Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра інформатики

Звіт з Практичного завдання №3

По предмету: «Штучний інтелект для інформаційних технологій»

Виконав:

студент групи ІТІНФ-20-1

Самченко С. О.

Харків 2023

**Мета**: ознайомитись з евристичними функціями на прикладі задачі гри у вісім, що дозволяє найкраще продемонструвати характерні особливості всіх евристичних функцій в цілому. Дослідити особливості використання генетичних алгоритмів для вирішення задачі гри в вісім.

1. **Задана ціль №1:**

goal = [1, 2, 3, 4, 5, 6, 7, 8, 0]

Реалізація евристики:

# Heuristics for 8 Puzzle Problem

import math

def linear(node):

    return sum([1 if node.state[i] != goal[i] else 0 for i in range(8)])

def manhattan(node):

    state = node.state

    index\_goal = {0:[2,2], 1:[0,0], 2:[0,1], 3:[0,2], 4:[1,0], 5:[1,1], 6:[1,2], 7:[2,0], 8:[2,1]}

    index\_state = {}

    index = [[0,0], [0,1], [0,2], [1,0], [1,1], [1,2], [2,0], [2,1], [2,2]]

    x, y = 0, 0

    for i in range(len(state)):

        index\_state[state[i]] = index[i]

    mhd = 0

    for i in range(8):

        for j in range(2):

            mhd = abs(index\_goal[i][j] - index\_state[i][j]) + mhd

    return mhd

def sqrt\_manhattan(node):

    state = node.state

    index\_goal = {0:[2,2], 1:[0,0], 2:[0,1], 3:[0,2], 4:[1,0], 5:[1,1], 6:[1,2], 7:[2,0], 8:[2,1]}

    index\_state = {}

    index = [[0,0], [0,1], [0,2], [1,0], [1,1], [1,2], [2,0], [2,1], [2,2]]

    x, y = 0, 0

    for i in range(len(state)):

        index\_state[state[i]] = index[i]

    mhd = 0

    for i in range(8):

        for j in range(2):

            mhd = (index\_goal[i][j] - index\_state[i][j])\*\*2 + mhd

    return math.sqrt(mhd)

def max\_heuristic(node):

    score1 = manhattan(node)

    score2 = linear(node)

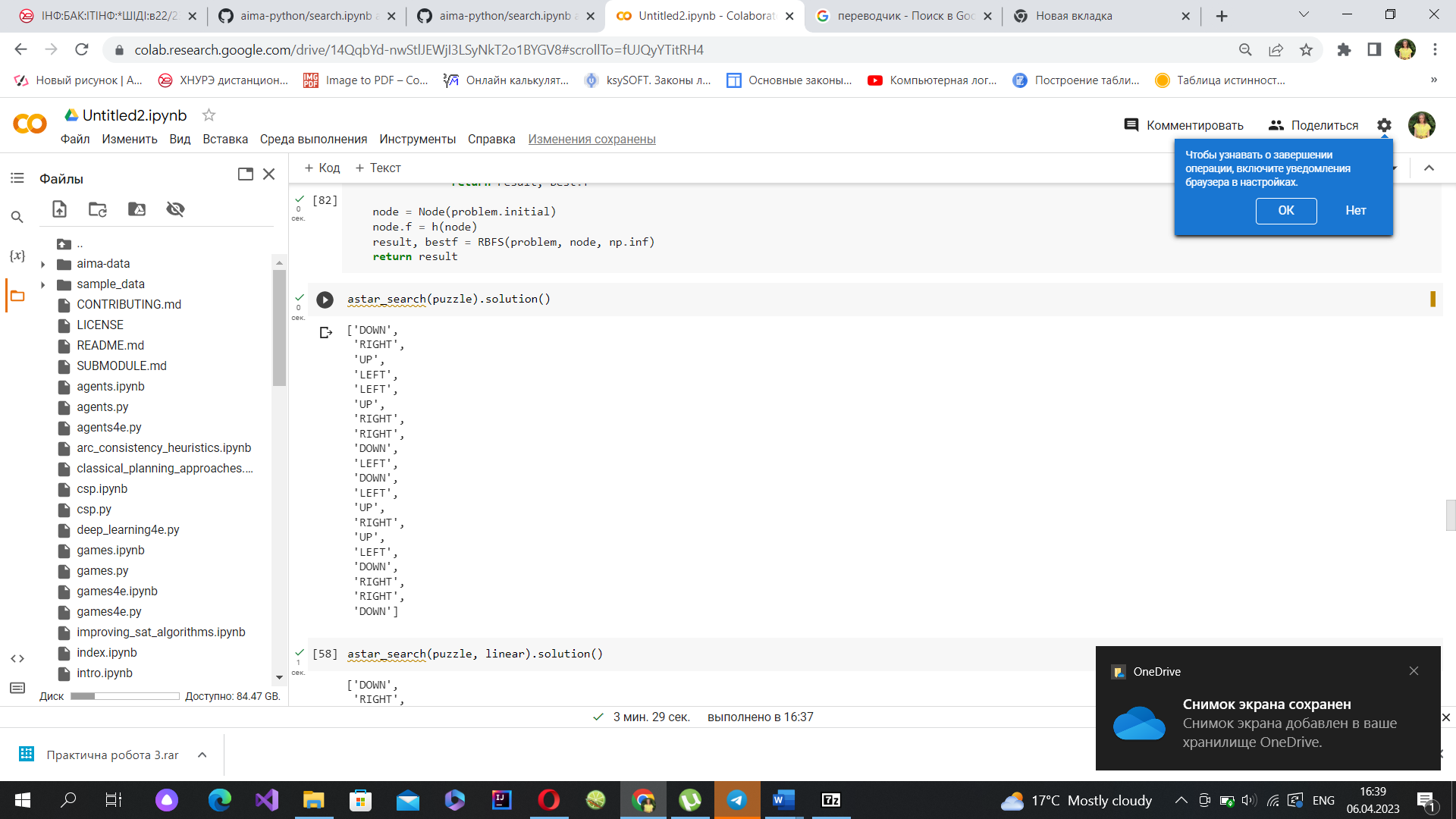
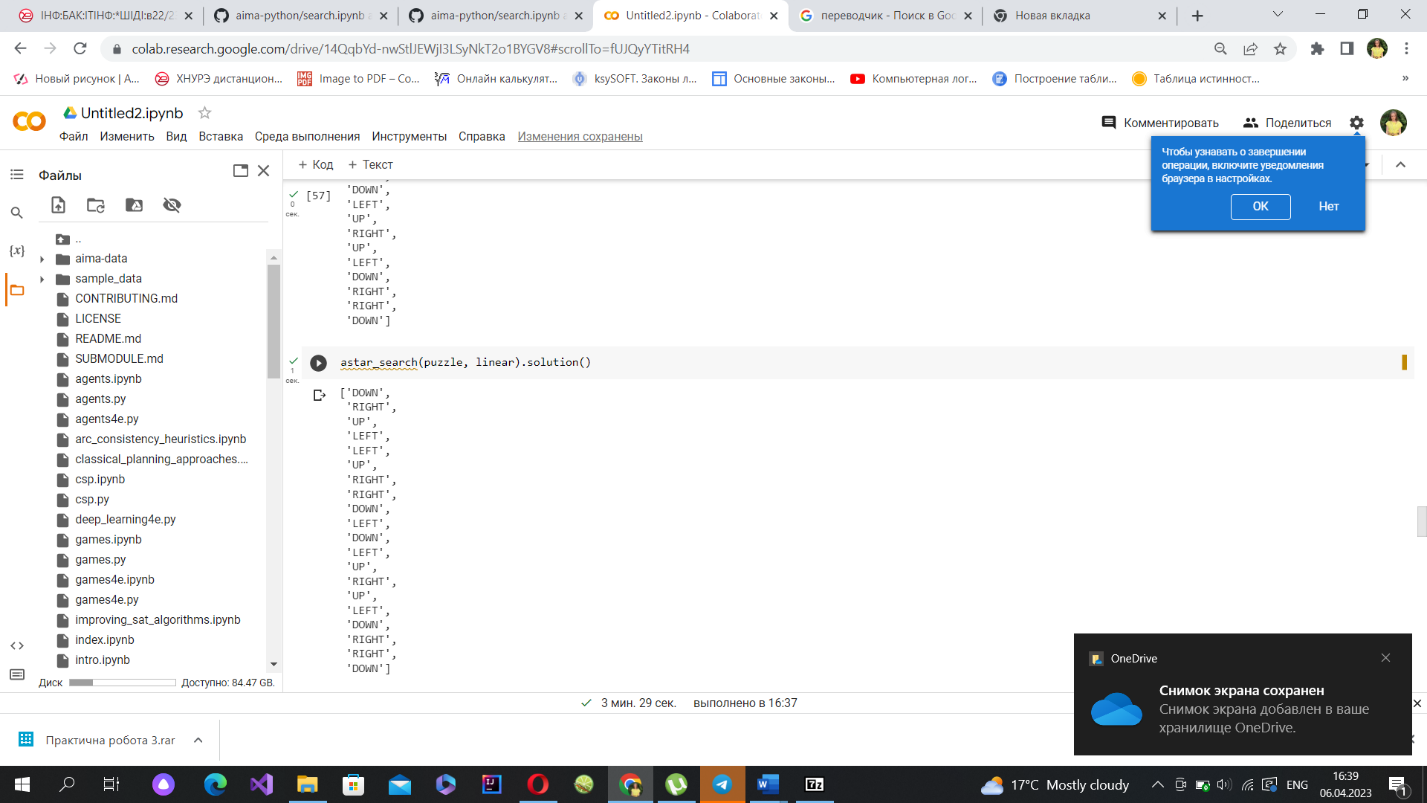
    return max(score1, score2)

