**13. praktiskais darbs. 2. semestris**

**1. uzdevums**

Sastādīt programmu, kas pārbauda vai Sudoku spēles lapiņa ir aizpildīta korekti. (2 punkti)

**Kods:**

# Programmas nosaukums: Sudoku spēles lapiņa korektas aizpildīšanas pārbaude

# 1. uzdevums (1MPR13\_Vladislavs\_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas pārbauda vai Sudoku spēles lapiņa ir aizpildīta korekti. (2 punkti)

# Programmas autors: Vladislavs Babaņins

# Versija 1.0

import numpy

def fill\_array\_randomly(n, m):

# Nejauši aizpilda n x m matricu ar naturāliem skaitļiem no 1 līdz 9.

# [1, 2, 3, 4, 5, 6, 7, 8, 9].

# n - rindu skaits matricā (sudoku ir 9 rindas) (int).

# m - kolonnu skaits matricā (sudoku ir 9 kolonnas) (int).

# Atgriež aizpildīto masīvu ar nejaušiem naturāliem skaitļiem no 1 līdz 9.

# Izveidojam tukšu 9x9 divdimensijas masīvu (matricu).

a = numpy.zeros((n, m), dtype=int)

# Ejam ciklā cauri katrai masīva rindai un kolonnai.

for i in range(n):

for j in range(m):

# Ģenerējam nejaušu naturālu skaitli no 1 līdz 9.

random\_number = numpy.random.randint(1, 10)

# Piešķiram [i][j] vietā nejaušu naturālu skaitli no 1 līdz 9.

a[i][j] = random\_number

# Atgriež aizpildīto masīvu ar nejaušiem naturāliem skaitļiem no 1 līdz 9.

return a

def check\_array\_rows\_and\_columns(a):

# Pārbauda vai matricā katra rindā un kolonnā visi skaitļi ir dažādi, izmantojot kopas.

# Atgriež True, ja visi skaitļi visas rindas un kolonnas ir dažādi.

# Atgriež False, ja ir kādi divi skaitļi kāda rinda vai kolonna kuri sakrīt.

# a - divdimensijas masīvs (matrica).

for i in range(9):

# Pārbauda, vai katrā rindā nav skaitļu dublikātu (nav vienādu skaitļu).

# Izmantojam kopas. Ja kopā nav ar gārumu 9, tad ir cipari kas atkartojas.

if len(set(a[i])) != 9:

print(str(i + 1) + ". rindā ir cipari, kas atkārtojas.") # Izvadam, kur tika atrāsta kļūda.

return False

for j in range(9):

# Pārbauda, vai katrā kolonnā nav skaitļu dublikātu (nav vienādu skaitļu).

# Izmantojam kopas. Ja kopā nav ar gārumu 9, tad ir cipari kas atkartojas.

col\_numbers = [a[i][j] for i in range(9)]

if len(set(col\_numbers)) != 9:

print(str(j + 1) + ". kolonnā ir cipari, kas atkārtojas.")

return False

return True

def check\_submatrix(matrix, i, j):

# Atgriež True ja 3x3 apakšmatricā (apakšmatricas ir paradītas zemāk) visi skaitļi ir dažādi.

# Atgriež False ja 3x3 apakšmatricā kādi divi skaitļi ir vienādi.

# Sudoku 3x3 apakšmatricas.

# matrix - pilnā 9x9 matrica (divdimensijas masīvs).

# i - no kuras rīndas sāksim (int).

# j - no kuras kolonnas sāksim (int).

submatrix = []

for row in range(i, i + 3):

for col in range(j, j + 3):

submatrix.append(matrix[row][col])

return len(set(submatrix)) == 9

def check\_3x3\_submatrixes(a):

# Pārbaudam katru 3x3 apakšmatricu un paziņojam lietotājam kāda apakšmatrica skaitļi ir dažādi un kurā ir vienādi.

# Izsauc check\_submatrix(a, i, j) un paziņo lietotājam "Visi skaitļi apakšmatricā [{i//3 + 1}][{j//3 + 1}] ir atšķirīgi.",

# vai "Ne visi skaitļi apakšmatricā [{i//3 + 1}][{j//3 + 1}] ir atšķirīgi."

# Atgriež True, ja nav nevienas apakšmatricas, kurai iekšā ir divi vienādi skaitļi.

# Atgriež False, ja ir kaut viena apakšmatrica, kurai iekšā ir divi vienādi skaitļi.

# a - divdimensiju masīvs.

# Sudoku deviņas apakšmatricas:

# [0][0] [0][1] [0][2] [0][3] [0][4] [0][5] [0][6] [0][7] [0][8]

# [1][0] [1][1] [1][2] [1][3] [1][4] [1][5] [1][6] [1][7] [1][8]

# [2][0] [2][1] [2][2] [2][3] [2][4] [2][5] [2][6] [2][7] [2][8]

# [3][0] [3][1] [3][2] [3][3] [3][4] [3][5] [3][6] [3][7] [3][8]

# [4][0] [4][1] [4][2] [4][3] [4][4] [4][5] [4][6] [4][7] [4][8]

# [5][0] [5][1] [5][2] [5][3] [5][4] [5][5] [5][6] [5][7] [5][8]

# [6][0] [6][1] [6][2] [6][3] [6][4] [6][5] [6][6] [6][7] [6][8]

# [7][0] [7][1] [7][2] [7][3] [7][4] [7][5] [7][6] [7][7] [7][8]

# [8][0] [8][1] [8][2] [8][3] [8][4] [8][5] [8][6] [8][7] [8][8]

for i in range(0, 9, 3):

for j in range(0, 9, 3):

if check\_submatrix(a, i, j):

print(f"Visi skaitļi apakšmatricā [{i//3 + 1}][{j//3 + 1}] ir atšķirīgi.")

else:

print(f"Kļūda! Apakšmatricā [{i//3 + 1}][{j//3 + 1}] ir divi vienādi skaitļi!")

return False

return True

def input\_sudoku\_matrix():

# Paprasa lietotājam ievādīt veselu skaitļi no 1 līdz 9 katrai "šūnas" vērtībai 9x9 matricai.

# Atgriež aizpildītu ar lietotāja ievādītam vērtībam matricu ar veseliem skaitļiem no 1 līdz 9.

# Ejam ciklā caur katru masīva rindu un kolonnu.

for i in range(9):

for j in range(9):

# Turpinam palūgt ievadi, līdz tiek norādīts derīgs skaitlis (cipars no 1 līdz 9).

while True:

# Palūdzam lietotājam ievadīt pašreizējās pozīcijas skaitļi (cipars no 1 līdz 9).

number = input("Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [" + str(i) + "][" + str(j) + "]: ")

# Pārbaudam, vai ievadītais skaitlis ir no 1 līdz 9 un vai vispār tas ir cipars.

if number.isdigit() and 1 <= int(number) <= 9:

# Konvertējam ievadi par veselu skaitli un piešķiram to masīvam.

a[i][j] = int(number)

break

else:

print("Nepareiza ievade! Lūdzu, ievadiet veselu skaitli no 1 līdz 9.")

# Agriež aizpildītu ar lietotāja ievādītam vērtībam matricu ar veseliem skaitļiem no 1 līdz 9.

return a

def matrix\_to\_string\_float\_3x3(matrix):

# Atgriež matricas virknes attēlojumu, kur katra rinda ir atdalīta ar \n un izlīdzināta atbilstoši maksimālās vērtības garumam.

# Ja vērtība ir vesels skaitlis, tā tiek parādīta bez komata. Pretējā gadījumā tas tiek parādīts ar decimālzīmi.

# Funkcija arī atrod maksimālo vērtību garumu matricā un aizpilda nepieciešamās atstarpes " ", lai tās glīti izlīdzinātu (glītas atkāpes).

