9. praktiskais darbs. 2. semestris

PU2. uzdevums

Realizēt kuģīšu šaušanas spēli jeb spēli KARTUPELIS – 2 spēlētāji dators un lietotājs.

1. Līmenis abu spēlētāju kuģu izvietojumu ievada lietotājs un uz ekrāna redzamas tikai lietotāja gājieni datora "jūrā" un lietotāja "jūra ar kuģiem" un datora izdarītiem gājieniem. Katrs secīgi var izdarīt tikai vienu gājienu.
2. Līmenis – dators pats izvieto savus kuģus "jūrā"
3. Līmenis – dators šāvienus izdara plānveidīgi nevis uz labu laimi.

Kods:

```
# Programmas nosaukums: Kuģīšu šaušanas spēle. 2.LĪMENIS.

# Papilduzdevums 2. (1MPR09_Vladislavs_Babaņins)

# Uzdevuma formulējums: Realizēt kuģīšu šaušanas spēli jeb spēli KARTUPELIS – 2 spēlētāji dators un lietotājs.

# Realizēt kuģīšu šaušanas spēli jeb spēli KARTUPELIS – 2 spēlētāji dators un lietotājs.

# 1. Līmenis abu spēlētāju kuģu izvietojumu ievada lietotājs un uz ekrāna redzamas tikai lietotāja gājieni datora "jūrā" un lietotāja "jūra ar kuģiem" un datora izdarītiem gājieniem.

#    Katrs secīgi var izdarīt tikai vienu gājienu.

# 2. Līmenis – dators pats izvieto savus kuģus "jūrā"

# 3. Līmenis – dators šāvienus izdara plānveidīgi nevis uz labu laimi.

# Programmas autors: Vladislavs Babaņins

# Versija 1.0



import numpy

import random



"""

TIKA REALIZĒTS 2. LĪMENIS -  dators pats izvieto savus kuģus "jūrā". Katrs secīgi var izdarīt tikai vienu gājienu.
```

Uz ekrāna redzamas tikai lietotāja gājieni datora “jūrā” un lietotāja “jūra ar kuģiem” un datora izdarītiem gājieniem.

Dators trāpa nejauši, bet tikai tas vietas, kas vel nebīja trāpītas.

Dators nejauši izvieto savus kuģus (un tie kuģi būs novietoti pēc noteiktumiem bez pārklašanos).

Cilvēks var izvelētis vai novietot nejauši savus kuģus (un tie kuģi būs novietoti pēc noteiktumiem bez pārklašanos), vai pats novietot.

Kad novieto pats, tad katru reizi pārbauda vai tas pārklajās ar citu kuģi (vai to var vispār tā novietot). Ja nevār, tad mēģinām atkāl.

Dators uzvār tad, kad uz cilvēka (players_ships) galda ir "X" skaitā 20. (count_x(sv) funkcija noteic simbolu virknes "X" skaitu).

Cilvēks uzvār tad, kad uz viņa (player_shots) galda ir "X" skaitā 20. (count_x(sv) funkcija noteic simbolu virknes "X" skaitu).

20 tas ir skaitlis, ko iegūvam saskaitot visu kuģu garumus.

kuģi ir šādiem gārumiem

SSSS

SSS

SSS

SS

SS

SS

S

S

S

S

Ar "S" tiek norādīti kuģa daļas (vai pats kuģis)

"""

globālie mainīgie kuģu izvietojumiem

ships_arrangement = numpy.zeros((10, 10), dtype=int)

```
players_ships = numpy.zeros((10, 10), dtype=int)
ship_lengths = [4, 3, 3, 2, 2, 2, 1, 1, 1, 1]
player_shots = numpy.zeros((10, 10), dtype=int)
hits_and_misses = numpy.zeros((10, 10), dtype=int)
computer_shots = numpy.zeros((10, 10), dtype=int)



# ----------------------------------------------------------
```