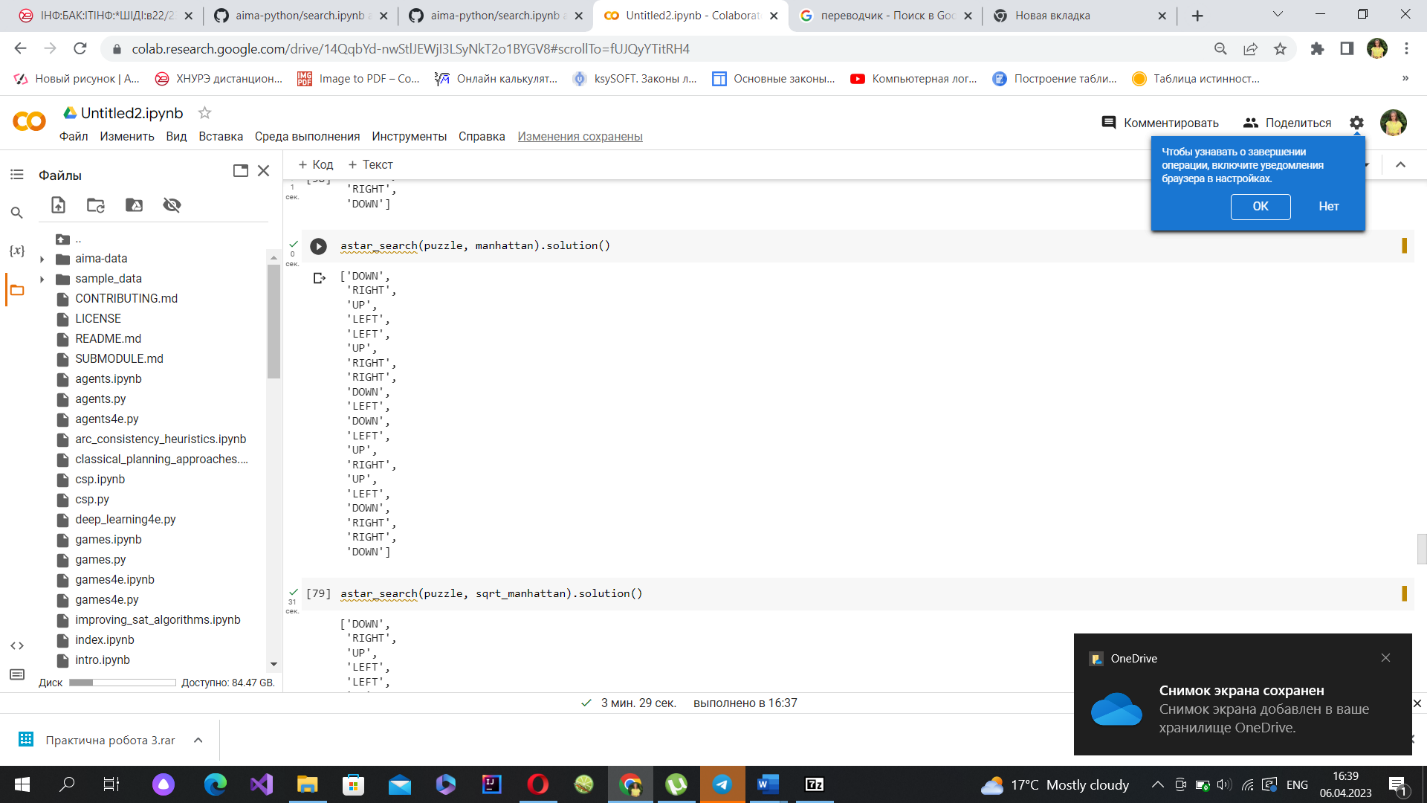
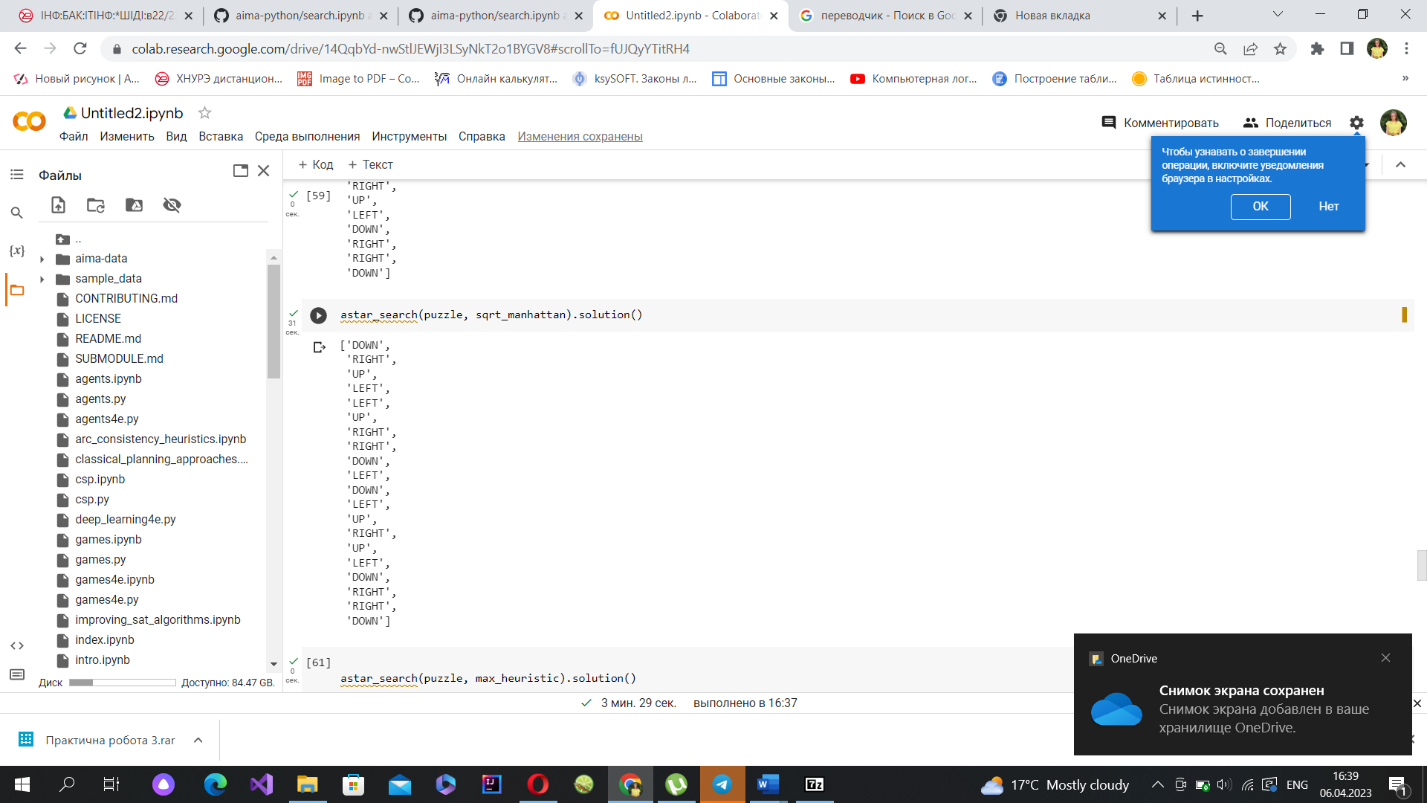
Перевірка можливості вирішити завдання:

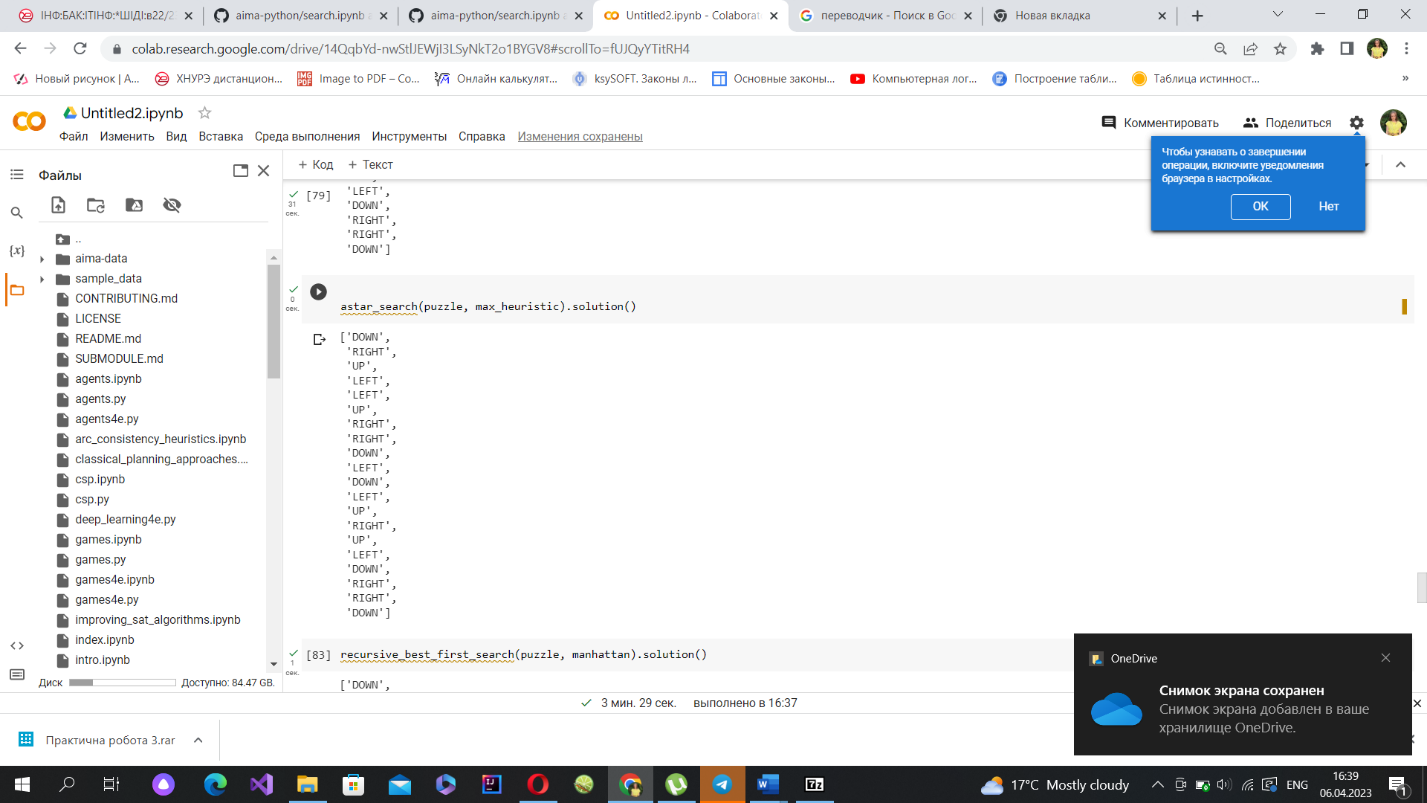
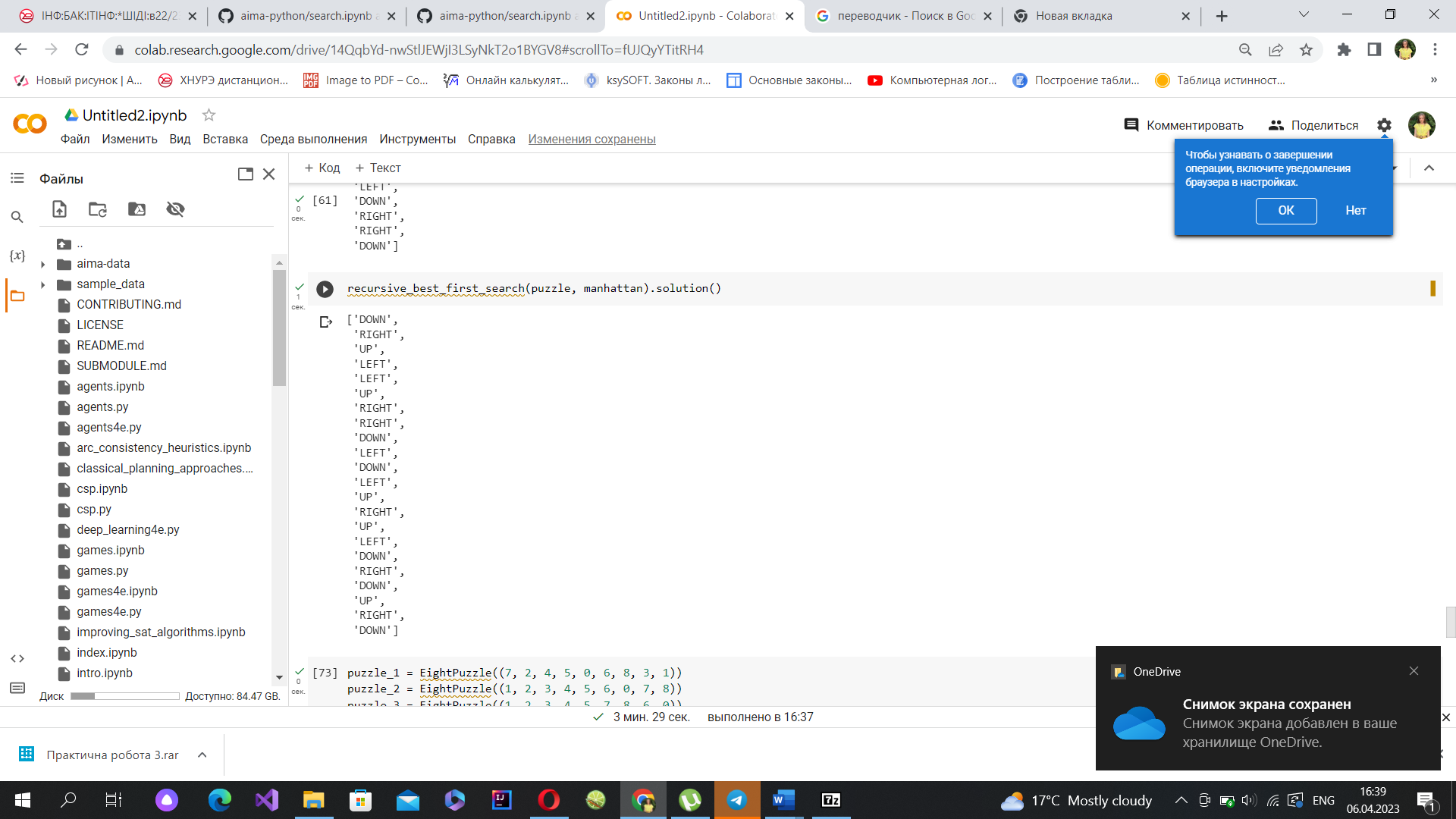
# Solving the puzzle

puzzle = EightPuzzle((7, 2, 4, 5, 0, 6, 8, 3, 1))

puzzle.check\_solvability((7, 2, 4, 5, 0, 6, 8, 3, 1)) # checks whether the initialized configuration is solvable or not

puzzle\_1 = EightPuzzle((7, 2, 4, 5, 0, 6, 8, 3, 1))

puzzle\_2 = EightPuzzle((1, 2, 3, 4, 5, 6, 0, 7, 8))

puzzle\_3 = EightPuzzle((1, 2, 3, 4, 5, 7, 8, 6, 0))