# Ja matricā ir 0, tas tiek aizstāts ar simbolu ∙.

# matrix - matrica (divdimensiju numpy masīvs ar izmēriem n x m).

# Piemērs, kāda veida tiek atgriezta simbolu virkne:

# 1 6 3 9 3 4 3 6 6

# 4 9 9 2 7 3 9 9 5

# 3 5 9 5 2 7 9 7 4

#

# 4 6 6 3 3 8 2 5 3

# 1 5 6 8 9 2 4 8 3

# 9 3 9 6 8 7 2 8 2

#

# 7 4 9 3 9 3 7 1 1

# 1 3 5 2 6 3 1 3 1

# 6 5 3 8 9 7 7 1 8

rindas = len(matrix)

kolonnas = len(matrix[0])

max\_len = 0

for i in range(rindas): # atrod max\_len, lai noteiktu nepieciešamo atkāpi.

for j in range(kolonnas):

number = matrix[i][j]

if number == int(number):

value\_len = len(str(int(number)))

else:

value\_len = len(str(float(number)))

if value\_len > max\_len:

max\_len = value\_len

# Izveido matricas virknes attēlojumu, kur katra rinda tiek atdalīta ar \n un izlīdzināta atbilstoši maksimālās vērtības garumam

sv = ""

for i in range(rindas):

for j in range(kolonnas):

number = matrix[i][j]

if number == 0:

number = '∙'

elif number == int(number):

number = int(number)

else:

number = str(float(number))

atkape = " " \* (max\_len - len(str(number)))

sv += atkape + str(number)

if j < kolonnas - 1 and (j + 1) % 3 != 0:

sv = sv + " "

elif j < kolonnas - 1 and (j + 1) % 3 == 0:

sv = sv + " "

sv = sv + "\n"

if (i + 1) % 3 == 0 and i < rindas - 1:

sv += "\n"

return sv

# ---------------------------------------------------------

# Galvenā programmas daļa

# ---------------------------------------------------------

a = numpy.zeros((9, 9))

choose = input("Vai gribat nejauši aizpildīt Sudoku spēlēs lapiņu vai manuāli? (n/m) ==> ")

while choose != "n" and choose != "m":

choose = input("Kļūda! Ievadīet \"n\" vai \"m\"! Vai gribāt nejauši aizpildīt Sudoku kartīti vai manuāli? (n/m) ==> ")

if choose == "n":

a = fill\_array\_randomly(9, 9)

print()

print(matrix\_to\_string\_float(a))

elif choose == "m":

b = input\_sudoku\_matrix()

print()

print(matrix\_to\_string\_float(b))

if check\_array\_rows\_and\_columns(a) and check\_3x3\_submatrixes(a):

print("Ir korekti aizpildīta Sudoku spēlēs lapiņa.") # VALID

else:

print("Nav korekti aizpildīta Sudoku spēlēs lapiņa.") # INVALID

# TESTĒŠANAI

'''

for i in range(1000):

a = fill\_array\_randomly(9, 9)

# Izdrukām masīvu

print()

print(matrix\_to\_string\_float(a))

# if check\_array\_rows\_and\_columns(a) and check\_3x3\_submatrixes(a):

if check\_3x3\_submatrixes(a):

print("VALID")

else:

print("INVALID")

'''

# TESTĒŠANAI

'''

for i in range(1000):

a = fill\_array\_randomly(9, 9)

# Izdrukām masīvu

print()

print(matrix\_to\_string\_float(a))

if check\_array\_rows\_and\_columns(a):

print("VALID")

else:

print("INVALID")

'''

**Testa piemēri:**

1)

A screen shot of a computer

Description automatically generated with medium confidence

2)

A screen shot of a computer

Description automatically generated with medium confidence

3)

Vai gribat nejauši aizpildīt Sudoku spēlēs lapiņu vai manuāli? (n/m) ==> m

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][0]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][1]: 1

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][2]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][3]: 5

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][4]: 4

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][5]: 3

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][6]: 6

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][7]: 7

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][8]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][0]: 5

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][1]: 4

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][2]: 3

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][3]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][4]: 7

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][5]: 6

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][6]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][7]: 1

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][8]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][0]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][1]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][2]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][3]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][4]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][5]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][6]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][7]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][8]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][0]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][1]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][2]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][3]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][4]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][5]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][6]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][7]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][8]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][0]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][1]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][2]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][3]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][4]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][5]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][6]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][7]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][8]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][0]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][1]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][2]:

Nepareiza ievade! Lūdzu, ievadiet veselu skaitli no 1 līdz 9.

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][2]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][3]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][4]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][5]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][6]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][7]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][8]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][0]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][1]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][2]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][3]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][4]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][5]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][6]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][7]: 22

Nepareiza ievade! Lūdzu, ievadiet veselu skaitli no 1 līdz 9.

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][7]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][8]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][0]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][1]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][2]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][3]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][4]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][5]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][6]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][7]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][8]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][0]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][1]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][2]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][3]: 22

Nepareiza ievade! Lūdzu, ievadiet veselu skaitli no 1 līdz 9.

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][3]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][4]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][5]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][6]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][7]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][8]: 2

2 1 9 5 4 3 6 7 8

5 4 3 8 7 6 9 1 2

2 2 2 2 2 2 2 2 2

2 2 2 2 2 2 2 2 2

2 2 2 2 2 2 2 2 2

2 2 2 2 2 2 2 2 2

2 2 2 2 2 2 2 2 2

2 2 2 2 2 2 2 2 2

2 2 2 2 2 2 2 2 2

3. rindā ir cipari, kas atkārtojas.

Nav korekti aizpildīta Sudoku spēlēs lapiņa.

A screenshot of a computer screen

Description automatically generated with medium confidence

4)

Vai gribāt nejauši aizpildīt Sudoku spēlēs lapiņu vai manuāli? (n/m) ==> m

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][0]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][1]: 1

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][2]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][3]: 5

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][4]: 4

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][5]: 3

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][6]: 6

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][7]: 7

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][8]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][0]: 5

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][1]: 4

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][2]: 3

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][3]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][4]: 7

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][5]: 6

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][6]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][7]: 1

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][8]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][0]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][1]: 7

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][2]: 6

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][3]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][4]: 1

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][5]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][6]: 3

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][7]: 4

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][8]: 5

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][0]: 4

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][1]: 3

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][2]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][3]: 7

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][4]: 6

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][5]: 5

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][6]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][7]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][8]: 1

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][0]: 7

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][1]: 6

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][2]: 5

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][3]: 1

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][4]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][5]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][6]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][7]: 3

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][8]: 4

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][0]: 1

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][1]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][2]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][3]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][4]: 3

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][5]: 4

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][6]: 1

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][7]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][8]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][0]: 4

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][1]: 3

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][2]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][3]: 5

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][4]: 6

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][5]: 7

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][6]: 3

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][7]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][8]: 1

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][0]: 6

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][1]: 5

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][2]: 4

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][3]: 7

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][4]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][5]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][6]: 6

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][7]: 5

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][8]: 4

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][0]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][1]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][2]: 7

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][3]: 1

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][4]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][5]: 3

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][6]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][7]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][8]: 7

2 1 9 5 4 3 6 7 8

5 4 3 8 7 6 9 1 2

8 7 6 2 1 9 3 4 5

4 3 2 7 6 5 8 9 1

7 6 5 1 9 8 2 3 4

1 9 8 2 3 4 1 9 8

4 3 2 5 6 7 3 2 1

6 5 4 7 8 9 6 5 4

9 8 7 1 2 3 9 8 7

6. rindā ir cipari, kas atkārtojas.

Nav korekti aizpildīta Sudoku spēlēs lapiņa.

A screenshot of a computer screen

Description automatically generated with medium confidence

5)

Vai gribat nejauši aizpildīt Sudoku spēlēs lapiņu vai manuāli? (n/m) ==> m

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][0]: 10

Nepareiza ievade! Lūdzu, ievadiet veselu skaitli no 1 līdz 9.

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][0]: 0

Nepareiza ievade! Lūdzu, ievadiet veselu skaitli no 1 līdz 9.

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][0]: 12.2

Nepareiza ievade! Lūdzu, ievadiet veselu skaitli no 1 līdz 9.

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][0]: 5.5

Nepareiza ievade! Lūdzu, ievadiet veselu skaitli no 1 līdz 9.

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][0]: labi

Nepareiza ievade! Lūdzu, ievadiet veselu skaitli no 1 līdz 9.

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][0]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][1]: 1

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][2]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][3]: 5

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][4]: 4

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][5]: 3

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][6]: 6

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][7]: 7

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][8]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][0]: 5

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][1]: 4

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][2]: 3

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][3]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][4]: 7

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][5]: 6

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][6]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][7]: 1

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][8]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][0]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][1]: 7

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][2]: 6

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][3]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][4]: 1

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][5]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][6]: 3

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][7]: 4

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][8]: 5

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][0]: 4

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][1]: 3

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][2]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][3]: 7

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][4]: 6

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][5]: 5

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][6]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][7]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][8]: 1

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][0]: 7

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][1]: 6

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][2]: 5

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][3]: 1

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][4]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][5]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][6]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][7]: 3

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][8]: 4

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][0]: 1

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][1]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][2]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][3]: 4

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][4]: 3

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][5]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][6]: 5

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][7]: 6

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][8]: 7

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][0]: 3

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][1]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][2]: 1

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][3]: 6

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][4]: 5

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][5]: 4

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][6]: 7

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][7]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][8]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][0]: 6

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][1]: 5

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][2]: 4

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][3]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][4]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][5]: 7

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][6]: 1

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][7]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][8]: 3

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][0]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][1]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][2]: 7

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][3]: 3

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][4]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][5]: 1

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][6]: 4

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][7]: 5

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][8]: 6

2 1 9 5 4 3 6 7 8

5 4 3 8 7 6 9 1 2

8 7 6 2 1 9 3 4 5

4 3 2 7 6 5 8 9 1

7 6 5 1 9 8 2 3 4

1 9 8 4 3 2 5 6 7

3 2 1 6 5 4 7 8 9

6 5 4 9 8 7 1 2 3

9 8 7 3 2 1 4 5 6

Visi skaitļi apakšmatricā [1][1] ir atšķirīgi.

