'girtumasPromilemis': None, 'dalyvioBusena': 'NEPAZEIDE', 'teisenosStadijaBusena': None, 'dalyvioKetPazeidimai': [], 'tpId': 32917786}, {'dalyvisId': 47767353, 'kategorija': 'Automobilio vairuotojas', 'dalyvioTipas': None, 'pasisalino': 'Taip', 'asmuoNezinomas': 'Taip', 'lytis': None, 'pilietybe': None, 'amzius': None, 'neigalumas': None, 'saugosDirzas': None, 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': None, 'bukle': None, 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': None, 'vairavimoStazas': None, 'busena': None, 'detaliBusena': None, 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Taip', 'girtumasPromilemis': None, 'dalyvioBusena': 'KALTININKAS', 'teisenosStadijaBusena': 'ADMINISTRACINIS', 'dalyvioKetPazeidimai': ['Saugaus atstumo nesilaikymas'], 'tpId': 32917787}],

[{'dalyvisId': 47756551, 'kategorija': 'Automobilio vairuotojas', 'dalyvioTipas': 'Kitas asmuo', 'pasisalino': 'Ne', 'asmuoNezinomas': 'Ne', 'lytis': 'Vyras', 'pilietybe': None, 'amzius': 35, 'neigalumas': 'Ne', 'saugosDirzas': 'Užsegtas', 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': 'Nesuveikė', 'bukle': 'Nenukentėjo', 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': 'Atitinka', 'vairavimoStazas': 10, 'busena': 'Blaivus', 'detaliBusena': None, 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Ne', 'girtumasPromilemis': None, 'dalyvioBusena': 'NEPAZEIDE', 'teisenosStadijaBusena': None, 'dalyvioKetPazeidimai': [], 'tpId': 32912217}],

[{'dalyvisId': 47756561, 'kategorija': 'Automobilio vairuotojas', 'dalyvioTipas': 'Kitas asmuo', 'pasisalino': 'Ne', 'asmuoNezinomas': 'Ne', 'lytis': 'Vyras', 'pilietybe': None, 'amzius': 36, 'neigalumas': 'Ne', 'saugosDirzas': 'Užsegtas', 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': 'Nesuveikė', 'bukle': 'Nenukentėjo', 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': 'Atitinka', 'vairavimoStazas': 15, 'busena': 'Blaivus', 'detaliBusena': None, 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Ne', 'girtumasPromilemis': None, 'dalyvioBusena': 'NEPAZEIDE', 'teisenosStadijaBusena': None, 'dalyvioKetPazeidimai': [], 'tpId': 32912221}],

[{'dalyvisId': 47756590, 'kategorija': None, 'dalyvioTipas': None, 'pasisalino': 'Taip', 'asmuoNezinomas': 'Taip', 'lytis': None, 'pilietybe': None, 'amzius': None, 'neigalumas': None, 'saugosDirzas': None, 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': None, 'bukle': None, 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': None, 'vairavimoStazas': None, 'busena': None, 'detaliBusena': None, 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Taip', 'girtumasPromilemis': None, 'dalyvioBusena': 'GALIMAS\_KALTININKAS', 'teisenosStadijaBusena': None, 'dalyvioKetPazeidimai': ['Saugaus atstumo nesilaikymas'], 'tpId': 32912237}, {'dalyvisId': 47756589, 'kategorija': 'Kiti', 'dalyvioTipas': 'Kitas asmuo', 'pasisalino': 'Ne', 'asmuoNezinomas': 'Ne', 'lytis': 'Moteris', 'pilietybe': None, 'amzius': 32, 'neigalumas': 'Ne', 'saugosDirzas': None, 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': None, 'bukle': 'Nenukentėjo', 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': None, 'vairavimoStazas': None, 'busena': 'Blaivus', 'detaliBusena': None, 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Ne', 'girtumasPromilemis': None, 'dalyvioBusena': 'NEPAZEIDE', 'teisenosStadijaBusena': None, 'dalyvioKetPazeidimai': [], 'tpId': None}],