**1**

%%timeit

astar\_search(puzzle\_1)

astar\_search(puzzle\_2)

astar\_search(puzzle\_3)

Результат:

1.24 s ± 198 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)

**2**

%%timeit

astar\_search(puzzle\_1, linear)

astar\_search(puzzle\_2, linear)

astar\_search(puzzle\_3, linear)

Результат:

1.24 s ± 287 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)

**3**

%%timeit

astar\_search(puzzle\_1, manhattan)

astar\_search(puzzle\_2, manhattan)

astar\_search(puzzle\_3, manhattan)

Результат:

62.1 ms ± 15.7 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)

**4**

%%timeit

astar\_search(puzzle\_1, sqrt\_manhattan)

astar\_search(puzzle\_2, sqrt\_manhattan)

astar\_search(puzzle\_3, sqrt\_manhattan)

Результат:

28.7 s ± 1.24 s per loop (mean ± std. dev. of 7 runs, 1 loop each)

**5**

%%timeit

astar\_search(puzzle\_1, max\_heuristic)

astar\_search(puzzle\_2, max\_heuristic)

astar\_search(puzzle\_3, max\_heuristic)

Результат:

50.6 ms ± 1.05 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)

**6**

%%timeit

recursive\_best\_first\_search(puzzle\_1, linear)

recursive\_best\_first\_search(puzzle\_2, linear)

recursive\_best\_first\_search(puzzle\_3, linear)

Результат:

26.2 s ± 437 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)

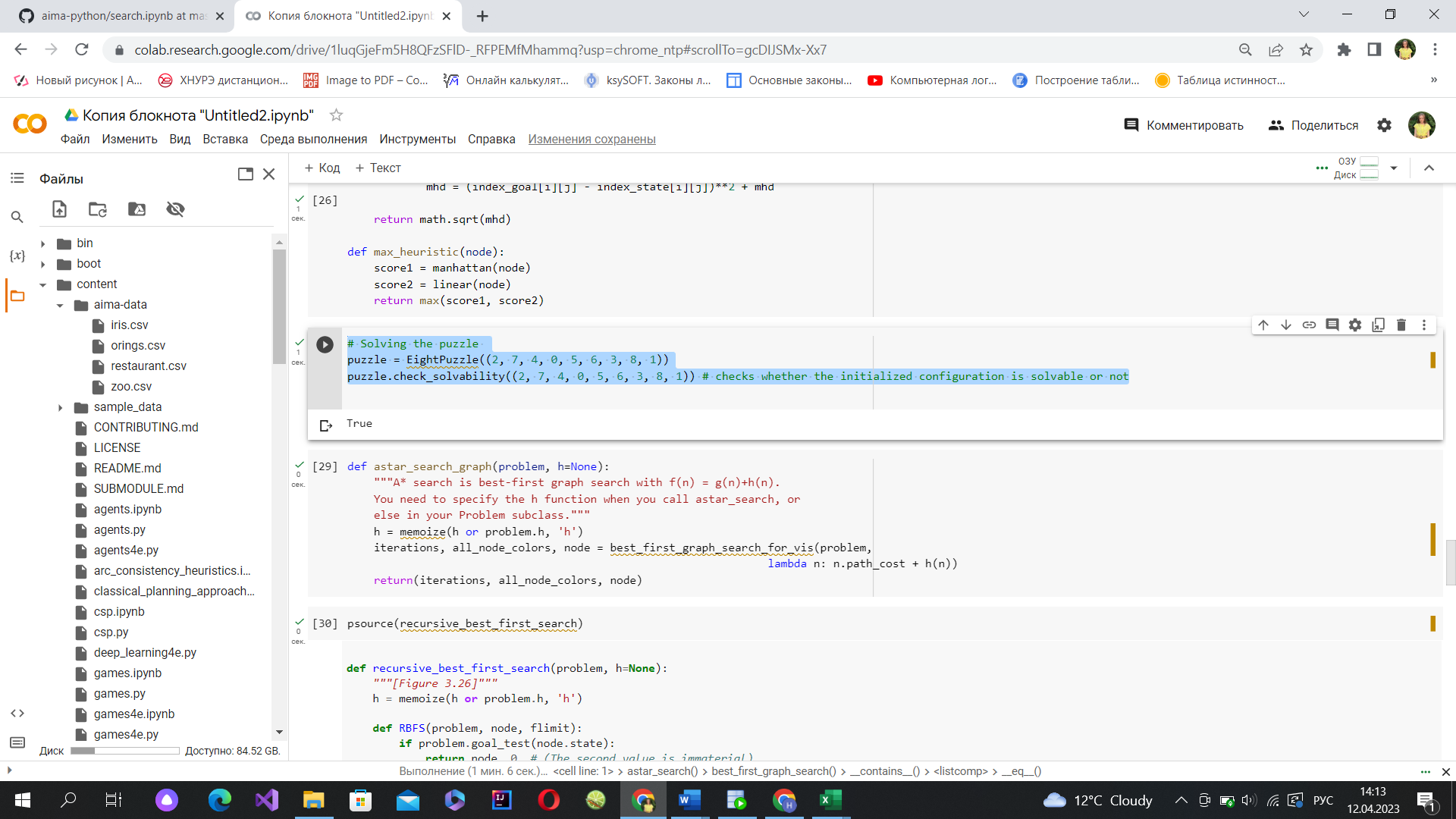
1. **Задана ціль №2:**

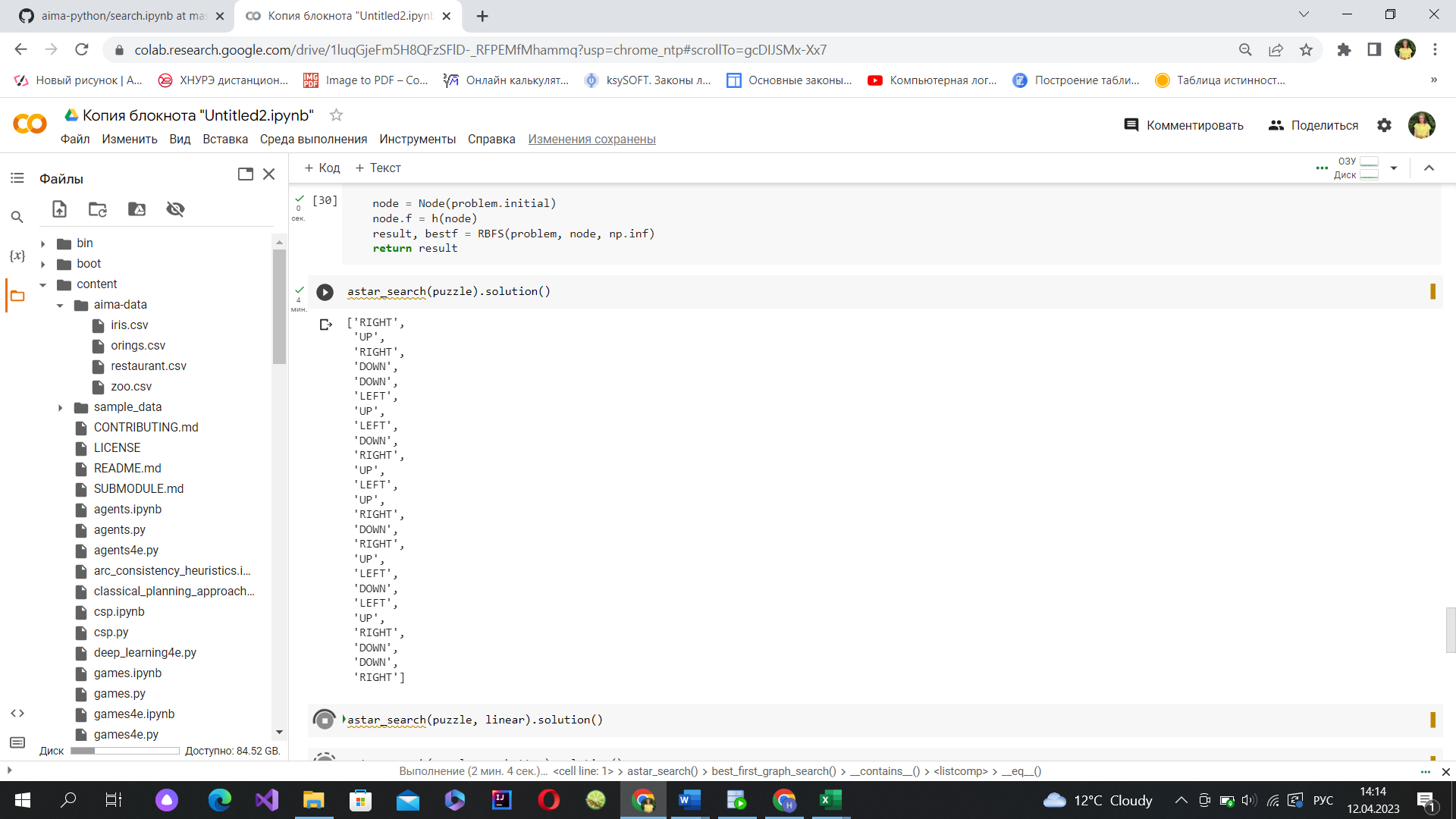
goal = [1, 2, 3, 4, 5, 6, 7, 8, 0]