Visi skaitļi apakšmatricā [1][2] ir atšķirīgi.

Visi skaitļi apakšmatricā [1][3] ir atšķirīgi.

Visi skaitļi apakšmatricā [2][1] ir atšķirīgi.

Visi skaitļi apakšmatricā [2][2] ir atšķirīgi.

Visi skaitļi apakšmatricā [2][3] ir atšķirīgi.

Visi skaitļi apakšmatricā [3][1] ir atšķirīgi.

Visi skaitļi apakšmatricā [3][2] ir atšķirīgi.

Visi skaitļi apakšmatricā [3][3] ir atšķirīgi.

Ir korekti aizpildīta Sudoku spēlēs lapiņa.

A screenshot of a computer screen

Description automatically generated with medium confidence

**PU1. uzdevums 1.līmenis.**

Sastādīt programmu, kas uzģenerē korekti aizpildītu Sudoku spēles lapiņu, ja ir zināms cik skaitļu ir atvērti.

1.līmenis - lapiņa sākotnēji uzģenerēta (aizpildīta) korekti ar visiem skaitļiem un tad izvēlas atklātos skaitļus un izdzēš pārējos.

2.līmenis - uzģenerē atvērtos skaitļus (visus uzreiz vai secīgi pa vienam) un pēc tam pārbauda, vai šis komplekts ir korekti atrisināms, bet ja nē, tad atkārto procesu, kamēr iegūst korekti aizpildāmu komplektu.

**Kods:**

# Programmas nosaukums: Sudoku spēlē ģenerācija (1.LĪMENIS)

# PU1. (1MPR13\_Vladislavs\_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas uzģenerē korekti aizpildītu Sudoku spēles lapiņu, ja ir zināms cik skaitļu ir atvērti.

# 1.līmenis - lapiņa sākotnēji uzģenerēta (aizpildīta) korekti ar visiem skaitļiem un tad izvēlas atklātos skaitļus un izdzēš pārējos.

# 2.līmenis - uzģenerē atvērtos skaitļus (visus uzreiz vai secīgi pa vienam) un pēc tam pārbauda, vai šis komplekts ir korekti atrisināms, bet ja nē, tad atkārto procesu,

# kamēr iegūst korekti aizpildāmu komplektu.

# Programmas autors: Vladislavs Babaņins

# Versija 1.0

"""

1.LĪMENIS TIKA IZPILDĪTS AR TĀ SAUCĀMO "BACKTRACKING" ALGORITMU.

"""

import random

def check\_array\_rows\_and\_columns(a):

# Pārbauda vai matricā katra rindā un kolonnā visi skaitļi ir dažādi, izmantojot kopas.

# Atgriež True, ja visi skaitļi visas rindas un kolonnas ir dažādi.

# Atgriež False, ja ir kādi divi skaitļi kāda rinda vai kolonna kuri sakrīt.

# a - divdimensijas masīvs (matrica).

for i in range(9):

# Pārbauda, vai katrā rindā nav skaitļu dublikātu (nav vienādu skaitļu).

# Izmantojam kopas. Ja kopā nav ar gārumu 9, tad ir cipari kas atkartojas.

if len(set(a[i])) != 9:

print(str(i + 1) + ". rindā ir cipari, kas atkārtojas.") # Izvadam, kur tika atrāsta kļūda.

return False

for j in range(9):

# Pārbauda, vai katrā kolonnā nav skaitļu dublikātu (nav vienādu skaitļu).

# Izmantojam kopas. Ja kopā nav ar gārumu 9, tad ir cipari kas atkartojas.

col\_numbers = [a[i][j] for i in range(9)]

if len(set(col\_numbers)) != 9:

print(str(j + 1) + ". kolonnā ir cipari, kas atkārtojas.")

return False

return True

def check\_submatrix(matrix, i, j):

# Atgriež True ja 3x3 apakšmatricā (apakšmatricas ir paradītas zemāk) visi skaitļi ir dažādi.

# Atgriež False ja 3x3 apakšmatricā kādi divi skaitļi ir vienādi.

# Sudoku 3x3 apakšmatricas.

# matrix - pilnā 9x9 matrica (divdimensijas masīvs).

# i - no kuras rīndas sāksim (int).

# j - no kuras kolonnas sāksim (int).

submatrix = []

for row in range(i, i + 3):

for col in range(j, j + 3):

submatrix.append(matrix[row][col])

return len(set(submatrix)) == 9

def check\_3x3\_submatrixes(a):

# Pārbaudam katru 3x3 apakšmatricu un paziņojam lietotājam kāda apakšmatrica skaitļi ir dažādi un kurā ir vienādi.

# Izsauc check\_submatrix(a, i, j) un paziņo lietotājam "Visi skaitļi apakšmatricā [{i//3 + 1}][{j//3 + 1}] ir atšķirīgi.",

# vai "Ne visi skaitļi apakšmatricā [{i//3 + 1}][{j//3 + 1}] ir atšķirīgi."

# Atgriež True, ja nav nevienas apakšmatricas, kurai iekšā ir divi vienādi skaitļi.

# Atgriež False, ja ir kaut viena apakšmatrica, kurai iekšā ir divi vienādi skaitļi.

# a - divdimensiju masīvs.

# Sudoku deviņas apakšmatricas:

# [0][0] [0][1] [0][2] [0][3] [0][4] [0][5] [0][6] [0][7] [0][8]

# [1][0] [1][1] [1][2] [1][3] [1][4] [1][5] [1][6] [1][7] [1][8]

# [2][0] [2][1] [2][2] [2][3] [2][4] [2][5] [2][6] [2][7] [2][8]

# [3][0] [3][1] [3][2] [3][3] [3][4] [3][5] [3][6] [3][7] [3][8]

# [4][0] [4][1] [4][2] [4][3] [4][4] [4][5] [4][6] [4][7] [4][8]

# [5][0] [5][1] [5][2] [5][3] [5][4] [5][5] [5][6] [5][7] [5][8]

# [6][0] [6][1] [6][2] [6][3] [6][4] [6][5] [6][6] [6][7] [6][8]

# [7][0] [7][1] [7][2] [7][3] [7][4] [7][5] [7][6] [7][7] [7][8]

# [8][0] [8][1] [8][2] [8][3] [8][4] [8][5] [8][6] [8][7] [8][8]

for i in range(0, 9, 3):

for j in range(0, 9, 3):

if check\_submatrix(a, i, j):

pass

#print(f"Visi skaitļi apakšmatricā [{i//3 + 1}][{j//3 + 1}] ir atšķirīgi.")

else:

#print(f"Kļūda! Apakšmatricā [{i//3 + 1}][{j//3 + 1}] ir divi vienādi skaitļi!")

return False

return True

def generate\_sudoku():

# Izveidojam 9x9 nulles matricu.

matrix = []

for i in range(9):

row = [0] \* 9

matrix.append(row)

# Aizpildam matricu, sākot no augšējā kreisā stūra.

fill\_matrix(matrix, 0, 0)

return matrix

def fill\_matrix(matrix, i, j):

# Aizpildām nulles matricu ar (backtracking algorithm) algoritmu tā, lai visas rindas būtu atšķirīgi skaitli.

if i == 9:

return True

# Aprēķina nākamo "šūnu" indeksus.

next\_i = i + (j + 1) // 9

next\_j = (j + 1) % 9

# Izveidot sarakstu ar veseliem skaitļiem no 1 līdz 9.

list\_random = list(range(1, 10))

# Sajaucam sarakstā vērtības.

random.shuffle(list\_random)

# Mēgīnām ielikt pēc kārtas katru ciparu no nejauši sajaukta saraksta.

for number in list\_random:

# Pārbaudam, vai skaitlis ir derīgs (ar karogu "valid").

valid = True

# Pārbaudam rindas.

for k in range(9):

if matrix[i][k] == number:

valid = False # Ja sakrīt kāds skaitlīs rindā ar izvelēto skaitli, tad karogs ir False, vajag paņemt citu skaitli no list\_random (ņēm pēc kārtas).

break

# Pārbaudam kolonnas.

for k in range(9):

if matrix[k][j] == number:

valid = False # Ja sakrīt kāds skaitlīs kolonna ar izvelēto skaitli, tad karogs ir False, vajag paņemt citu skaitli no list\_random (ņēm pēc kārtas).

break

# Pārbaudam 3x3 noteiktas apakšmatricas.

sub\_i = i // 3 \* 3 # "apakš\_i"

sub\_j = j // 3 \* 3 # "apakš\_j"

for k in range(sub\_i, sub\_i + 3):

for l in range(sub\_j, sub\_j + 3):

if matrix[k][l] == number:

valid = False

break

# Ja skaitlis ir derīgs, aizpildam to "šūnu" un rekursīvi sākam aizpildīt nākamo "šūnu".

if valid:

matrix[i][j] = number

if fill\_matrix(matrix, next\_i, next\_j):

return True

# Ja esam izmēģinājuši visus skaitļus un neviens nedēr, tad jāatkāpjas par vienu soli atpakāļ, un jau citus skaitļus likt. (backtracking algorithm)

matrix[i][j] = 0

return False

def delete\_cells(matrix, delete\_count):

# Atgriež matricu, kur nejauši tiek izvelēti matricas skaitļi delete\_count skaitā un tie tiek pārverti par nullem.

# matrix - divdimensijas masīvs (matrica), kurai gribām pārverst delete\_count skaitļus.

# delete\_count - cik skaitļus gribām "pārverst" par nullem. Jo lielāk, jo ir lielāka Sudoku spēlēs grūtība.

for k in range(delete\_count):

i = random.randint(0, 8)

j = random.randint(0, 8)

matrix[i][j] = 0

return matrix

def matrix\_to\_string\_float\_3x3(matrix):

# Atgriež matricas virknes attēlojumu, kur katra rinda ir atdalīta ar \n un izlīdzināta atbilstoši maksimālās vērtības garumam.

# Ja vērtība ir vesels skaitlis, tā tiek parādīta bez komata. Pretējā gadījumā tas tiek parādīts ar decimālzīmi.

# Funkcija arī atrod maksimālo vērtību garumu matricā un aizpilda nepieciešamās atstarpes " ", lai tās glīti izlīdzinātu (glītas atkāpes).

# Ja matricā ir 0, tas tiek aizstāts ar simbolu ∙.

# matrix - matrica (divdimensiju masīvs ar izmēriem n x m).

# Piemērs, kāda veida tiek atgriezta simbolu virkne:

# 1 6 3 9 3 4 3 6 6

# 4 9 9 2 7 3 9 9 5

# 3 5 9 5 2 7 9 7 4

#

# 4 6 6 3 3 8 2 5 3

# 1 5 6 8 9 2 4 8 3

# 9 3 9 6 8 7 2 8 2

#

# 7 4 9 3 9 3 7 1 1

# 1 3 5 2 6 3 1 3 1

# 6 5 3 8 9 7 7 1 8

rindas = len(matrix)

kolonnas = len(matrix[0])

max\_len = 0

for i in range(rindas): # atrod max\_len, lai noteiktu nepieciešamo atkāpi.

for j in range(kolonnas):

number = matrix[i][j]

if number == int(number):

value\_len = len(str(int(number)))

else:

value\_len = len(str(float(number)))

if value\_len > max\_len:

max\_len = value\_len

# Izveido matricas virknes attēlojumu, kur katra rinda tiek atdalīta ar \n un izlīdzināta atbilstoši maksimālās vērtības garumam

sv = ""

for i in range(rindas):

for j in range(kolonnas):

number = matrix[i][j]

if number == 0:

number = "∙"

elif number == int(number):

number = int(number)

else:

number = str(float(number))

atkape = " " \* (max\_len - len(str(number)))

sv += atkape + str(number)

if j < kolonnas - 1 and (j + 1) % 3 != 0:

sv = sv + " "

elif j < kolonnas - 1 and (j + 1) % 3 == 0:

sv = sv + " "

sv = sv + "\n"

if (i + 1) % 3 == 0 and i < rindas - 1:

sv = sv + "\n"

return sv

# ---------------------------------------------------------

# Galvenā programmas daļa

# ---------------------------------------------------------

a = generate\_sudoku()

if check\_array\_rows\_and\_columns(a) and check\_3x3\_submatrixes(a):

print("Korekti aizpildīta Sudoku spēlēs lapiņa:")

print()

else:

print("Nav korekti aizpildīta Sudoku spēlēs lapiņa:")

print(matrix\_to\_string\_float\_3x3(a))

print("-------------------\n")

c = delete\_cells(a, 40)

print("Sudoku lapiņa ar nejauši izdzēstiem skaitļiem:\n")

print(matrix\_to\_string\_float\_3x3(c))

**Testa piemēri:**

1)

A screenshot of a computer

Description automatically generated with medium confidence

2)

A screenshot of a computer

Description automatically generated with medium confidence

3)

A screenshot of a computer

Description automatically generated with medium confidence

4)

A screenshot of a computer

Description automatically generated with medium confidence

**2. uzdevums**

Sastādīt programmu, kas realizē pamatdarbības ar kompleksiem skaitļiem.

**Kods:**

# Programmas nosaukums: Kompleksa skaitļi

# 2.uzdevums (1MPR13\_Vladislavs\_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas realizē pamatdarbības ar kompleksiem skaitļiem.

# Programmas autors: Vladislavs Babaņins

# Versija 1.0

import math

class ComplexNumber:

# Kompleksu skaitļu klase.

def \_\_init\_\_(self, re=0, im=0):

# Pēc noklusējuma izveido tukšu komplēksa skaitli (0 + 0i)

# Ja ir norādots citādi, tad izveido tā, ka ievadīja lietotājs.

self.re = re

self.im = im

def \_\_repr\_\_(self):

# print()

# Kompleksā skaitļu izvadīšanai lietotājam.

if self.re != 0: # Ja ir kāda reāla daļa, tad izvadam komplēkso skaitli ar realu daļu (neviss 0 + i\*n)

if self.im > 0 and self.im != 1: # Ja imagināra daļa nav 1 un tā ir lielāka par 0, tad rakstām n + i, neviss n + k\*i

return f"{self.re} + {self.im}i"

elif self.im == 1: # Ja imagināra daļa ir 1, tad rakstām n + i, neviss n + 1\*i

return f"{self.re} + i"

elif self.im == 0: # Ja imagināra daļa ir 0, tad rakstām tikai reālu daļu n

return f"{self.re}"

elif self.im == -1: # Ja imagināra daļa ir -1, tad rakstām n - i, neviss n - 1\*i

return f"{self.re} - i"

else: # Citā gadījumā rakstām n - k\*i

return f"{self.re} - {-self.im}i"

else: # Ja nav reālas daļas, tad nav jēgas rakstīt 0 + k\*i, tad izvadam komplēkso skaitli tikai ar imagināru daļu k\*i

if self.im > 0 and self.im != 1: # Ja imagināra daļa ir pozitīva un nav viens, tad izvadam k\*i, neviss 0 + k\*i

return f"{self.im}i"

elif self.im == 1: # Ja imagināra daļa ir 1, tad izvadam i, neviss 0 + 1\*i

return "i"

elif self.im == 0: # Ja imagināra daļa ir 0, tad izvadam 0, neviss 0 + 0\*i

return "0"

elif self.im == -1: # Ja imagināra daļa ir 1, tad izvadam -i, neviss 0 - 1\*i

return "-i"

else: # Citādi izvādam -k\*i (nav realas daļas un imagināra daļa ir negatīva un nav -1)

return f"-{-self.im}i"

def arg(self):

# Atgriež komplēksa skaitļa argumentu.

return math.atan2(self.im, self.re)

def \_\_add\_\_(self, other):

# +

# Atgriež komplēksa skaitļa summu (self + other).

real\_sum = self.re + other.re

imaginary\_sum = self.im + other.im

return ComplexNumber(real\_sum, imaginary\_sum)

def \_\_iadd\_\_(self, other):

# +=

# Atgriež komplēksa skaitļa summu (self + other), bet kā \_\_iadd\_\_ (+=).

# Atgriež jau eksistējošu mainīgu, neviss izveido jaunu mainīgu.

self.re += other.re

self.im += other.im

return self

def \_\_sub\_\_(self, other):