[{'dalyvisId': 47798291, 'kategorija': 'Kiti', 'dalyvioTipas': 'Kitas asmuo', 'pasisalino': 'Ne', 'asmuoNezinomas': 'Ne', 'lytis': 'Vyras', 'pilietybe': None, 'amzius': 28, 'neigalumas': 'Ne', 'saugosDirzas': None, 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': None, 'bukle': 'Nenukentėjo', 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': None, 'vairavimoStazas': None, 'busena': 'Blaivus', 'detaliBusena': None, 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Ne', 'girtumasPromilemis': None, 'dalyvioBusena': 'NEPAZEIDE', 'teisenosStadijaBusena': None, 'dalyvioKetPazeidimai': [], 'tpId': None}, {'dalyvisId': 47798292, 'kategorija': 'Automobilio vairuotojas', 'dalyvioTipas': 'Kitas asmuo', 'pasisalino': 'Ne', 'asmuoNezinomas': 'Ne', 'lytis': 'Vyras', 'pilietybe': None, 'amzius': 29, 'neigalumas': 'Ne', 'saugosDirzas': 'Užsegtas', 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': 'Nesuveikė', 'bukle': 'Nenukentėjo', 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': 'Neturi teisės vairuoti', 'vairavimoStazas': None, 'busena': 'Neblaivus', 'detaliBusena': None, 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Taip', 'girtumasPromilemis': 1.5, 'dalyvioBusena': 'KALTININKAS', 'teisenosStadijaBusena': 'ADMINISTRACINIS', 'dalyvioKetPazeidimai': ['Saugaus atstumo nesilaikymas'], 'tpId': 32934169}],

[{'dalyvisId': 47756953, 'kategorija': 'Automobilio vairuotojas', 'dalyvioTipas': 'Kitas asmuo', 'pasisalino': 'Ne', 'asmuoNezinomas': 'Ne', 'lytis': 'Vyras', 'pilietybe': None, 'amzius': 26, 'neigalumas': 'Ne', 'saugosDirzas': 'Užsegtas', 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': 'Nesuveikė', 'bukle': 'Nenukentėjo', 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': 'Atitinka', 'vairavimoStazas': 6, 'busena': 'Neblaivus', 'detaliBusena': None, 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Taip', 'girtumasPromilemis': 1.52, 'dalyvioBusena': 'KALTININKAS', 'teisenosStadijaBusena': 'ADMINISTRACINIS', 'dalyvioKetPazeidimai': ['Saugaus greičio nepasirinkimas'], 'tpId': 32912411}],

[{'dalyvisId': 47756607, 'kategorija': 'Automobilio vairuotojas', 'dalyvioTipas': 'Kitas asmuo', 'pasisalino': 'Ne', 'asmuoNezinomas': 'Ne', 'lytis': 'Moteris', 'pilietybe': None, 'amzius': 50, 'neigalumas': 'Ne', 'saugosDirzas': 'Užsegtas', 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': 'Nesuveikė', 'bukle': 'Nenukentėjo', 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': 'Atitinka', 'vairavimoStazas': 31, 'busena': 'Blaivus', 'detaliBusena': None, 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Ne', 'girtumasPromilemis': None, 'dalyvioBusena': 'NEPAZEIDE', 'teisenosStadijaBusena': None, 'dalyvioKetPazeidimai': [], 'tpId': 32912246}],