def find_sum(saraksts): # Šājā gadījumā ir 20. Jā būtu vairāk kuģu, (vai citās kombinācijas), tad cits skaitlis. # find_sum(ship_lengths).

```
    # saraksts - saraksts, kurā meklēsim to visu elementu summa (visi elementi ir skaitļi).
    # Atgriež visu saraksta elementu summu.
    # Ir nepieciešāma programmā, lai noteiktu cik krustiņu (X) ir nepieciešāms, lai būtu uz galda, lai tiek paziņots par uzvaru vai zaudējumu.

    total_sum = 0
    for num in saraksts:
        total_sum = total_sum + num
    return total_sum



def count_x(sv):
    # Skaitā cik reizes parādas X simbolu virknē un atgriež to paradīšanas skaitu
    # sv - simbolu virkne (string)

    sk = 0
    for s in sv:
        if str(s) == "X":
            sk = sk + 1
    return sk
```

```
        def create_random_ships_arrangement_for_computer():  # "create_random_ships_arrangement_for_computer()", kas izveidos nejauši kuģu izvietojumu datoram.

            # Nejaušam kuģu izvietojumam datoram


            global ships_arrangement  # global ir izmantots datoram kuģu izvietojumam


            # izveidot tukšu 10x10 masīvu

            board = numpy.zeros((10, 10), dtype=int)


            # definēt kuģus un to garumus

            ships = [(4, 1), (3, 2), (2, 3), (1, 4)]  # Tiek definēti četri kuģi ar to garumiem un skaitiem. (garums, cik ir tas tipa kuģu)


            # Tiek veikta iterācija pāri katram kuģim ar to garumu un skaitu.

            for ship in ships:

               (length, count) = ship  # Kortežs. Katrs kuģis ir kortežs ar garumu un to skaitu).

               for i in range(count):  # Novietojam tik reizes, cik ir count (pēc nosācijumiem cik var novietot kuģus.

                  placed = False  # "placed" tiek definēts kā "False". (Nav pagaidām novietots).

                  while not placed:  # Kamēr kuģis nav novietots pareizi, tiek veikts while cikls.

                     # nejauši novietot kuģus

                     direction = random.randint(0, 1)  # vai nu 0 vai 1

                     if direction == 0:  # horizontāli h

                        x = random.randint(0, 9 - length + 1)

                        y = random.randint(0, 9)


                        # pārbauda, vai kuģis pārklājas ar citu kuģi

                        parklajums = False  # pēc noklusējuma domājam ka varam novietot (nepārklājas)

                        for j in range(max(0, y - 1), min(10, y + 2)):
```

```
                for k in range(max(0, x - 1), min(10, x + length + 1)):
                    if board[j, k] == 1:
                        parklajums = True  # tas nozīme kā nevarēsim novietot
            if not parklajums:  # Ja visos pārbaudījumos ir OK
                board[y, x:x + length] = 1
                placed = True  # Novietots

        else:  # vertikāli
            x = random.randint(0, 9)
            y = random.randint(0, 9 - length + 1)

            # pārbauda, vai kuģis pārklājas ar citu kuģi
            parklajums = False  # pēc noklusējuma domājam, ka varam novietot (nepārklājas)
            for j in range(max(0, y - 1), min(10, y + length + 1)):
                for k in range(max(0, x - 1), min(10, x + 2)):
                    if board[j, k] == 1:
                        parklajums = True  # tas nozīme, ka nevarēsim novietot
            if not parklajums:  # Ja visos pārbaudījumos ir OK
                board[y:y + length, x] = 1
                placed = True  # Novietots

    # iestatīt globālo mainīgo (tas datoram)
    ships_arrangement = board


def create_random_ships_arrangement_for_player():
    # Funkcija nejaušam kuģu izvietojumam cilvēkam.
    # Tas pats kā datoram, bet tikai cits globālais mainīgais
    # global players_ships - numpy 10x10 divdimensijas masīvs ar cilvēka kuģiem

    global players_ships
```

```
# izveidot tukšu 10x10 masīvu
board = numpy.zeros((10, 10), dtype=int)

# definēt kuģus un to garumus
ships = [(4, 1), (3, 2), (2, 3), (1, 4)]

for ship in ships:  # cikls ar sarakstu no kortēžiem
    (length, count) = ship
    for i in range(count):  # ejam cikla tik, cik ir to noteiktu kuģu (length, count)
        placed = False
        while not placed:
            # nejauši novietot kuģus
            direction = numpy.random.randint(2)
            if direction == 0:  # horizontāli
                x = numpy.random.randint(0, 10 - length + 1)
                y = numpy.random.randint(0, 10)

                # pārbauda, vai kuģis pārklājas ar citu kuģi