# -

# Atgriež komplēksa skaitļa starpību (self - other).

real\_diff = self.re - other.re

imaginary\_diff = self.im - other.im

return ComplexNumber(real\_diff, imaginary\_diff)

def \_\_isub\_\_(self, other):

# -=

# Atgriež komplēksa skaitļa summu (self - other), bet kā \_\_isub\_\_ (-=).

# Atgriež jau eksistējošu mainīgu, neviss izveido jaunu mainīgu.

self.re -= other.re

self.im -= other.im

return self

def \_\_mul\_\_(self, other):

# \*

# Atgriež komplēksa skaitļa reizinājumu (self \* other).

real\_product = (self.re \* other.re) - (self.im \* other.im)

imaginary\_product = (self.re \* other.im) + (self.im \* other.re)

return ComplexNumber(real\_product, imaginary\_product)

def \_\_imul\_\_(self, other):

# \*=

# Atgriež komplēksa skaitļa reizinājumu (self \* other) bet kā \_\_imul\_\_ (\*=).

# Atgriež jau eksistējošu mainīgu, neviss izveido jaunu mainīgu.

re1 = self.re

im1 = self.im

self.re = (re1 \* other.re) - (im1 \* other.im)

self.im = (re1 \* other.im) + (im1 \* other.re)

return self

def \_\_truediv\_\_(self, other):

# /

# Atgriež komplēksa skaitļa dalījumu (self / other).

denominator = (other.re \* other.re) + (other.im \* other.im)

real\_quotient = ((self.re \* other.re) + (self.im \* other.im)) / denominator

imaginary\_quotient = ((self.im \* other.re) - (self.re \* other.im)) / denominator

return ComplexNumber(real\_quotient, imaginary\_quotient)

def \_\_itruediv\_\_(self, other):

# /=

# Atgriež komplēksa skaitļa dalījumu (self / other) bet kā \_\_itruediv\_\_ (/=).

# Atgriež jau eksistējošu mainīgu, neviss izveido jaunu mainīgu.

denominator = (other.re \*\* 2) + (other.im \*\* 2)

real\_quotient = ((self.re \* other.re) + (self.im \* other.im)) / denominator

imaginary\_quotient = ((self.im \* other.re) - (self.re \* other.im)) / denominator

self.re = real\_quotient

self.im = imaginary\_quotient

return self

def \_\_abs\_\_(self):

# Atgriež komplēksa skaitļa moduli.

return math.sqrt(self.re \* self.re + self.im \* self.im)

def conjugate(self):

# Atgriež komplēksa skaitļas komplēksa saistīto skaitli.

return ComplexNumber(self.re, -self.im)

def \_\_pow\_\_(self, power):

# Atgriež komplēksa skaitļi, kurš tika pacēlts naturāla pakāpe.

modulus = self.\_\_abs\_\_() \*\* power

arg = power \* self.arg()

re = modulus \* math.cos(arg)

im = modulus \* math.sin(arg)

return ComplexNumber(re, im)

def complex\_power(z, n):

# Atgriež komplēksa skaitļi, kurš tika pacēlts pakāpe.

r = math.sqrt(z.re\*\*2 + z.im\*\*2)

theta = math.atan2(z.im, z.re)

re = r \*\* n \* math.cos(n \* theta)

im\_part = r \*\* n \* math.sin(n \* theta)

return ComplexNumber(re, im\_part)

def n\_roots(self, n):

# Atgriež sarakstu, ar visiem komplēksa skaitļa saknēm.

# n - kuru sakni gribām izvilkt

roots = []

modulus = abs(self)

arg = self.arg()

for k in range(n):

root\_argument = (arg + 2 \* k \* math.pi) / n

re = modulus \* math.cos(root\_argument)

im\_part = modulus \* math.sin(root\_argument)

roots.append(ComplexNumber(re, im\_part))

return roots

def trigonometric\_form(self):

# Izvadīt lietotājam komplēksu skaitli trigonometriskajā formā.

r = abs(self)

theta = self.arg()

return f"{r:.2f}(cos({theta:.2f}) + isin({theta:.2f}))"

def exponent\_form(self):

# Izvadīt lietotājam komplēksu skaitli eksponenciāla formā.

modulus = abs(self)

arg = self.arg()

return f"{modulus} \* e^({arg}i)"

# ---------------------------------------------------------

# Galvenā programmas daļa

# ---------------------------------------------------------

print("Operācijas ar komplēksa skaitliem:")

num1 = ComplexNumber(-1, -4)

num2 = ComplexNumber(1, 3)

# print(num1 + num4)

print("(" + str(num1) + ") + (" + str(num2) + ") = " + str(num1 + num2))

num1 = ComplexNumber(-1, 4)

num2 = ComplexNumber(1, -3)

# print(num1 + num4)

print("(" + str(num1) + ") + (" + str(num2) + ") = " + str(num1 + num2))

num1 = ComplexNumber(2, 4)

num2 = ComplexNumber(1, -3)

# print(num1 + num4)

print("(" + str(num1) + ") + (" + str(num2) + ") = " + str(num1 + num2))

num1 = ComplexNumber(-1, -4)

num2 = ComplexNumber(-1, 3)

# print(num1 + num4)

print("(" + str(num1) + ") + (" + str(num2) + ") = " + str(num1 + num2))

num3 = ComplexNumber(19, 15)

num4 = ComplexNumber(19, 15)

# print(num3 - num4)

print("(" + str(num3) + ") - (" + str(num4) + ") = " + str(num3 - num4))

# print(num1 / num2)

# print(num3 / num4)

print("(" + str(num1) + ") / (" + str(num2) + ") = " + str(num1 / num2))

print("(" + str(num3) + ") / (" + str(num4) + ") = " + str(num3 / num4))

# print(num1 \* num2)

# print(num3 \* num4)

print("(" + str(num1) + ") \* (" + str(num2) + ") = " + str(num1 \* num2))

print("(" + str(num3) + ") \* (" + str(num4) + ") = " + str(num3 \* num4))

# print(abs(num1))

# print(abs(num3))

print("|" + str(num1) + "| = " + str(abs(num1)))

print("|" + str(num3) + "| = " + str(abs(num3)))

# print(conjugate(num1))

# print(conjugate(num3))

print("conj(" + str(num1) + ") = " + str(ComplexNumber.conjugate(num1)))

print("conj(" + str(num3) + ") = " + str(ComplexNumber.conjugate(num3)) + "\n")

num5 = ComplexNumber(2, 4)

num6 = ComplexNumber(3, 6)

numiadd = ComplexNumber(3, 6)

numiadd1 = ComplexNumber(3, 6)

numiadd += num5

print("Izmantojot \_\_iadd\_\_:")

print("(" + str(numiadd1) + ") + (" + str(num5) + ") = " + str(numiadd) + "\n")

print("Izmantojot \_\_add\_\_:")

num\_add = num5 + num6

print("(" + str(num5) + ") + (" + str(num6) + ") = " + str(num\_add) + "\n")

numisub = ComplexNumber(3, 6)

numisub1 = ComplexNumber(3, 6)

numisub -= num5

print("Izmantojot \_\_isub\_\_:")

print("(" + str(numisub1) + ") - (" + str(num5) + ") = " + str(numisub) + "\n")

numisub = ComplexNumber(3, 6)

numisub1 = ComplexNumber(3, 6)

numisub = numisub1 - num5

print("Izmantojot \_\_sub\_\_:")

print("(" + str(numisub1) + ") - (" + str(num5) + ") = " + str(numisub) + "\n")

num7 = ComplexNumber(0.5, math.sqrt(3) / 2)

num8 = ComplexNumber(5, 5)

power1 = 20

print("Komplēksa skaitļa pakāpe:")

print("(" + str(num7) + ") \*\* " + str(power1) + " = " + str(num7 \*\* 20) + "\n")

z = ComplexNumber(3, 4)

z\_cubed = ComplexNumber.complex\_power(z, 3)

print("Komplēksa skaitļa pakāpe:")

print("(" + str(z) + ") \*\* " + str(3) + " = " + str(z\_cubed) + "\n")

b = ComplexNumber(0.5, math.sqrt(3) / 2)

print("Komplēksa skaitļa saknes:")

print("sqrt(" + str(b) + ") = " + str(ComplexNumber.n\_roots(b, 2)) + "\n")

b = ComplexNumber(0.5, math.sqrt(3) / 2)

print("Komplēksa skaitļa saknes:")

print("sqrt(" + str(b) + ") = " + str(ComplexNumber.n\_roots(b, 4)) + "\n")

c = ComplexNumber(1, 1)

print("Komplēksa saistītajs:")