[{'dalyvisId': 47756632, 'kategorija': 'Automobilio vairuotojas', 'dalyvioTipas': None, 'pasisalino': 'Ne', 'asmuoNezinomas': 'Taip', 'lytis': None, 'pilietybe': None, 'amzius': None, 'neigalumas': None, 'saugosDirzas': None, 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': None, 'bukle': None, 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': None, 'vairavimoStazas': None, 'busena': None, 'detaliBusena': None, 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Ne', 'girtumasPromilemis': None, 'dalyvioBusena': 'NEPAZEIDE', 'teisenosStadijaBusena': None, 'dalyvioKetPazeidimai': [], 'tpId': 32912259}, {'dalyvisId': 47756629, 'kategorija': 'Automobilio vairuotojas', 'dalyvioTipas': 'Kitas asmuo', 'pasisalino': 'Taip', 'asmuoNezinomas': 'Ne', 'lytis': 'Vyras', 'pilietybe': None, 'amzius': 25, 'neigalumas': 'Ne', 'saugosDirzas': 'Užsegtas', 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': 'Nesuveikė', 'bukle': 'Nenukentėjo', 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': 'Neturi teisės vairuoti', 'vairavimoStazas': None, 'busena': 'Neblaivus', 'detaliBusena': 'Vidutinis neblaivumo (girtumo) laipsnis', 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Taip', 'girtumasPromilemis': 2.15, 'dalyvioBusena': 'KALTININKAS', 'teisenosStadijaBusena': 'ADMINISTRACINIS', 'dalyvioKetPazeidimai': ['Saugaus atstumo nesilaikymas'], 'tpId': 32912258}],

[{'dalyvisId': 47767014, 'kategorija': None, 'dalyvioTipas': None, 'pasisalino': 'Taip', 'asmuoNezinomas': 'Taip', 'lytis': None, 'pilietybe': None, 'amzius': None, 'neigalumas': None, 'saugosDirzas': None, 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': None, 'bukle': None, 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': None, 'vairavimoStazas': None, 'busena': None, 'detaliBusena': None, 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Taip', 'girtumasPromilemis': None, 'dalyvioBusena': 'GALIMAS\_KALTININKAS', 'teisenosStadijaBusena': None, 'dalyvioKetPazeidimai': ['Saugaus atstumo nesilaikymas'], 'tpId': 32917607}],

[{'dalyvisId': 47756694, 'kategorija': 'Automobilio vairuotojas', 'dalyvioTipas': 'Kitas asmuo', 'pasisalino': 'Ne', 'asmuoNezinomas': 'Ne', 'lytis': 'Vyras', 'pilietybe': None, 'amzius': 31, 'neigalumas': 'Ne', 'saugosDirzas': 'Užsegtas', 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': 'Nesuveikė', 'bukle': 'Nenukentėjo', 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': 'Atitinka', 'vairavimoStazas': 13, 'busena': 'Neblaivus', 'detaliBusena': 'Sunkus neblaivumo (girtumo) laipsnis', 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Taip', 'girtumasPromilemis': 3.03, 'dalyvioBusena': 'KALTININKAS', 'teisenosStadijaBusena': 'ADMINISTRACINIS', 'dalyvioKetPazeidimai': ['Saugaus atstumo nesilaikymas'], 'tpId': 32912289}],

[{'dalyvisId': 47838043, 'kategorija': 'Automobilio vairuotojas', 'dalyvioTipas': 'Kitas asmuo', 'pasisalino': 'Ne', 'asmuoNezinomas': 'Ne', 'lytis': 'Vyras', 'pilietybe': None, 'amzius': 52, 'neigalumas': 'Ne', 'saugosDirzas': 'Neįrengtas', 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': 'Neįrengta', 'bukle': 'Nenukentėjo', 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': 'Atitinka', 'vairavimoStazas': 20, 'busena': 'Blaivus', 'detaliBusena': None, 'papBusena': 'Neapsvaigęs', 'detaliPapBusena': None, 'kaltininkas': 'Taip', 'girtumasPromilemis': None, 'dalyvioBusena': 'KALTININKAS', 'teisenosStadijaBusena': 'ADMINISTRACINIS', 'dalyvioKetPazeidimai': ['Saugaus greičio nepasirinkimas'], 'tpId': 32955417}, {'dalyvisId': 47838044, 'kategorija': 'Keleivis', 'dalyvioTipas': 'Kitas asmuo', 'pasisalino': 'Ne', 'asmuoNezinomas': 'Ne', 'lytis': 'Moteris', 'pilietybe': None, 'amzius': 43, 'neigalumas': 'Ne', 'saugosDirzas': 'Neįrengtas', 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': 'Neįrengta', 'bukle': 'Sužeistas', 'detaliBukle': 'Gydomas stacionare', 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': None, 'vairavimoStazas': None, 'busena': 'Blaivus', 'detaliBusena': None, 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Ne', 'girtumasPromilemis': None, 'dalyvioBusena': 'NEPAZEIDE', 'teisenosStadijaBusena': None, 'dalyvioKetPazeidimai': [], 'tpId': 32955417}],