                parklajums = False  # pieņēmam kā neparklājas (karogs), ja kaut vienā pārbaudījuma pārklājas, tad nevaram novietot
                for j in range(max(0, y - 1), min(10, y + 2)):
                    for k in range(max(0, x - 1), min(10, x + length + 1)):
                        if board[j, k] == 1:
                            parklajums = True  # ja neizpildās, tad pārklājas un nevaram novietot
                if not parklajums:
                    board[y, x:x + length] = 1
                    placed = True
            else:  # vertikāli
                x = numpy.random.randint(0, 10)
                y = numpy.random.randint(0, 10 - length + 1)
```

```
                # pārbauda, vai kuģis pārklājas ar citu kuģi
                parklajums = False
                for j in range(max(0, y - 1), min(10, y + length + 1)):
                    for k in range(max(0, x - 1), min(10, x + 2)):
                        if board[j, k] == 1:
                            parklajums = True
                if not parklajums:
                    board[y:y + length, x] = 1
                    placed = True

        # iestatīt globālo mainīgo (cilvēkam)
        players_ships = board


    def player_hits_and_misses():
        # Izveido cilvēka (spēlētaju) glīto galda attēlojumu izmantojot string
        # global player_shots - 10x10 numpy divdimensijas masīvs, kurš reprezentē cilvēka gājienus (trāpijumus)
        # global ships_arrangement - 10x10 numpy divdimensijas masīvs, kurš reprezentē datora kuģu novietošanu (nejauši izveidotu)
        # global hits_and_misses - 10x10 numpy divdimensijas masīvs, kurš reprezentē cilvēka kuģu novietošanu (tas varētu būt vai nejauši izveidots, vai manuāli)

        global player_shots, ships_arrangement, hits_and_misses

        # Atjaunināt hits_and_miss dēli ar spēlētāja trāpijumiem
        for i in range(10):
            for j in range(10):
                if player_shots[i, j] == 1 and ships_arrangement[i, j] == 1:
                    hits_and_misses[i, j] = 1
                elif player_shots[i, j] == 1 and ships_arrangement[i, j] == 0:
```

```
                hits_and_misses[i, j] = -1


        # Spēles galda izveidošanas virknes attēlojumā
        board_str = "   A B C D E F G H I J\n"
        for i in range(10):
            if i != 9:
                board_str = board_str + " " + str(i + 1) + " "
            else:
                board_str = board_str + str(i + 1) + " "
            for j in range(10):
                if hits_and_misses[i][j] == 1:
                    board_str = board_str + "X "
                elif hits_and_misses[i][j] == -1:
                    board_str = board_str + "O "
                else:
                    board_str = board_str + ". "
            board_str = board_str + "\n"


        return board_str



    def check_overlap(players_ships, x, y, length, direction):
        # Pārbauda vai jaunais kuģis pārklājas ar kādu no eksistējošiem kuģiem uz galda. Funkcija cilvēkam.
        # Atgriež True vai False. True - pārklājas (overlaps). False - neparklājas var novietot.
        # players_ships - 10x10 divdimensijas masīvs ar cilvēka kuģiem
        # x - x koordināta (int skaitlis no 0 līdz 9)
        # y - y koordināta (int skaitlis no 0 līdz 9)
        # length - garums kuģiem
        # direction - tas ir str "h" vai "H", vai "v", vai "V"
```

```
if direction == "h" or direction == "H":  # horizontāls novietojums

    if x + length > 10:
        return True  # True == nedēr, nevar šādi novietot

    for i in range(x, x + length):
        if players_ships[y, i] == 1:
            return True
        if y > 0 and players_ships[y - 1, i] == 1:
            return True
        if y < 9 and players_ships[y + 1, i] == 1:
            return True
        if i > 0 and players_ships[y, i - 1] == 1:
            return True
        if i < 9 and players_ships[y, i + 1] == 1:
            return True
        if y > 0 and i > 0 and players_ships[y - 1, i - 1] == 1:
            return True
        if y < 9 and i < 9 and players_ships[y + 1, i + 1] == 1:
            return True
        if y > 0 and i < 9 and players_ships[y - 1, i + 1] == 1:
            return True
        if y < 9 and i > 0 and players_ships[y + 1, i - 1] == 1:
            return True

elif direction == "v" or direction == "V":  # vertikāls novietojums

    if y + length > 10:
        return True

    for i in range(y, y + length):
```

```
            if players_ships[i, x] == 1:
                return True