print("conj(" + str(c) + ") = " + str(ComplexNumber.conjugate(c)) + "\n")

d = ComplexNumber(-1, 1)

print("Pārveidot trigonometriskā formā:")

print("(" + str(d) + ") = " + str(ComplexNumber.trigonometric\_form(d)) + "\n")

print("Pārveidot eksponenciālā formā:")

t = ComplexNumber(1, math.sqrt(3) / 3)

print("(" + str(t) + ") = " + str(ComplexNumber.exponent\_form(t)) + "\n")

t = ComplexNumber(1, math.sqrt(3) / 3)

print("Komplēksa skaitļa arguments:")

print("arg(" + str(t) + ") = " + str(ComplexNumber.arg(t)) + "\n")

k = ComplexNumber(1, 2)

print("Komplēksa skaitļa izvadīšana (print):")

print(k)

print()

num5 = ComplexNumber(2, 4)

numimul = ComplexNumber(3, 6)

numimul1 = ComplexNumber(3, 6)

numimul \*= num5

print("Izmantojot \_\_imul\_\_:")

print("(" + str(numimul1) + ") \* (" + str(num5) + ") = " + str(numimul) + "\n")

num5 = ComplexNumber(2, 4)

numimul = ComplexNumber(3, 6)

numimul1 = ComplexNumber(3, 6)

numimul1 = numimul \* num5

print("Izmantojot \_\_mul\_\_:")

print("(" + str(numimul) + ") \* (" + str(num5) + ") = " + str(numimul1) + "\n")

num5 = ComplexNumber(2, 4)

numidiv = ComplexNumber(3, 6)

numidiv1 = ComplexNumber(3, 6)

numidiv /= num5

print("Izmantojot \_\_itruediv\_\_:")

print("(" + str(numidiv1) + ") / (" + str(num5) + ") = " + str(numidiv) + "\n")

num5 = ComplexNumber(2, 4)

numidiv = ComplexNumber(3, 6)

numidiv1 = ComplexNumber(3, 6)

numidiv1 = numidiv / num5

print("Izmantojot \_\_truediv\_\_:")

print("(" + str(numimul) + ") / (" + str(num5) + ") = " + str(numidiv1) + "\n")

print("Jaunais komplēksa skaitlis:")

new\_num = ComplexNumber()

print(new\_num)

**Testa piemēri:**

1)

Operācijas ar komplēksa skaitliem:

(-1 - 4i) + (1 + 3i) = -i

(-1 + 4i) + (1 - 3i) = i

(2 + 4i) + (1 - 3i) = 3 + i

(-1 - 4i) + (-1 + 3i) = -2 - i

(19 + 15i) - (19 + 15i) = 0

(-1 - 4i) / (-1 + 3i) = -1.1 + 0.7i

(19 + 15i) / (19 + 15i) = 1.0

(-1 - 4i) \* (-1 + 3i) = 13 + i

(19 + 15i) \* (19 + 15i) = 136 + 570i

|-1 - 4i| = 4.123105625617661

|19 + 15i| = 24.20743687382041

conj(-1 - 4i) = -1 + 4i

conj(19 + 15i) = 19 - 15i

Izmantojot \_\_iadd\_\_:

(3 + 6i) + (2 + 4i) = 5 + 10i

Izmantojot \_\_add\_\_:

(2 + 4i) + (3 + 6i) = 5 + 10i

Izmantojot \_\_isub\_\_:

(3 + 6i) - (2 + 4i) = 1 + 2i

Izmantojot \_\_sub\_\_:

(3 + 6i) - (2 + 4i) = 1 + 2i

Komplēksa skaitļa pakāpe:

(0.5 + 0.8660254037844386i) \*\* 20 = -0.4999999999999961 + 0.8660254037844384i

Komplēksa skaitļa pakāpe:

(3 + 4i) \*\* 3 = -117.0 + 44.000000000000036i

Komplēksa skaitļa saknes:

sqrt(0.5 + 0.8660254037844386i) = [0.8660254037844386 + 0.4999999999999999i, -0.8660254037844387 - 0.49999999999999967i]

Komplēksa skaitļa saknes:

sqrt(0.5 + 0.8660254037844386i) = [0.9659258262890682 + 0.2588190451025207i, -0.2588190451025206 + 0.9659258262890682i, -0.9659258262890682 - 0.25881904510252074i, 0.25881904510252024 - 0.9659258262890683i]

Komplēksa saistītajs:

conj(1 + i) = 1 - i

Pārveidot trigonometriskā formā:

(-1 + i) = 1.41(cos(2.36) + isin(2.36))

Pārveidot eksponenciālā formā:

(1 + 0.5773502691896257i) = 1.1547005383792515 \* e^(0.5235987755982988i)

Komplēksa skaitļa arguments:

arg(1 + 0.5773502691896257i) = 0.5235987755982988

Komplēksa skaitļa izvadīšana (print):

1 + 2i

Izmantojot \_\_imul\_\_:

(3 + 6i) \* (2 + 4i) = -18 + 24i

Izmantojot \_\_mul\_\_:

(3 + 6i) \* (2 + 4i) = -18 + 24i

Izmantojot \_\_itruediv\_\_:

(3 + 6i) / (2 + 4i) = 1.5

Izmantojot \_\_truediv\_\_:

(3 + 6i) / (2 + 4i) = 1.5

Jaunais komplēksa skaitlis:

0

2)

A screenshot of a computer program

Description automatically generated with low confidence

3)

A screenshot of a computer program

Description automatically generated with medium confidence

**3. uzdevums (I variants)**

Sastādīt programmu, kas aprēķina N-stūra laukumu, tā virsotņu koordinātas ievadot no tastatūras (ar ierakstiem).

**Kods:**

# Programmas nosaukums: N-stūra laukums ar ierakstiem

# 3.uzdevums (1MPR13\_Vladislavs\_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas aprēķina N-stūra laukumu, tā virsotņu koordinātas ievadot no tastatūras.

# Programmas autors: Vladislavs Babaņins

# Versija 1.0

"""

UZDEVUMS TIKA REALIZĒTS AR IERAKSTIEM.

"""

import types

import numpy

def input\_polygon\_coords(n):

# Paprasa lietotājam ievadīt virsotnes X un Y koordinātas.

# Atgriež ierakstu a, ar lietotāja ievadītam X un Y koordinātam.

# n - naturāls skaitlis - daudzstūra virsotņu skaits.

for i in range(1, n + 1):

punkts = types.SimpleNamespace()

temp\_x = input("Ievadiet " + str(i) + ". virsotnes X koordināti ==> ")

while not is\_real\_check(temp\_x):

temp\_x = input("Kļūda! Ievadiet reālu skaitli! Ievadiet " + str(i) + ". virsotnes X koordināti ==> ")

temp\_x = float(temp\_x)

temp\_y = input("Ievadiet " + str(i) + ". virsotnes Y koordināti ==> ")

while not is\_real\_check(temp\_y):

temp\_y = input("Kļūda! Ievadiet reālu skaitli! Ievadiet " + str(i) + ". virsotnes X koordināti ==> ")

temp\_y = float(temp\_y)

print()

punkts.x = temp\_x

punkts.y = temp\_y

a[i] = punkts

return a

def area\_using\_shoelace\_methode\_list(a):

# Aprēķina laukumu daudzstūrim, kurā punktu koordinātas ir definētas ierakstā a.

# a - ieraksts, kurā glābajas daudzstūra koordinātas.

# Ieraksta piemērs: [0 namespace(x=1.0, y=2.0) namespace(x=3.0, y=4.0)].

# Atgriež daudzstūra laukumu izmantojot "Shoelace formula".

a[0] = a[n] # Fiktīvais punkts.

s = 0

for i in range(n):

x = a[i].x + a[i + 1].x

y = a[i].y - a[i + 1].y

s = s + x \* y

s = abs(s) / 2

return s

def is\_real\_check(n):

# Pārbauda vai simbolu virkne ir reāls skaitlis vai nav.

# Atgriež True, ja tas ir reāls skaitlis (float).

# Atgriež False, ja tas nav reāls skaitlis (float).

# n - pārbaudāma simbolu virkne.

try:

float(n)

except:

return False

else:

return True

def is\_natural(n):

# Pārbauda vai simbolu virkne ir naturāls skaitlis vai nav

# Ja ir naturāls skaitlis, tad True. Ja nav tad False.

# n - simbolu virkne, kuru pārbauda.

if str(n).isdigit() and float(n) == int(n) and int(n) > 0:

return True

else:

return False

# ---------------------------------------------------------

# Galvenā programmas daļa

# ---------------------------------------------------------

n = input("Ievadiet virsotņu skaitu ==> ")

while not is\_natural(n):

n = input("Kļūda! Virsotņu skaitam ir jābūt naturālam skaitlim! Ievadiet virsotņu skaitu ==> ")

n = int(n)

a = numpy.zeros(n + 1, "O")

print()

koord\_saraksts = input\_polygon\_coords(n)

area = area\_using\_shoelace\_methode\_list(koord\_saraksts)

print("Laukums ievadītam daudzstūrim:")

print("S = " + str(area))

**Testa piemēri:**

1)

A screenshot of a computer code

Description automatically generated with low confidence

2)

A screenshot of a computer program

Description automatically generated with medium confidence

3)

A picture containing text, screenshot, font

Description automatically generated

4)

A picture containing text, screenshot, font

Description automatically generated

**3. uzdevums (II variants)**

Sastādīt programmu, kas aprēķina N-stūra laukumu, tā virsotņu koordinātas ievadot no tastatūras. (ar vārdnīcām).