[{'dalyvisId': 47790937, 'kategorija': 'Automobilio vairuotojas', 'dalyvioTipas': 'Kitas asmuo', 'pasisalino': 'Ne', 'asmuoNezinomas': 'Ne', 'lytis': 'Vyras', 'pilietybe': None, 'amzius': 23, 'neigalumas': 'Ne', 'saugosDirzas': 'Užsegtas', 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': 'Nesuveikė', 'bukle': 'Nenukentėjo', 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': 'Atitinka', 'vairavimoStazas': 4, 'busena': 'Neblaivus', 'detaliBusena': 'Lengvas neblaivumo (girtumo) laipsnis', 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Taip', 'girtumasPromilemis': 0.87, 'dalyvioBusena': 'KALTININKAS', 'teisenosStadijaBusena': 'ADMINISTRACINIS', 'dalyvioKetPazeidimai': ['Saugaus greičio nepasirinkimas'], 'tpId': 32930281}, {'dalyvisId': 47790938, 'kategorija': 'Keleivis', 'dalyvioTipas': 'Kitas asmuo', 'pasisalino': 'Ne', 'asmuoNezinomas': 'Ne', 'lytis': 'Vyras', 'pilietybe': None, 'amzius': 23, 'neigalumas': 'Ne', 'saugosDirzas': 'Užsegtas', 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': 'Nesuveikė', 'bukle': 'Nenukentėjo', 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': None, 'vairavimoStazas': None, 'busena': 'Neblaivus', 'detaliBusena': 'Lengvas neblaivumo (girtumo) laipsnis', 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Ne', 'girtumasPromilemis': 1.46, 'dalyvioBusena': 'NEPAZEIDE', 'teisenosStadijaBusena': None, 'dalyvioKetPazeidimai': [], 'tpId': 32930281}],

[{'dalyvisId': 47756705, 'kategorija': 'Automobilio vairuotojas', 'dalyvioTipas': 'Kitas asmuo', 'pasisalino': 'Ne', 'asmuoNezinomas': 'Ne', 'lytis': 'Vyras', 'pilietybe': None, 'amzius': 44, 'neigalumas': 'Ne', 'saugosDirzas': 'Užsegtas', 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': 'Suveikė', 'bukle': 'Nenukentėjo', 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': 'Atitinka', 'vairavimoStazas': 27, 'busena': 'Blaivus', 'detaliBusena': None, 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Taip', 'girtumasPromilemis': None, 'dalyvioBusena': 'KALTININKAS', 'teisenosStadijaBusena': 'ADMINISTRACINIS', 'dalyvioKetPazeidimai': ['Saugaus greičio nepasirinkimas'], 'tpId': 32912295}],