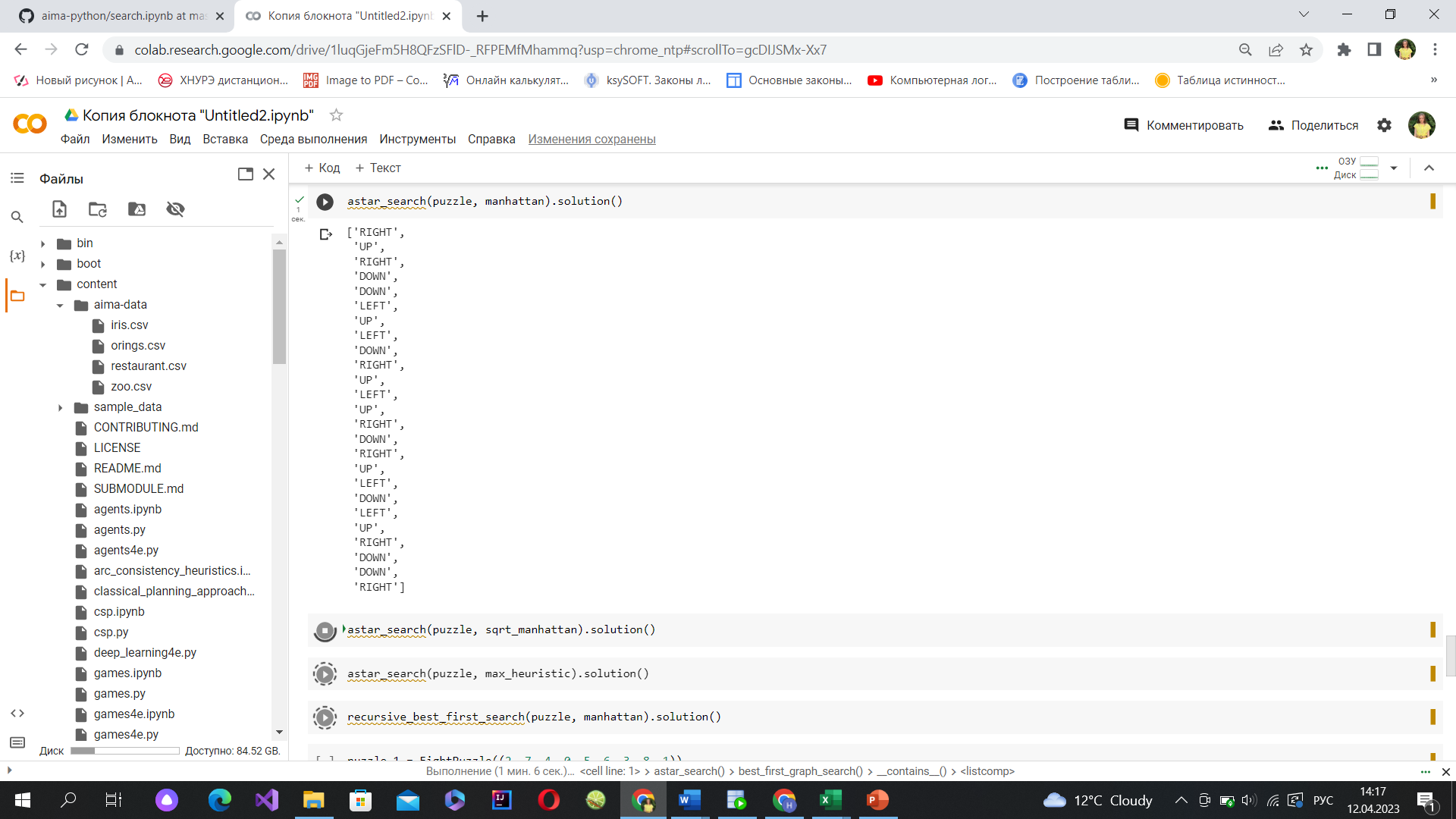
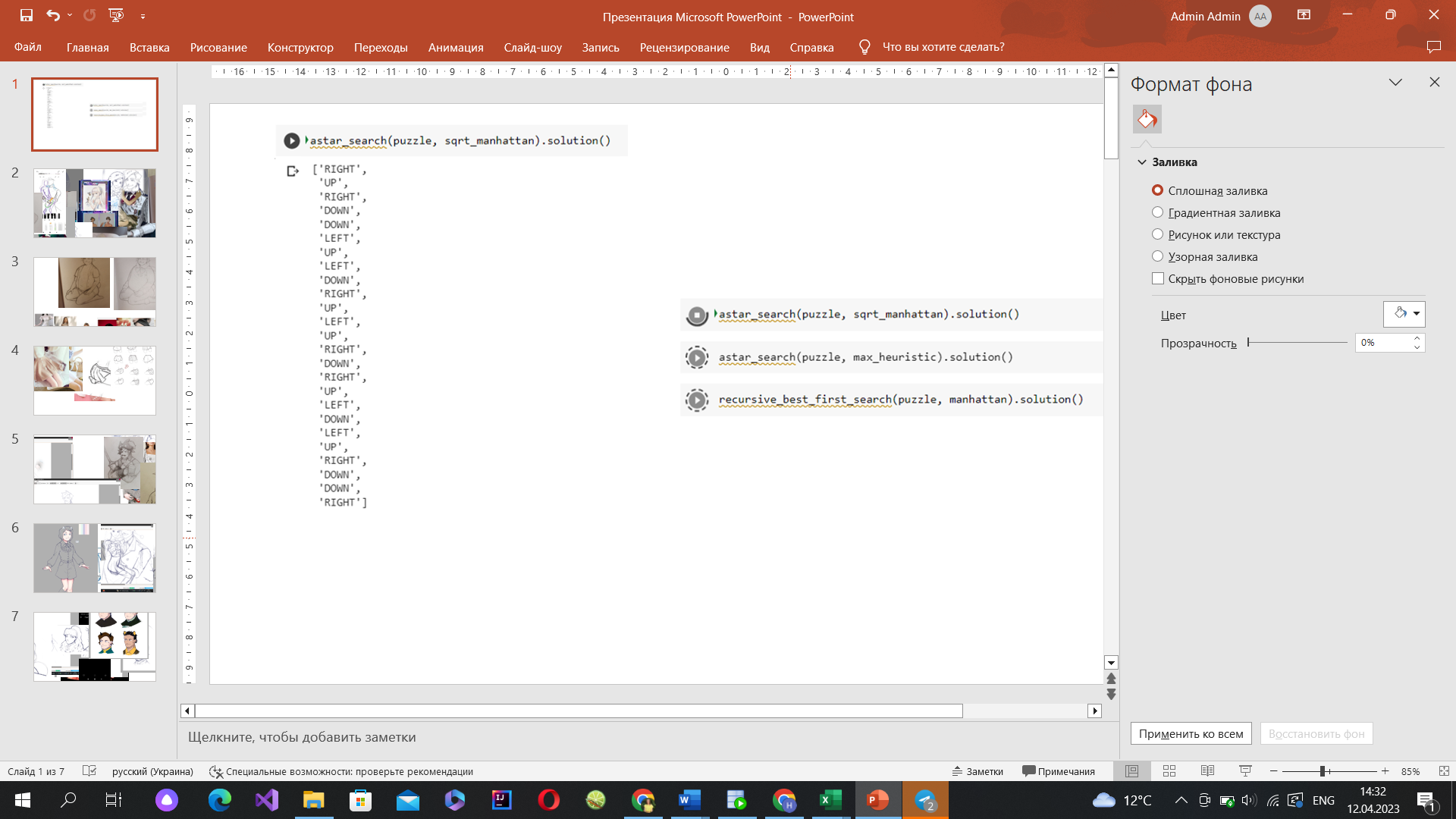
# Solving the puzzle