**Kods:**

# Programmas nosaukums: N-stūra laukums ar vārdnīcām

# 3.uzdevums (1MPR13\_Vladislavs\_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas aprēķina N-stūra laukumu, tā virsotņu koordinātas ievadot no tastatūras.

# Programmas autors: Vladislavs Babaņins

# Versija 1.0

"""

UZDEVUMS TIKA REALIZĒTS AR VĀRDNĪCAM.

"""

import types

def area\_using\_shoelace\_methode\_dict(vardnica, n):

# Aprēķina laukumu daudzstūrim, kurā punktu koordinātas ir definētas vārdnīcā, šādā veidā {"Koordinātas nosaukums" : (x, y)}.

# Atgriež daudzstūra laukumu izmantojot "Shoelace formula".

# vardnica - vārdnīca, kurā ir saraksts ar koordinātam šadā veidā {"Koordinātas nosaukums" : (x, y)}.

# n - naturāls skaitlis - daudzstūra virsotņu skaits.

# Atgriež daudzstūra laukumu.

s = 0

for i in range(0, n):

# Izmantojot Gausa formulu atrodam laukumu daudzstūrim (Shoelace formula).

x = vardnica[t[i]].x + vardnica[t[i - 1]].x

y = vardnica[t[i]].y - vardnica[t[i - 1]].y

s = s + x \* y

s = abs(s) / 2

return s

def create\_dict\_with\_coords(n):

# Paprasa lietotājam ievadīt virsotnes nosaukumus un tā X un Y koordinātas.

# Izveido vārdnīcu tāda veidā {"Virsotnes nosaukums" : (x, y)}, kur katru "Virsotnes nosaukums" ievada lietotājs, (x, y) arī ievada lietotājs.

# Atgriež vārdnīcu tāda veidā {"Virsotnes nosaukums" : (x, y)}.

# n - naturāls skaitlis - daudzstūra virsotņu skaits.

i = 0

for i in range(i, n):

name = input("Ievadiet " + str(i + 1) + ". virsotnes nosaukumu ==> ")

punkts = types.SimpleNamespace()

temp\_x = input("Ievadiet " + str(i + 1) + ". virsotnes X koordināti ==> ")

while not is\_real\_check(temp\_x):

temp\_x = input("Kļūda! Ievadiet reālu skaitli! Ievadiet " + str(i + 1) + ". virsotnes X koordināti ==> ")

temp\_x = float(temp\_x)

temp\_y = input("Ievadiet " + str(i + 1) + ". virsotnes Y koordināti ==> ")

while not is\_real\_check(temp\_y):

temp\_y = input("Kļūda! Ievadiet reālu skaitli! Ievadiet " + str(i + 1) + ". virsotnes X koordināti ==> ")

temp\_y = float(temp\_y)

punkts.x = temp\_x

punkts.y = temp\_y

vardnica.update({name: punkts})

t.append(name)

t.append("Fiktīvais punkts")

vardnica["Fiktīvais punkts"] = vardnica[t[n - 1]]

return vardnica

def is\_natural(n):

# Pārbauda vai simbolu virkne ir naturāls skaitlis vai nav

# Ja ir naturāls skaitlis, tad True. Ja nav tad False.

# n - simbolu virkne, kuru pārbauda.

if str(n).isdigit() and float(n) == int(n) and int(n) > 0:

return True

else:

return False

def is\_real\_check(n):

# Pārbauda vai simbolu virkne ir reāls skaitlis vai nav.

# Atgriež True, ja tas ir reāls skaitlis (float).

# Atgriež False, ja tas nav reāls skaitlis (float).

# n - pārbaudāma simbolu virkne.

try:

float(n)

except:

return False

else:

return True

# ---------------------------------------------------------

# Galvenā programmas daļa

# ---------------------------------------------------------

vardnica = {}

t = []

n = input("Ievadiet virsotņu skaitu ==> ")

while not is\_natural(n):

n = input("Kļūda! Virsotņu skaitam ir jābūt naturālam skaitlim! Ievadiet virsotņu skaitu ==> ")

n = int(n)

create\_dict\_with\_coords(n)

s = area\_using\_shoelace\_methode\_dict(vardnica, n)

vardnica.pop("Fiktīvais punkts") # izņemt no vārdnīcas atslēgu-vērtību pāri "Fiktīvais punkts".

print()

print("Punktu sarakstu izvadīšanas veids:")

print("\"Punkta nosaukums\" : (x, y)")

print(vardnica)

print("S = " + str(s))

**Testa piemēri:**

A picture containing screenshot

Description automatically generated1)

A picture containing screenshot

Description automatically generated2)

A picture containing text, screenshot, font

Description automatically generated3)

A screen shot of a computer

Description automatically generated with medium confidence4)

**4. uzdevums (Tekstuāla saskarne)**

Sastādīt programmu, kas realizē iepirkšanos veikalā. Tiek nodrošināta iegādāto preču ievade (nosaukums, daudzums un cena) un "pirkumu čeka" izdrukāšana. (Vienkārša tekstuāla saskarne - 1 punkts, komplicēta grafiskā saskarne - 3 punkti.)

**Kods (Tekstuāla saskarne):**

# Programmas nosaukums: Veikals

# 4.uzdevums (1MPR13\_Vladislavs\_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas realizē iepirkšanos veikalā. Tiek nodrošināta iegādāto preču ievade (nosaukums, daudzums un cena)

# un "pirkumu čeka" izdrukāšana. (Vienkārša tekstuāla saskarne - 1 punkts, komplicēta grafiskā saskarne - 3 punkti.)

# Programmas autors: Vladislavs Babaņins

# Versija 1.0

"""

UZDEVUMS TIKA REALIZĒTS AR TEKSTUALU SASKARNI.

"""

from random import randint

class Veikals:

# Klase veikals.

def \_\_init\_\_(self):

# Izveidojam preces un selected\_preces (izvelētas preces) vārdnīcas.

# self.preces - vārdnica ar visam precem veikalā.

# self.selected\_preces - vārdnica ar lietotāja izvelētam precem.

self.preces = dict()

self.selected\_preces = dict()

def pievienot\_preces(self, prece, cena, daudzums):

# Metode veikala īpašniekam.

# Preces pievienošana veikalām. Atjaunojam vārdnicu self.preces, preci, ar norādītu cenu un daudzumu.

# prece - preces nosaukums (str).

# cena - preces cena par vienu vienumu (float).

# daudzums - cik daudz preci jāpiegāda veikalām (naturāls skaitlis lielāks par 0) (int).

self.preces[prece] = {"cena": cena, "daudzums": daudzums}

def dzest\_preces(self, prece):

# Metode veikala īpašniekam.

# Nodzēs norādītu preci no veikalā, nodzēs noradītu preci no saraksta.

# prece - preces nosaukums (str).

print(f"Prece \"{prece}\" tika izmesta Getliņu izgāztuvē.")

del self.preces[prece]

def paradit\_izveletas\_preces(self):

# Izdruka visas lietotaja izvelētas preces.

print("Jūsu izvelētas preces:")

for prece, info in self.selected\_preces.items():

cena = info["cena"]

daudzums = info["daudzums"]

print(f"{prece} - cena: {cena:.2f}€ , daudzums: {daudzums}")

def paradit\_preces(self):

# Izdruka visas veikalā pieejamas preces lietotājam.

print("Pieejamas preces veikalā:")

for prece, info in self.preces.items():

cena = info["cena"]

daudzums = info["daudzums"]

print(f"{prece} - cena: {cena:.2f}€ , daudzums: {daudzums}")

def izveleties\_preces(self, prece, daudzums):