[{'dalyvisId': 48295244, 'kategorija': 'Pėsčiasis', 'dalyvioTipas': 'Kitas asmuo', 'pasisalino': 'Ne', 'asmuoNezinomas': 'Ne', 'lytis': 'Moteris', 'pilietybe': None, 'amzius': 72, 'neigalumas': 'Ne', 'saugosDirzas': None, 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': None, 'bukle': 'Sužeistas', 'detaliBukle': 'Gydomas ambulatoriškai', 'suzalojimoMastas': 'Nesunkus sveikatos sutrikdymas', 'vairuotojoKvalifikacija': None, 'vairavimoStazas': None, 'busena': 'Blaivus', 'detaliBusena': None, 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Ne', 'girtumasPromilemis': None, 'dalyvioBusena': 'NEPAZEIDE', 'teisenosStadijaBusena': None, 'dalyvioKetPazeidimai': [], 'tpId': None}, {'dalyvisId': 48295243, 'kategorija': 'Automobilio vairuotojas', 'dalyvioTipas': 'Kitas asmuo', 'pasisalino': 'Taip', 'asmuoNezinomas': 'Ne', 'lytis': 'Vyras', 'pilietybe': None, 'amzius': 31, 'neigalumas': 'Ne', 'saugosDirzas': 'Užsegtas', 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': 'Nesuveikė', 'bukle': 'Nenukentėjo', 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': 'Neturi teisės vairuoti', 'vairavimoStazas': None, 'busena': 'Blaivus', 'detaliBusena': None, 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Taip', 'girtumasPromilemis': None, 'dalyvioBusena': 'GALIMAS\_KALTININKAS', 'teisenosStadijaBusena': 'IKITEISMINIS', 'dalyvioKetPazeidimai': ['Saugaus atstumo nesilaikymas'], 'tpId': 33242282}],

[{'dalyvisId': 47756688, 'kategorija': 'Automobilio vairuotojas', 'dalyvioTipas': 'Kitas asmuo', 'pasisalino': 'Ne', 'asmuoNezinomas': 'Ne', 'lytis': None, 'pilietybe': None, 'amzius': None, 'neigalumas': 'Ne', 'saugosDirzas': 'Užsegtas', 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': 'Nesuveikė', 'bukle': 'Nenukentėjo', 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': 'Atitinka', 'vairavimoStazas': 5, 'busena': 'Blaivus', 'detaliBusena': None, 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Ne', 'girtumasPromilemis': None, 'dalyvioBusena': 'NEPAZEIDE', 'teisenosStadijaBusena': None, 'dalyvioKetPazeidimai': [], 'tpId': 32912286}, {'dalyvisId': 47756689, 'kategorija': 'Automobilio vairuotojas', 'dalyvioTipas': 'Kitas asmuo', 'pasisalino': 'Ne', 'asmuoNezinomas': 'Ne', 'lytis': None, 'pilietybe': None, 'amzius': None, 'neigalumas': 'Ne', 'saugosDirzas': 'Užsegtas', 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': 'Nesuveikė', 'bukle': 'Nenukentėjo', 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': 'Atitinka', 'vairavimoStazas': 5, 'busena': 'Blaivus', 'detaliBusena': None, 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Taip', 'girtumasPromilemis': None, 'dalyvioBusena': 'KALTININKAS', 'teisenosStadijaBusena': 'ADMINISTRACINIS', 'dalyvioKetPazeidimai': ['Saugaus greičio nepasirinkimas'], 'tpId': 32912287}],

[{'dalyvisId': 47756709, 'kategorija': 'Automobilio vairuotojas', 'dalyvioTipas': 'Kitas asmuo', 'pasisalino': 'Ne', 'asmuoNezinomas': 'Ne', 'lytis': 'Vyras', 'pilietybe': None, 'amzius': 39, 'neigalumas': 'Ne', 'saugosDirzas': 'Užsegtas', 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': 'Nesuveikė', 'bukle': 'Nenukentėjo', 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': 'Atitinka', 'vairavimoStazas': 20, 'busena': 'Blaivus', 'detaliBusena': None, 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Ne', 'girtumasPromilemis': None, 'dalyvioBusena': 'NEPAZEIDE', 'teisenosStadijaBusena': None, 'dalyvioKetPazeidimai': [], 'tpId': 32912297}],