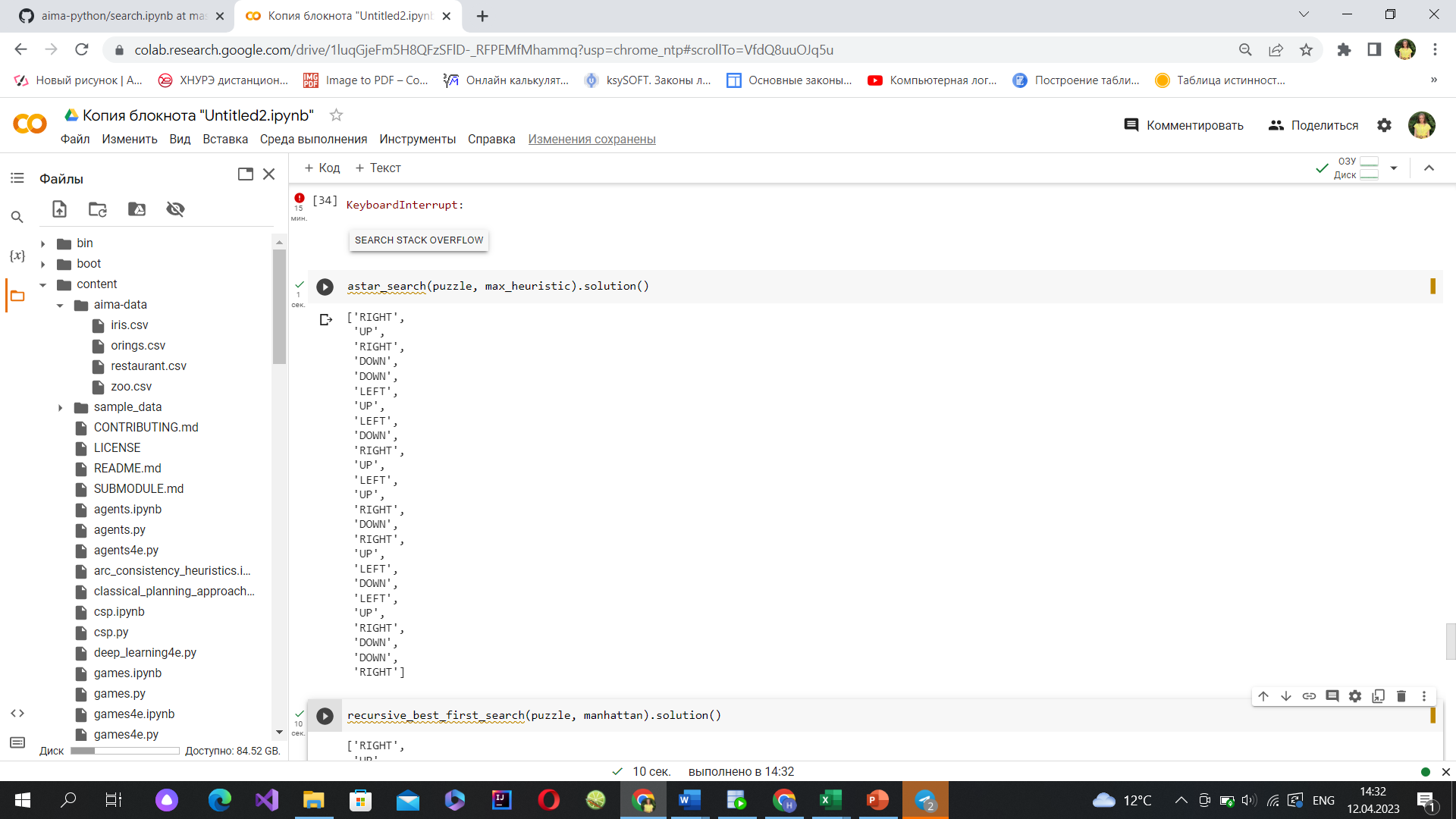
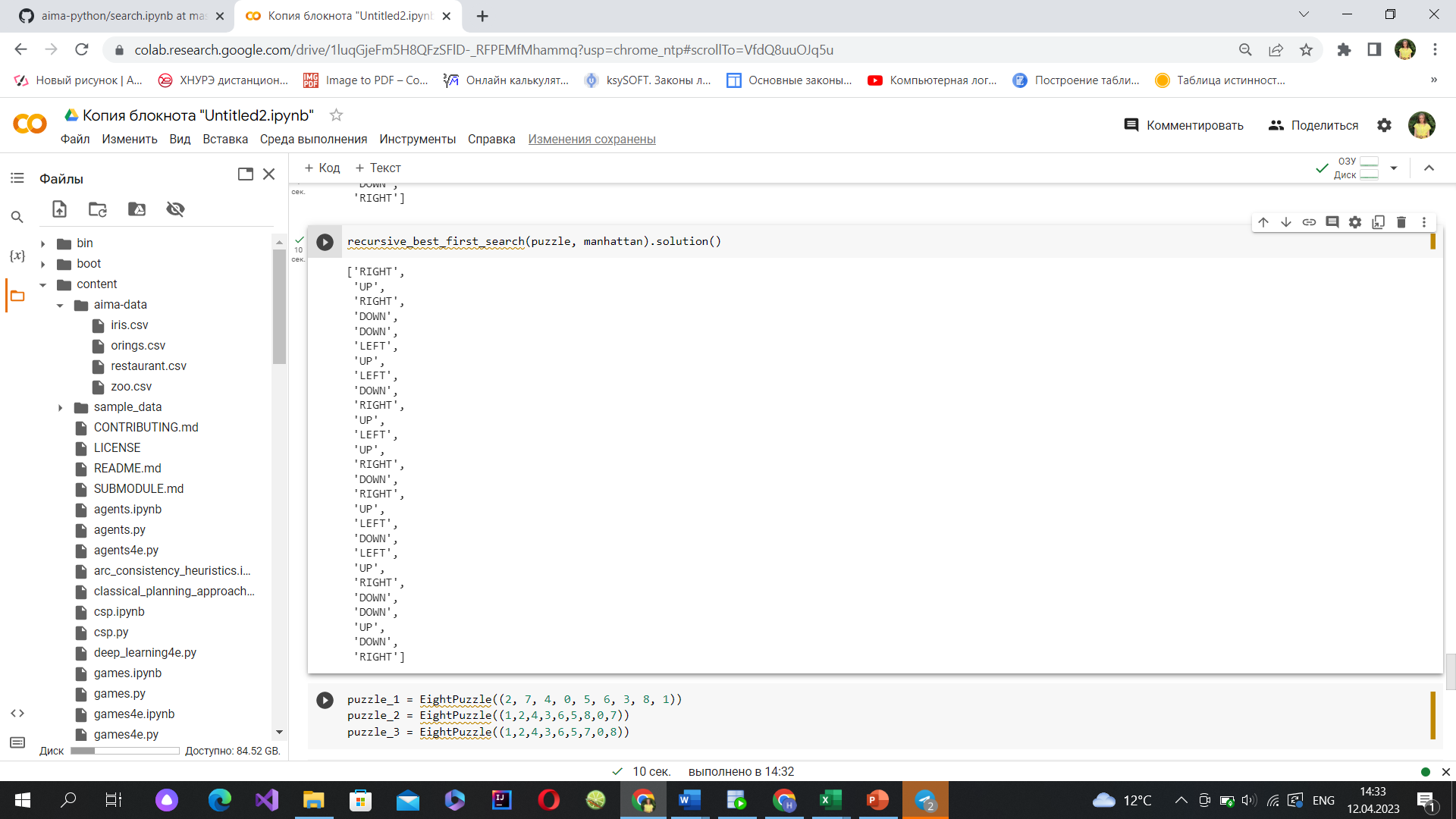
puzzle = EightPuzzle((2, 7, 4, 0, 5, 6, 3, 8, 1))

puzzle.check\_solvability((2, 7, 4, 0, 5, 6, 3, 8, 1)) # checks whether the initialized configuration is solvable or not



puzzle\_1 = EightPuzzle((2, 7, 4, 0, 5, 6, 3, 8, 1))

puzzle\_2 = EightPuzzle((1,2,4,3,6,5,8,0,7))

puzzle\_3 = EightPuzzle((1,2,4,3,6,5,7,0,8))

**1**

%%timeit

astar\_search(puzzle\_1)

astar\_search(puzzle\_2)

astar\_search(puzzle\_3)