            if x > 0 and players_ships[i, x - 1] == 1:
                return True
            if x < 9 and players_ships[i, x + 1] == 1:
                return True
            if i > 0 and players_ships[i - 1, x] == 1:
                return True
            if i < 9 and players_ships[i + 1, x] == 1:
                return True
            if x > 0 and i > 0 and players_ships[i - 1, x - 1] == 1:
                return True
            if x < 9 and i < 9 and players_ships[i + 1, x + 1] == 1:
                return True
            if x > 0 and i < 9 and players_ships[i + 1, x - 1] == 1:
                return True
            if x < 9 and i > 0 and players_ships[i - 1, x + 1] == 1:
                return True

        return False


    def place_player_ships():
        # Cilvēka (spēlētaja) kuģu novietošanai

        random_placement = input("Vai vēlaties nejauši novietot savus kuģus? (j - jā / n - nē): ")  # Vai gribāt random novietošanu?

        # Tas ir lai būtu lielāka izvēle starp j, ja, jā, Ja, Jā, JA, JĀ, n, ne, nē, Ne, Nē, NE, NĒ.
        while (random_placement != "j" and random_placement != "n" and random_placement != "J" and random_placement != "jā"
```

```
                and random_placement != "ja" and random_placement != "Jā" and random_placement
!= "Ja" and random_placement != "JA"

                and random_placement != "JĀ" and random_placement != "N" and random_placement
!= "ne" and random_placement != "nē"

                and random_placement != "Ne" and random_placement != "Nē" and
random_placement != "NE" and random_placement != "NĒ"):

            random_placement = input("Ķļūda! Vai vēlaties nejauši novietot savus kuģus? (j - jā / n -
nē): ")


        # Pārbaudam vai izvēle starp j, J, ja, jā, Ja, Jā, JA, JĀ tika izdarīta.

        if (random_placement == "j" or random_placement == "J" or random_placement == "jā" or
random_placement == "ja"

                or random_placement == "Jā" or random_placement == "Ja" or random_placement ==
"JA" or random_placement == "JĀ"):

            create_random_ships_arrangement_for_player()  # Ja tika izvelēts j, ja, jā, Ja, Jā, JA, JĀ
(viens no tiem), tad izveidosim cilvēka kuģus nejauši


        # Pārbaudam vai izvēle starp n, ne, nē, Ne, Nē, NE, NĒ tika izdarīta.

        if (random_placement == "n" or random_placement == "N" or random_placement == "ne"
or random_placement == "nē"

                or random_placement == "Ne" or random_placement == "Nē" or random_placement
== "NE" or random_placement == "NĒ"):


            # Ja tika izvelēts n, N, ne, nē, Ne, Nē, NE, NĒ  (viens no tiem), tad manuāli veidosim kuģu
novietojumu un kātru reizi pārbaudīsim vai tur varam to novietot.

            for length in ship_lengths:

                placed = False


                while not placed:

                    print("Kuģa novietošana ar garumu", length)

                    x = get_input_x()  # paprāsam x koordinātu (A, B, C, D, E, F G, H, I, J (0-9))

                    y = get_input_y()

                    direction = input("Ievadiet virzienu (h - horizontāli, v - vertikāli): ")
```

```
            if direction == 'h' or direction == 'H':

                if check_overlap(players_ships, x, y, length, direction):
                    print("Šis kuģis pārklājas ar citu kuģi! Lūdzu, izvēlieties citu atrašanās vietu!")
                    continue

                if not check_overlap(players_ships, x, y, length, direction):
                    players_ships[y, x:x + length] = 1
                    print(players_ships_board())
                    placed = True

                else:
                    print("Šis kuģis pārklājas ar citu kuģi! Lūdzu, izvēlieties citu atrašanās vietu!")

            elif direction == 'v' or direction == 'V':

                if check_overlap(players_ships, x, y, length, direction):
                    print("Šis kuģis pārklājas ar citu kuģi! Lūdzu, izvēlieties citu atrašanās vietu!")
                    continue

                if not check_overlap(players_ships, x, y, length, direction):
                    players_ships[y:y + length, x] = 1
                    print(players_ships_board())
                    placed = True

                else:
                    print("Šis kuģis pārklājas ar citu kuģi! Lūdzu, izvēlieties citu atrašanās vietu!")

            else:
                print("Nederīgs virziens! Lūdzu, ievadiet “h” horizontālam kuģu novietojumam vai
“v” vertikālam kuģu noveitojumam.")