[{'dalyvisId': 47823829, 'kategorija': 'Automobilio vairuotojas', 'dalyvioTipas': None, 'pasisalino': 'Taip', 'asmuoNezinomas': 'Taip', 'lytis': None, 'pilietybe': None, 'amzius': None, 'neigalumas': None, 'saugosDirzas': None, 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': None, 'bukle': None, 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': None, 'vairavimoStazas': None, 'busena': None, 'detaliBusena': None, 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Taip', 'girtumasPromilemis': None, 'dalyvioBusena': 'GALIMAS\_KALTININKAS', 'teisenosStadijaBusena': 'ADMINISTRACINIS', 'dalyvioKetPazeidimai': ['Kiti Kelių eismo taisyklių pažeidimai'], 'tpId': 32947795}, {'dalyvisId': 47823839, 'kategorija': 'Kiti', 'dalyvioTipas': 'Kitas asmuo', 'pasisalino': 'Ne', 'asmuoNezinomas': 'Ne', 'lytis': 'Moteris', 'pilietybe': None, 'amzius': 21, 'neigalumas': 'Ne', 'saugosDirzas': None, 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': None, 'bukle': 'Nenukentėjo', 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': None, 'vairavimoStazas': None, 'busena': 'Blaivus', 'detaliBusena': None, 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Taip', 'girtumasPromilemis': None, 'dalyvioBusena': 'KALTININKAS', 'teisenosStadijaBusena': 'ADMINISTRACINIS', 'dalyvioKetPazeidimai': ['Kiti Kelių eismo taisyklių pažeidimai'], 'tpId': None}],

[{'dalyvisId': 47767276, 'kategorija': 'Automobilio vairuotojas', 'dalyvioTipas': 'Kitas asmuo', 'pasisalino': 'Ne', 'asmuoNezinomas': 'Ne', 'lytis': 'Vyras', 'pilietybe': None, 'amzius': 26, 'neigalumas': 'Ne', 'saugosDirzas': 'Užsegtas', 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': 'Suveikė', 'bukle': 'Nenukentėjo', 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': 'Neturi teisės vairuoti', 'vairavimoStazas': None, 'busena': 'Neblaivus', 'detaliBusena': 'Vidutinis neblaivumo (girtumo) laipsnis', 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Taip', 'girtumasPromilemis': 1.77, 'dalyvioBusena': 'KALTININKAS', 'teisenosStadijaBusena': 'ADMINISTRACINIS', 'dalyvioKetPazeidimai': ['Reikalavimo duoti kelią nevykdymas važiuojant per sankryžą'], 'tpId': 32917748}, {'dalyvisId': 47767277, 'kategorija': 'Automobilio vairuotojas', 'dalyvioTipas': 'Kitas asmuo', 'pasisalino': 'Ne', 'asmuoNezinomas': 'Ne', 'lytis': 'Vyras', 'pilietybe': None, 'amzius': 21, 'neigalumas': 'Ne', 'saugosDirzas': 'Užsegtas', 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': 'Suveikė', 'bukle': 'Nenukentėjo', 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': 'Atitinka', 'vairavimoStazas': 3, 'busena': 'Blaivus', 'detaliBusena': None, 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Ne', 'girtumasPromilemis': None, 'dalyvioBusena': 'NEPAZEIDE', 'teisenosStadijaBusena': None, 'dalyvioKetPazeidimai': [], 'tpId': 32917749}, {'dalyvisId': 47767278, 'kategorija': 'Keleivis', 'dalyvioTipas': 'Kitas asmuo', 'pasisalino': 'Ne', 'asmuoNezinomas': 'Ne', 'lytis': 'Moteris', 'pilietybe': None, 'amzius': 27, 'neigalumas': 'Ne', 'saugosDirzas': 'Užsegtas', 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': 'Suveikė', 'bukle': 'Nenukentėjo', 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': None, 'vairavimoStazas': None, 'busena': 'Neblaivus', 'detaliBusena': 'Lengvas neblaivumo (girtumo) laipsnis', 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Ne', 'girtumasPromilemis': 0.9500000000000001, 'dalyvioBusena': 'NEPAZEIDE', 'teisenosStadijaBusena': None, 'dalyvioKetPazeidimai': [], 'tpId': 32917748}],