# Metode preces izvelēšanai lietotājam, jāizvēlas preci un to daudzumu.

# prece - preces nosaukums (str).

# daudzums - preces daudzums (int).

if prece in self.preces: # Ja tāda prece ir pieejama veikalā, tad tālak pārbaudam.

if daudzums > self.preces[prece]["daudzums"]: # Ja pieprasītais daudzums ir lielāks nekā ir veikalā, tad paziņojam, ka nevar nopirkt.

print(f"Kļūda! Veikalā nav tik daudz preču! Precei \"{prece}\" pieejamais daudzums: {self.preces[prece]['daudzums']}")

else:

self.preces[prece]["daudzums"] -= daudzums # Atņēmam nopirktu daudzumu.

if prece not in self.selected\_preces:

self.selected\_preces[prece] = {"cena": self.preces[prece]["cena"], "daudzums": daudzums}

else:

self.selected\_preces[prece]["daudzums"] += daudzums # Pievienojam daudzumu, cik daudz preces izvelējamies nopirkt.

print(f"Izvēlētas preces: {prece}, cena: " + str(format(self.preces[prece]["cena"], ".2f")) + f"€ , daudzums: {daudzums}")

else:

print(f"Kļūda! Prece \"{prece}\" nav pieejama veikalā!") # Ja tādas preces nav veikalā, tad paziņojam to.

def atteikt\_prece(self, prece, daudzums=None):

# Lietotājs var atgriezt veikālā kādu no jau izvelētam precem ar norādīto daudzumu.

# Ja lietotājs nenorāda daudzumu, tad tiek atgriezti visas preces.

# prece - preces nosaukums (str).

# daudzums - preces daudzums (int).

if prece in self.selected\_preces:

if daudzums is None: # Ja lietotājs nenorāda daudzumu, tad tiek atgriezti visas preces.

daudzums = self.selected\_preces[prece]["daudzums"]

if daudzums > self.selected\_preces[prece]["daudzums"]: # Ja lietotājs grib atgriezt vairāk nekā paņem, tad kļūda.

print(f"Kļūda! Precei \"{prece}\" ir izvēlēts mazāks daudzums. Nevar atgriezt vairāk nekā paņemat!")

elif daudzums < 0: # Ja lietotājs grib atgriezt negatīvu daudzumu, tad kļūda.

print(f"Kļūda! Nevar atgriezt negatīvu daudzumu!")

elif daudzums == 0: # Ja lietotājs grib atgriezt neko, tad kļūda.

print(f"Kļūda! Nevar atgriezt neko!")

else:

self.selected\_preces[prece]["daudzums"] -= daudzums # Atņēmam noradītu daudzumu no izvelētam precēm.

self.preces[prece]["daudzums"] += daudzums # Pievienojam noradītu daudzumu par izvelētam precēm.

if self.selected\_preces[prece]["daudzums"] == 0: # Ja atņēmam tik daudz, ka vairāk tadas preces nav (nav izvelēta), tad nodzēsam to.

del self.selected\_preces[prece]

print(f"Izvēlētā prece \"{prece}\" tika atgriezta veikalā. Atgrieztais daudzums: {daudzums}") # Paziņojam lietotājam informāciju.

else:

print(f"Kļūda! Prece netika \"{prece}\" izvēlēta.") # Paziņojam lietotājam.

def pirkumu\_cekis(self):

# Izdruka pirkumu lietotājam "pirkumu čeku", ar izvelētam precēm.

total\_price = 0

print("---------------- ČEKS ----------------")

print("SIA \"MAKSIMA\" Latvija")

print("Veikals \"Maksima\"")

print("Dārza iela 21, Bauska, t.22813371")

print("Juridiskā adrese: \"Abras\", Krustkalni,")

print("Ķekavas pagasts, Ķekavas novads, LV-2222")

print("Čeks " + str(randint(100, 999)) + "/" + str(randint(100, 999)) + " #" + str(randint(10000000, 99999999)))

print("==========================================")

for prece, info in self.selected\_preces.items():

cena = info["cena"]

daudzums = info["daudzums"]

preces\_cena = cena \* daudzums

total\_price += preces\_cena

print(f"{prece} - cena: {cena:.2f}€, daudzums: {daudzums}, {preces\_cena:.2f}€")

print("==========================================")

print(f"Kopā apmaksai: {total\_price:.2f}€")

print(f"Nopelnīta \"MAKSIMA\" nauda: {total\_price \* 0.01:.2f}€")

print("---------------------------------------")

# ---------------------------------------------------------

# Galvenā programmas daļa

# ---------------------------------------------------------

veikals = Veikals()

veikals.pievienot\_preces("Āboli", 1.00, 50)

veikals.pievienot\_preces("Piens", 1.40, 20)

veikals.pievienot\_preces("Olas", 1.50, 30)

veikals.pievienot\_preces("Pipari", 1.00, 100)

veikals.paradit\_preces()

print()

veikals.dzest\_preces("Piens")

print()

veikals.paradit\_preces()

print()

veikals.izveleties\_preces("Olas", 60)

veikals.izveleties\_preces("Olas", 30)

veikals.izveleties\_preces("Vārdnīca", 1)

veikals.izveleties\_preces("Āboli", 50)

veikals.izveleties\_preces("Pipari", 66)

print()

veikals.paradit\_izveletas\_preces()

print()

veikals.atteikt\_prece("Āboli", 30)

veikals.atteikt\_prece("Olas")

print()

veikals.paradit\_izveletas\_preces()

print()

veikals.atteikt\_prece("Āboli", -10)

print()

veikals.paradit\_izveletas\_preces()

print()

veikals.atteikt\_prece("Āboli", 0)

veikals.paradit\_izveletas\_preces()

print()

print()

veikals.pirkumu\_cekis()

print()

print()

veikals.paradit\_preces()

**Testa piemēri:**

1)

Pieejamas preces veikalā:

Āboli - cena: 1.00€ , daudzums: 50

Piens - cena: 1.40€ , daudzums: 20

Olas - cena: 1.50€ , daudzums: 30

Pipari - cena: 1.00€ , daudzums: 100

Prece "Piens" tika izmesta Getliņu izgāztuvē.

Pieejamas preces veikalā:

Āboli - cena: 1.00€ , daudzums: 50

Olas - cena: 1.50€ , daudzums: 30

Pipari - cena: 1.00€ , daudzums: 100

Kļūda! Veikalā nav tik daudz preču! Precei "Olas" pieejamais daudzums: 30

Izvēlētas preces: Olas, cena: 1.50€ , daudzums: 30

Kļūda! Prece "Vārdnīca" nav pieejama veikalā!

Izvēlētas preces: Āboli, cena: 1.00€ , daudzums: 50

Izvēlētas preces: Pipari, cena: 1.00€ , daudzums: 66

Jūsu izvelētas preces:

Olas - cena: 1.50€ , daudzums: 30

Āboli - cena: 1.00€ , daudzums: 50

Pipari - cena: 1.00€ , daudzums: 66

Izvēlētā prece "Āboli" tika atgriezta veikalā. Atgrieztais daudzums: 30

Izvēlētā prece "Olas" tika atgriezta veikalā. Atgrieztais daudzums: 30

Jūsu izvelētas preces:

Āboli - cena: 1.00€ , daudzums: 20

Pipari - cena: 1.00€ , daudzums: 66

Kļūda! Nevar atgriezt negatīvu daudzumu!

Jūsu izvelētas preces:

Āboli - cena: 1.00€ , daudzums: 20

Pipari - cena: 1.00€ , daudzums: 66

Kļūda! Nevar atgriezt neko!

Jūsu izvelētas preces:

Āboli - cena: 1.00€ , daudzums: 20

Pipari - cena: 1.00€ , daudzums: 66

---------------- ČEKS ----------------

SIA "MAKSIMA" Latvija

Veikals "Maksima"

Dārza iela 21, Bauska, t.22813371

Juridiskā adrese: "Abras", Krustkalni,

Ķekavas pagasts, Ķekavas novads, LV-2222

Čeks 442/706 #80908615

==========================================

Āboli - cena: 1.00€, daudzums: 20, 20.00€

Pipari - cena: 1.00€, daudzums: 66, 66.00€

==========================================

Kopā apmaksai: 86.00€

Nopelnīta "MAKSIMA" nauda: 0.86€

---------------------------------------

Pieejamas preces veikalā:

Āboli - cena: 1.00€ , daudzums: 30

Olas - cena: 1.50€ , daudzums: 30

Pipari - cena: 1.00€ , daudzums: 34

2)

A screenshot of a computer screen

Description automatically generated with medium confidence

3)

A screenshot of a computer screen

Description automatically generated with medium confidence