```

```
def get_input_x():
    # Paprasīt ievādīt x koordinātas, kāmer nav ievādītas no A līdz J str.
    # Pārkonvertēt A-J kā skaitļi no 0 līdz 9.
    # Atgriež skaitli num no 0 līdz 9 (num - intervālā 0 līdz 9)

    while True:
        try:
            symbol = input("Ievadiet koordinātu A-J: ")

            # Var arī izmantot match case.
            # Neizmantoju match case, jo nevisas Python versijas tas strāda.
            if symbol == "A" or symbol == "a":
                num = 0
            elif symbol == "B" or symbol == "b":
                num = 1
            elif symbol == "C" or symbol == "c":
                num = 2
            elif symbol == "D" or symbol == "d":
                num = 3
            elif symbol == "E" or symbol == "e":
                num = 4
            elif symbol == "F" or symbol == "f":
                num = 5
            elif symbol == "G" or symbol == "g":
                num = 6
            elif symbol == "H" or symbol == "h":
                num = 7
            elif symbol == "I" or symbol == "i":
                num = 8
```

```
            elif symbol == "J" or symbol == "j":
                num = 9
            else:
                print("Nekorekta ievāde. Ievadiet koordinātas!")
                raise ValueError

            return num

        except:
            pass


def get_input_y():
    # Paprasīt ievādīt y koordinātas, kāmer nav ievādītas no 1 līdz 10 int.
    # Atgriež ievādīto skaitli -1 (num-1 intervālā 0 līdz 9)

    while True:
        try:
            num = int(input("Ievadiet koordinātu (1-10): "))
            num = num - 1
            if num >= 0 and num <= 9:
                return num
            else:
                print("Nekorekta ievāde. Ievadiet koordinātu (1-10): ")
        except ValueError:
            print("Nekorekta ievāde. Ievadiet koordinātu (1-10):")


def player_turn():
    # Cilvēka gājiens
```

```
    # global ships_arrangement - 10x10 numpy divdimensijas masīvs, kurš reprezentē datora kuģu novietošanu (nejauši izveidotu)
    # global players_ships - 10x10 numpy divdimensijas masīvs, kurš reprezentē cilvēka kuģu novietošanu (tas varētu būt vai nejauši izveidots, vai manuāli)
    # global player_shots - 10x10 numpy divdimensijas masīvs, kurš reprezentē cilvēka gājienus (trāpijumus)

    global ships_arrangement, players_ships, player_shots

    # Paprāsit lietotājam ievādit koordinātas
    x = get_input_x()  # x koordināta
    y = get_input_y()  # y koordināta

    # Pārbaudām vai tas trāpis vai garām un izmainām vērtības matricas (mainām vērtības globālajos divdimensijas masīviem)
    if ships_arrangement[y][x] == 1:  # Ja ir kuģis šajā "šuna"
        print("Jūs trāpijat datora kuģi!")
        ships_arrangement[y][x] = 1
        player_shots[y][x] = 1
    else:
        print("Garām!\n")
        ships_arrangement[y][x] = 0
        player_shots[y][x] = 1

    print(player_hits_and_misses())  # izvādīt galdiņu ar cilvēka gājieniem


def computer_turn():
    # Datora gājiens
    # global players_ships - numpy masīvs 10x10 ar cilvēka kuģiem
    # computer_shots - numpy masīvs 10x10 ar datora gājieniem
```

```
    global players_ships, computer_shots


    # paņemam nejaušus koordinātas kā kortēžu coords([x,y]), no 0 līdz 9. (programmā ir no 0 līdz 9, cilvēkam tas izvādīts kā no 1 līdz 10)
    coords = tuple(random.randint(0, 9) for i in range(2))


    while computer_shots[coords[1]][coords[0]] == -1 or computer_shots[coords[1]][coords[0]] == -2:  # X vai O
        coords = tuple(random.randint(0, 9) for i in range(2))  # kāmer nav izšauts tāda vietā, kur tas vēl nebīja


    if coords[0] == 0:  # var izmantot match case, bet ne visiem Python vērsijas tas strādā, tāpēc ar elif
        symbol = "A"
    elif coords[0] == 1:  # symbols, lai izvadītu to glīti uz ekrāna
        symbol = "B"
    elif coords[0] == 2:  # 0 -> A, 1 -> B, 2 -> C, 3 -> D, 4 -> E, 5 -> F, 6 -> G, 7 -> H, 8 -> I, 9 -> J
        symbol = "C"
    elif coords[0] == 3:
        symbol = "D"
    elif coords[0] == 4:
        symbol = "E"
    elif coords[0] == 5:
        symbol = "F"
    elif coords[0] == 6:
        symbol = "G"
    elif coords[0] == 7:
        symbol = "H"
    elif coords[0] == 8:
        symbol = "I"
    elif coords[0] == 9:
        symbol = "J"
```

```
        # Pārbaudam vai dators trāpija  Jūsu kuģi vai nē un izvadam to un pamainam vērtības global mainīgajos.
        if players_ships[coords[1]][coords[0]] == 1:  # ja ir kuģis tad tur ir 1