[{'dalyvisId': 47769521, 'kategorija': 'Automobilio vairuotojas', 'dalyvioTipas': 'Kitas asmuo', 'pasisalino': 'Taip', 'asmuoNezinomas': 'Ne', 'lytis': 'Vyras', 'pilietybe': None, 'amzius': 60, 'neigalumas': 'Ne', 'saugosDirzas': 'Užsegtas', 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': 'Nesuveikė', 'bukle': 'Nenukentėjo', 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': 'Atitinka', 'vairavimoStazas': 27, 'busena': 'Blaivus', 'detaliBusena': None, 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Taip', 'girtumasPromilemis': None, 'dalyvioBusena': 'KALTININKAS', 'teisenosStadijaBusena': 'ADMINISTRACINIS', 'dalyvioKetPazeidimai': ['Kiti Kelių eismo taisyklių pažeidimai'], 'tpId': 32918919}, {'dalyvisId': 47769522, 'kategorija': 'Automobilio vairuotojas', 'dalyvioTipas': 'Kitas asmuo', 'pasisalino': 'Ne', 'asmuoNezinomas': 'Ne', 'lytis': 'Moteris', 'pilietybe': None, 'amzius': 33, 'neigalumas': 'Ne', 'saugosDirzas': 'Užsegtas', 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': 'Nesuveikė', 'bukle': 'Nenukentėjo', 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': 'Atitinka', 'vairavimoStazas': 15, 'busena': 'Blaivus', 'detaliBusena': None, 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Ne', 'girtumasPromilemis': None, 'dalyvioBusena': 'NEPAZEIDE', 'teisenosStadijaBusena': None, 'dalyvioKetPazeidimai': [], 'tpId': 32918920}],

[{'dalyvisId': 47756831, 'kategorija': 'Automobilio vairuotojas', 'dalyvioTipas': 'Kitas asmuo', 'pasisalino': 'Ne', 'asmuoNezinomas': 'Ne', 'lytis': 'Vyras', 'pilietybe': None, 'amzius': 36, 'neigalumas': 'Ne', 'saugosDirzas': 'Užsegtas', 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': 'Nesuveikė', 'bukle': 'Nenukentėjo', 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': 'Atitinka', 'vairavimoStazas': 17, 'busena': 'Blaivus', 'detaliBusena': None, 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Ne', 'girtumasPromilemis': None, 'dalyvioBusena': 'NEPAZEIDE', 'teisenosStadijaBusena': None, 'dalyvioKetPazeidimai': [], 'tpId': 32912354}, {'dalyvisId': 47756830, 'kategorija': None, 'dalyvioTipas': None, 'pasisalino': 'Taip', 'asmuoNezinomas': 'Taip', 'lytis': None, 'pilietybe': None, 'amzius': None, 'neigalumas': None, 'saugosDirzas': None, 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': None, 'bukle': None, 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': None, 'vairavimoStazas': None, 'busena': None, 'detaliBusena': None, 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Taip', 'girtumasPromilemis': None, 'dalyvioBusena': 'GALIMAS\_KALTININKAS', 'teisenosStadijaBusena': None, 'dalyvioKetPazeidimai': ['Saugaus atstumo nesilaikymas'], 'tpId': None}],

[{'dalyvisId': 47816484, 'kategorija': 'Kiti', 'dalyvioTipas': 'Kitas asmuo', 'pasisalino': 'Ne', 'asmuoNezinomas': 'Ne', 'lytis': 'Vyras', 'pilietybe': None, 'amzius': 67, 'neigalumas': 'Ne', 'saugosDirzas': None, 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': None, 'bukle': 'Nenukentėjo', 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': None, 'vairavimoStazas': None, 'busena': 'Blaivus', 'detaliBusena': None, 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Ne', 'girtumasPromilemis': None, 'dalyvioBusena': 'NEPAZEIDE', 'teisenosStadijaBusena': None, 'dalyvioKetPazeidimai': [], 'tpId': None}, {'dalyvisId': 47816483, 'kategorija': 'Automobilio vairuotojas', 'dalyvioTipas': 'Kitas asmuo', 'pasisalino': 'Ne', 'asmuoNezinomas': 'Ne', 'lytis': 'Vyras', 'pilietybe': None, 'amzius': 58, 'neigalumas': 'Ne', 'saugosDirzas': 'Užsegtas', 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': 'Nesuveikė', 'bukle': 'Nenukentėjo', 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': 'Atitinka', 'vairavimoStazas': 17, 'busena': 'Neblaivus', 'detaliBusena': None, 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Taip', 'girtumasPromilemis': None, 'dalyvioBusena': 'GALIMAS\_KALTININKAS', 'teisenosStadijaBusena': 'IKITEISMINIS', 'dalyvioKetPazeidimai': ['Saugaus atstumo nesilaikymas'], 'tpId': 32943941}]

]

Sprendimas su

import json

from statistics import mean

# Funkcija skaičiavimui

def analyze\_data(data):

    males = 0

    females = 0

    ages = []

    offenders\_who\_fled = 0

    for entry in data:

        for participant in entry:

            # Patikriname dalyvio lytį

            if participant.get('lytis') == 'Vyras':

                males += 1

            elif participant.get('lytis') == 'Moteris':

                females += 1

            # Patikriname dalyvio amžių

            age = participant.get('amzius')

            if age is not None:

                ages.append(age)

            # Patikriname ar kaltininkas pasišalino

            if participant.get('dalyvioBusena') == 'KALTININKAS' and participant.get('pasisalino') == 'Taip':

                offenders\_who\_fled += 1

    avg\_age = mean(ages) if ages else 0

    return {

        'males': males,

        'females': females,

        'avg\_age': avg\_age,

        'offenders\_who\_fled': offenders\_who\_fled

    }

# Pvz. duomenų imtis

data = [

    [{'dalyvisId': 47756477, 'kategorija': 'Automobilio vairuotojas', 'dalyvioTipas': 'Kitas asmuo', 'pasisalino': 'Ne', 'asmuoNezinomas': 'Ne', 'lytis': 'Moteris', 'pilietybe': None, 'amzius': 19, 'neigalumas': 'Ne', 'saugosDirzas': 'Užsegtas', 'vaikoKedesNaudojimas': None, 'saugosSalmoNaudojimas': None, 'oroPagalves': 'Nesuveikė', 'bukle': 'Nenukentėjo', 'detaliBukle': None, 'suzalojimoMastas': None, 'vairuotojoKvalifikacija': 'Atitinka', 'vairavimoStazas': 0, 'busena': 'Blaivus', 'detaliBusena': None, 'papBusena': None, 'detaliPapBusena': None, 'kaltininkas': 'Taip', 'girtumasPromilemis': None, 'dalyvioBusena': 'KALTININKAS', 'teisenosStadijaBusena': 'ADMINISTRACINIS', 'dalyvioKetPazeidimai': ['Kiti Kelių eismo taisyklių pažeidimai'], 'tpId': 1}],

    # Įtraukite daugiau duomenų čia

]

# Skaičiuojame ir spausdiname rezultatus

results = analyze\_data(data)

print(f"Vyrų skaičius: {results['males']}")

print(f"Moterų skaičius: {results['females']}")

print(f"Vidutinis amžius: {results['avg\_age']:.2f}")

print(f"Pasišalinusių kaltininkų skaičius: {results['offenders\_who\_fled']}")