            print("Dators trāpija Jūsu kuģi ar koordinātam (" + symbol + ", " + str(coords[1] + 1) + ")")
            players_ships[coords[1]][coords[0]] = -1
            computer_shots[coords[1]][coords[0]] = -1  # kad trāpija, tad -1

        else:
            print("Dators trāpija garām! Datora šaviena koordinātas (" + symbol + ", " + str(coords[1] + 1) + ")")
            players_ships[coords[1]][coords[0]] = -2
            computer_shots[coords[1]][coords[0]] = -2  # kad garām, tad -2


    def players_ships_board():
        # Glīti ar koordinātam izvāda cilvēka kuģus uz ekrāna
        # Tikai atgriež to galdu, bet neprintē.
        # Lai printētu vajag print(players_ships_board())
        # global players_ships - numpy masīvs 10x10 ar cilvēka kuģiem

        global players_ships

        board = "   A B C D E F G H I J \n"  # koordinātas

        # glītai izvadīšanai
        k = 1
        for row in players_ships:
            if k != 10:
                board += " " + str(k) + " "
            else:
                board += str(k) + " "
```

```
            for cell in row:
                if cell == 1:  # S - tur ir kuģis (kuģa daļa)
                    board += "S "
                elif cell == -1:  # X - trāpita kuģa daļa
                    board += "X "
                elif cell == -2:  # O - garām (nav trāpīts)
                    board += "O "
                else:
                    board += ". "  # punkts uz galda
            k = k + 1
            board += "\n"

        return board



    def play_game():
        # Funckija spēles sākumam (kad jau notiek trāpīšanas un gājieni)
        # ships_arrangement -  10x10 divdimensiju masīvs ar datora kuģiem
        # players_ships - 10x10 divdimensiju masīvs ar cilvēka kuģiem
        # player_shots - 10x10 divdimensiju masīvs ar cilvēka izdarītiem gājieniem

        global ships_arrangement, players_ships, player_shots

        print("Jūsu kūģi:")
        print(players_ships_board())  # Izvadīt cilvēka galdiņu ar kuģiem

        while True:
            player_turn()  # cilvēka gājiens

            summa = find_sum(ship_lengths)  # summa ir 20 ar šadiem noteikumiem (20 - cik ir X
maksimāli, cik punktu būs novietoti kuģi - visu kuģu gārumu summa)
```

```
            if count_x(player_hits_and_misses()) == summa:  # Ja cilvēks trāpija visus 20 (parādas 20 X uz galda), tad cilvēks uzvarēja
                # Testēšanai: Ja summa == 1, tad pēc pirmas trāpīšanas (ir viens X uz galda) paradīsies tas, ka cilvēks uzvarēja.
                print("Jūs uzvarējat! Apsveicam!")
                quit()  # programma beidzās

            print("Datora gājiens...")
            computer_turn()  # procedūra datora gājienam

            # pārbauda, vai visi spēlētāja (cilvēka) kuģi ir iznīcināti
            if count_x(players_ships_board()) == summa:  # Testēšanai summu var pamainīt (tā ir 20 tagad).
                # Testēšanai: Ja summa == 1, tad pēc pirmas trāpīšanas (ir viens X uz galda) paradīsies tas, ka dators uzvarēja.
                print("")
                print(players_ships_board())  # Izvadīt cilvēka galdiņu ar kuģiem
                print("Jūs zaudējat! Dators uzvarēja!")
                quit()  # programma beidzās

            # Izvadīt cilvēka galdiņu ar kuģiem
            print("Jūsu kuģi:")
            print(players_ships_board())


    '''
    # TESTIEM
    def print_random_ships():  # Tika izmantota testēšanai. Print datora kuģus.
        global ships_arrangement

        # drukāt kolonnas koordinātas
```

```
        print("   A B C D E F G H I J")


        # drukas "spēles galdiņu" uz ekrāna un vertikāļu koordinātas
        for i in range(10):
            if i != 9:
                print(" " + str(i + 1) + " ", end="")  # end="" nozīme kā nav pāreju uz jauno rindu pēc print("", end=""). Jo parasti print izdāra arī kā /n
            else:
                print(str(i + 1) + " ", end="")


            for j in range(10):
                if ships_arrangement[i][j] == 1:
                    print("S ", end="")
                else:
                    print(". ", end="")
            print("")
    '''



    # ----------------------------------------------------------
    # Galvenā programmas daļa
    # ----------------------------------------------------------
    '''
    globālie mainīgie programmā
    ships_arrangement
    players_ships
    ship_lengths
    player_shots
    hits_and_misses
    computer_shots
    '''
```

create_random_ships_arrangement_for_computer() # Funkcija nejaušam kuģu izvietojumam datoram

print_random_ships() # Testēšanai (izvada datora kuģus)

place_player_ships() # Funkcija kuģu izvietojumam cilvēkam (jautās, vai nejauši izvietot, vai manuāli)

play_game() # gājieni

Testa piemēri:

1) Testa piemēri tiek parādīti video: https://youtu.be/lI3G3xCvc_o
2) Testā piemērus skatīt failos testa_piemērs_1.txt, testa_piemērs_2.txt, testa_piemērs_3.txt, testa_piemērs_4.txt. Šeit nepieliku testa piemērus, jo tad tas aizņemtu pārāk daudz atmiņas un lappuses.

```
Vai vēlaties nejauši novietot savus kuģus? (j - jā / n - nē): fa
Kļūda! Vai vēlaties nejauši novietot savus kuģus? (j - jā / n - nē): ja
Jūsu kūģi:
   A B C D E F G H I J
 1 . . . . . . . . . .
 2 . . S . . S S S S .
 3 . . S . . . . . . .
 4 . . S . . . S . . S
 5 . . . . . . . . . .
 6 . . . . . S S S . .
 7 . . . . . . . . . .
 8 . S . . S . . . . .
 9 . . . . . . . . . .
10 S S . S S . . S S .

Ievadiet koordinātu A-J: asfa
Nekorekta ievāde. Ievadiet koordinātas!
Ievadiet koordinātu A-J: asfaf
Nekorekta ievāde. Ievadiet koordinātas!
Ievadiet koordinātu A-J: asfa
Nekorekta ievāde. Ievadiet koordinātas!
Ievadiet koordinātu A-J: A
Ievadiet koordinātu (1-10): adas
Nekorekta ievāde. Ievadiet koordinātu (1-10):
Ievadiet koordinātu (1-10): dasd
Nekorekta ievāde. Ievadiet koordinātu (1-10):
Ievadiet koordinātu (1-10): ads
Nekorekta ievāde. Ievadiet koordinātu (1-10):
Ievadiet koordinātu (1-10): 11
Nekorekta ievāde. Ievadiet koordinātu (1-10):
Ievadiet koordinātu (1-10): 2
Garām!

   A B C D E F G H I J
 1 . . . . . . . . . .
 2 O . . . . . . . . .
 3 . . . . . . . . . .
 4 . . . . . . . . . .
 5 . . . . . . . . . .
 6 . . . . . . . . . .
 7 . . . . . . . . . .
 8 . . . . . . . . . .
 9 . . . . . . . . . .
10 . . . . . . . . . .

Datora gājiens...
Dators trāpija garām! Datora šaviena koordinātas (D, 6)
Jūsu kuģi:
   A B C D E F G H I J
 1 . . . . . . . . . .
 2 . . S . . S S S S .
 3 . . S . . . . . . .
```

3)

```
Vai vēlaties nejauši novietot savus kuģus? (j - jā / n - nē): ne
Kuģa novietošana ar garumu 4
Ievadiet koordinātu A-J: a
Ievadiet koordinātu (1-10): 1
Ievadiet virzienu (h - horizontāli, v - vertikāli): h
   A B C D E F G H I J
 1 S S S S . . . . . .
 2 . . . . . . . . . .
 3 . . . . . . . . . .
 4 . . . . . . . . . .
 5 . . . . . . . . . .
 6 . . . . . . . . . .
 7 . . . . . . . . . .
 8 . . . . . . . . . .
 9 . . . . . . . . . .
10 . . . . . . . . . .

Kuģa novietošana ar garumu 3
Ievadiet koordinātu A-J: f
Ievadiet koordinātu (1-10): 2
Ievadiet virzienu (h - horizontāli, v - vertikāli): h
   A B C D E F G H I J
 1 S S S S . . . . . .
 2 . . . . . S S S . .
 3 . . . . . . . . . .
 4 . . . . . . . . . .
 5 . . . . . . . . . .
 6 . . . . . . . . . .
 7 . . . . . . . . . .
 8 . . . . . . . . . .
 9 . . . . . . . . . .
10 . . . . . . . . . .

Kuģa novietošana ar garumu 3
Ievadiet koordinātu A-J: f
Ievadiet koordinātu (1-10): 1
Ievadiet virzienu (h - horizontāli, v - vertikāli): h
Šis kuģis pārklājas ar citu kuģi! Lūdzu, izvēlieties citu atrašanās vietu!
Kuģa novietošana ar garumu 3
Ievadiet koordinātu A-J: f
Ievadiet koordinātu (1-10): 2
Ievadiet virzienu (h - horizontāli, v - vertikāli): v
Šis kuģis pārklājas ar citu kuģi! Lūdzu, izvēlieties citu atrašanās vietu!
Kuģa novietošana ar garumu 3
Ievadiet koordinātu A-J: 10
Nekorekta ievāde. Ievadiet koordinātas!
Ievadiet koordinātu A-J: j
Ievadiet koordinātu (1-10): 10
Ievadiet virzienu (h - horizontāli, v - vertikāli): h
Šis kuģis pārklājas ar citu kuģi! Lūdzu, izvēlieties citu atrašanās vietu!
Kuģa novietošana ar garumu 3
Ievadiet koordinātu A-J: i
```

4)

```
 8 X X X O O X O O . O
 9 O O O O O O . O . O
10 O O O X O O . O O O

Ievadiet koordinātu A-J: d
Ievadiet koordinātu (1-10): 6
Garām!

   A B C D E F G H I J
 1 X O O . . O O O . O
 2 O O X X O O X O . X
 3 O . O . . O X O O X
 4 X . O O . O X O . O
 5 O . O O . O X . O O
 6 O O O O . O O O X O
 7 O O O X . O O O X O
 8 O O O . . O O O X O
 9 X O O X X X O O O O
10 O . . . . O O O O X

Datora gājiens...
Dators trāpija garām! Datora šaviena koordinātas (I, 8)
Jūsu kuģi:
   A B C D E F G H I J
 1 O S O O . O O X . O
 2 O . O O X . . O O O
 3 O X O O X O O X O X
 4 O X O O O O O X O X
 5 . O O O . S O X O .
 6 S O O O O S O O O O
 7 O O O O O X O O O O
 8 X X X O O X O O O O
 9 O O O O O O . O . O
10 O O O X O O . O O O

Ievadiet koordinātu A-J: e
Ievadiet koordinātu (1-10): 7
Jūs trāpijat datora kuģi!
   A B C D E F G H I J
 1 X O O . . O O O . O
 2 O O X X O O X O . X
 3 O . O . . O X O O X
 4 X . O O . O X O . O
 5 O . O O . O X . O O
 6 O O O O . O O O X O
 7 O O O X X O O O X O
 8 O O O . . O O O X O
 9 X O O X X X O O O O
10 O . . . . O O O O X

Jūs uzvarējat! Apsveicam!
```