Результат:

59.1 ms ± 43.54 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)

**2**

%%timeit

astar\_search(puzzle\_1, linear)

astar\_search(puzzle\_2, linear)

astar\_search(puzzle\_3, linear)

Результат:

32.7 ms ± 18.2 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)

**3**

%%timeit

astar\_search(puzzle\_1, manhattan)

astar\_search(puzzle\_2, manhattan)

astar\_search(puzzle\_3, manhattan)

Результат:

62.1 ms ± 31.02 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)

**4**

%%timeit

astar\_search(puzzle\_1, sqrt\_manhattan)

astar\_search(puzzle\_2, sqrt\_manhattan)

astar\_search(puzzle\_3, sqrt\_manhattan)

Результат:

43.6 ms ± 325 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)

**5**

%%timeit

astar\_search(puzzle\_1, max\_heuristic)

astar\_search(puzzle\_2, max\_heuristic)

astar\_search(puzzle\_3, max\_heuristic)

Результат:

28.3 ms ± 412 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)

%%timeit

recursive\_best\_first\_search(puzzle\_1, linear)

recursive\_best\_first\_search(puzzle\_2, linear)

recursive\_best\_first\_search(puzzle\_3, linear)

**6**

Результат:

56.4 ms ± 7.05 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)

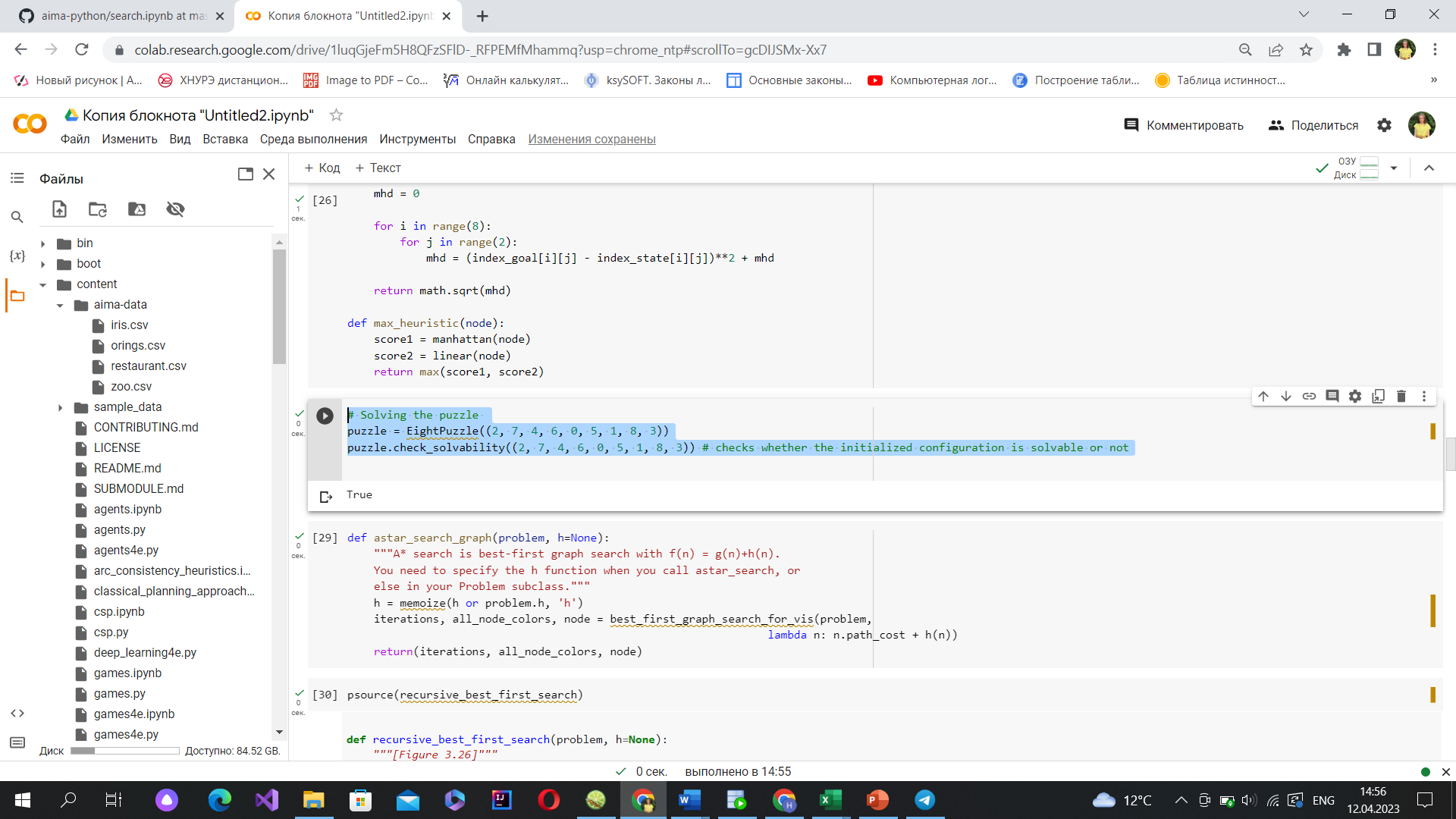
1. **Задана ціль №3:**

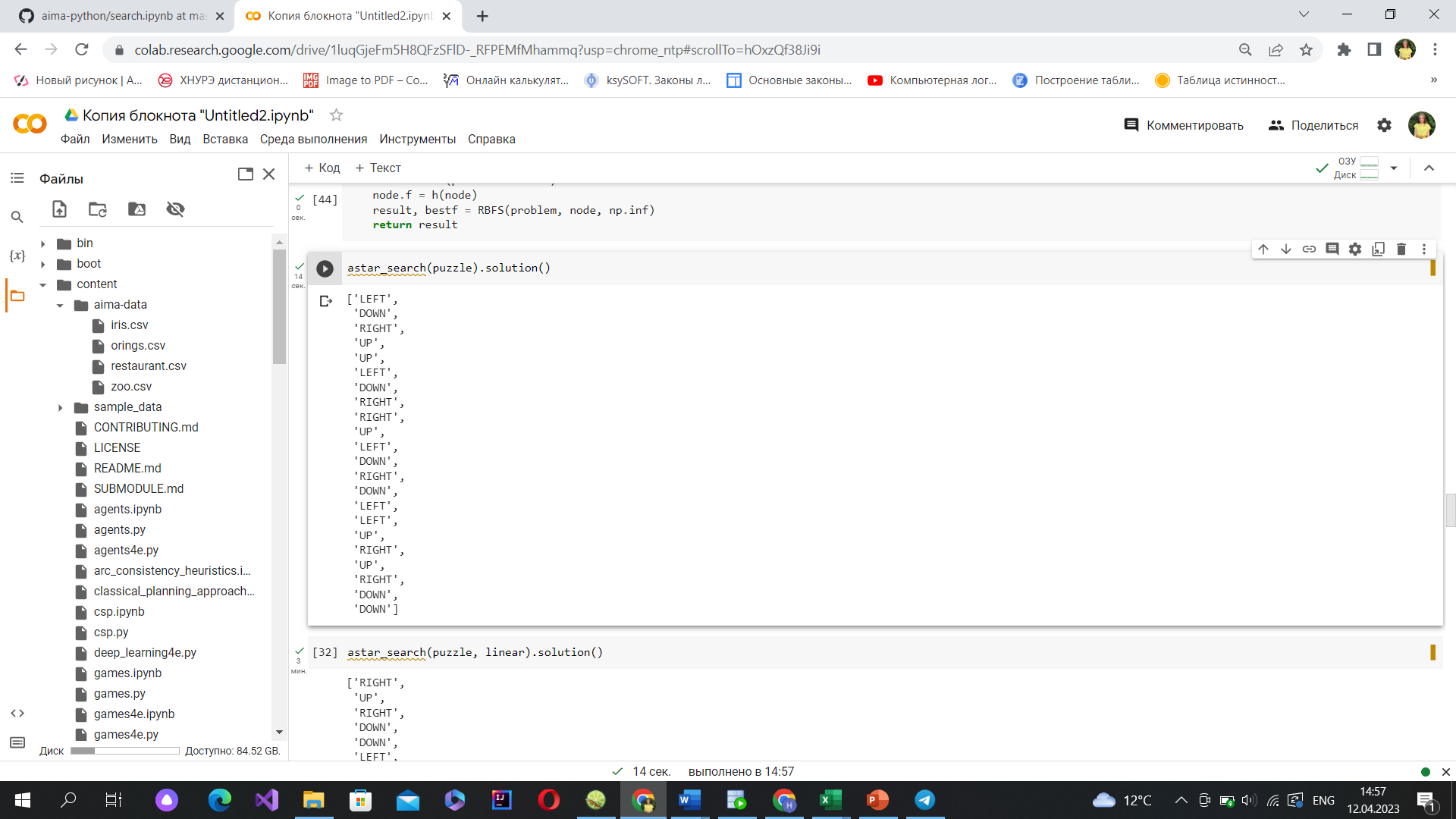
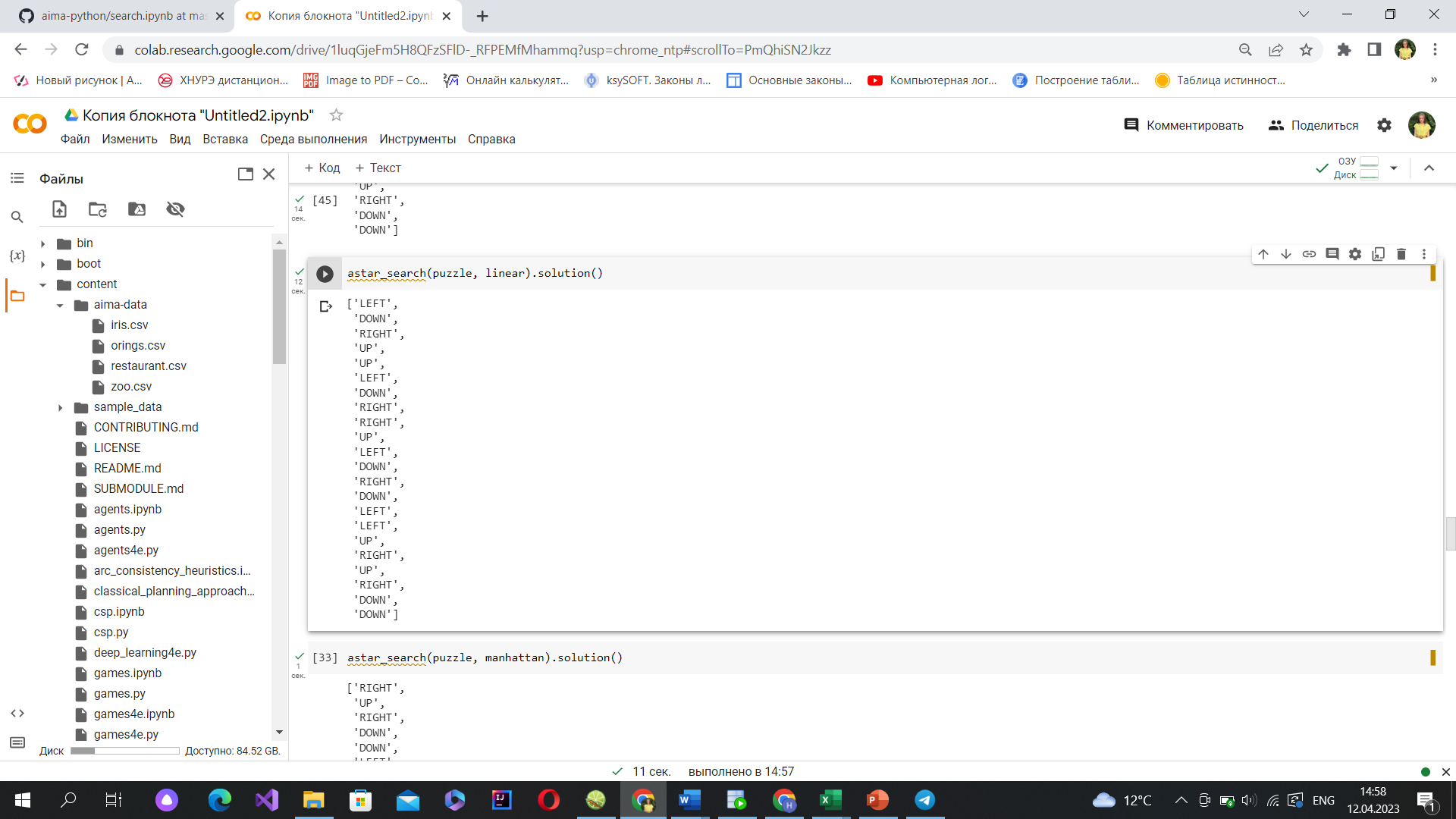
goal = [1, 2, 3, 4, 5, 6, 7, 8, 0]

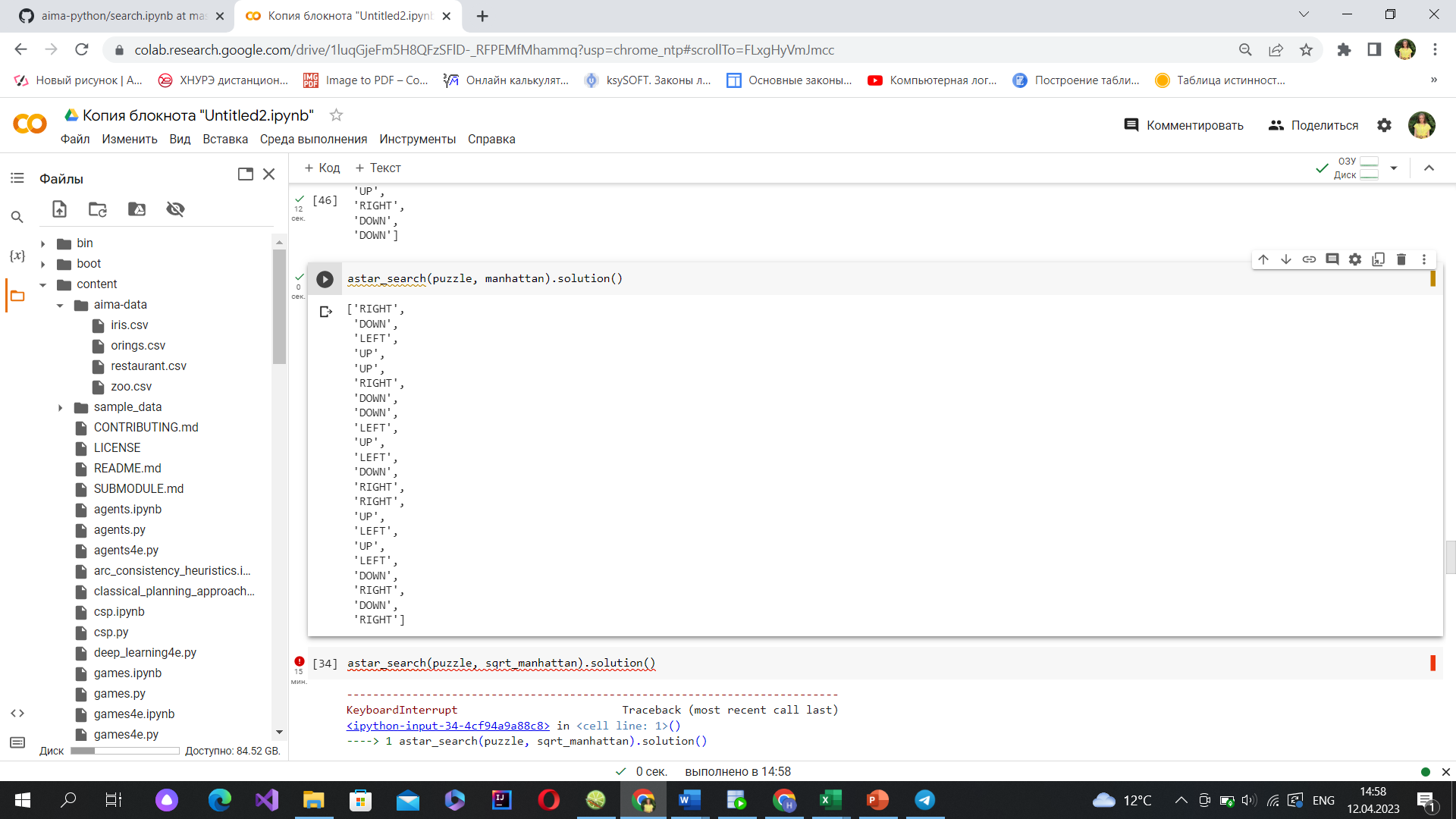
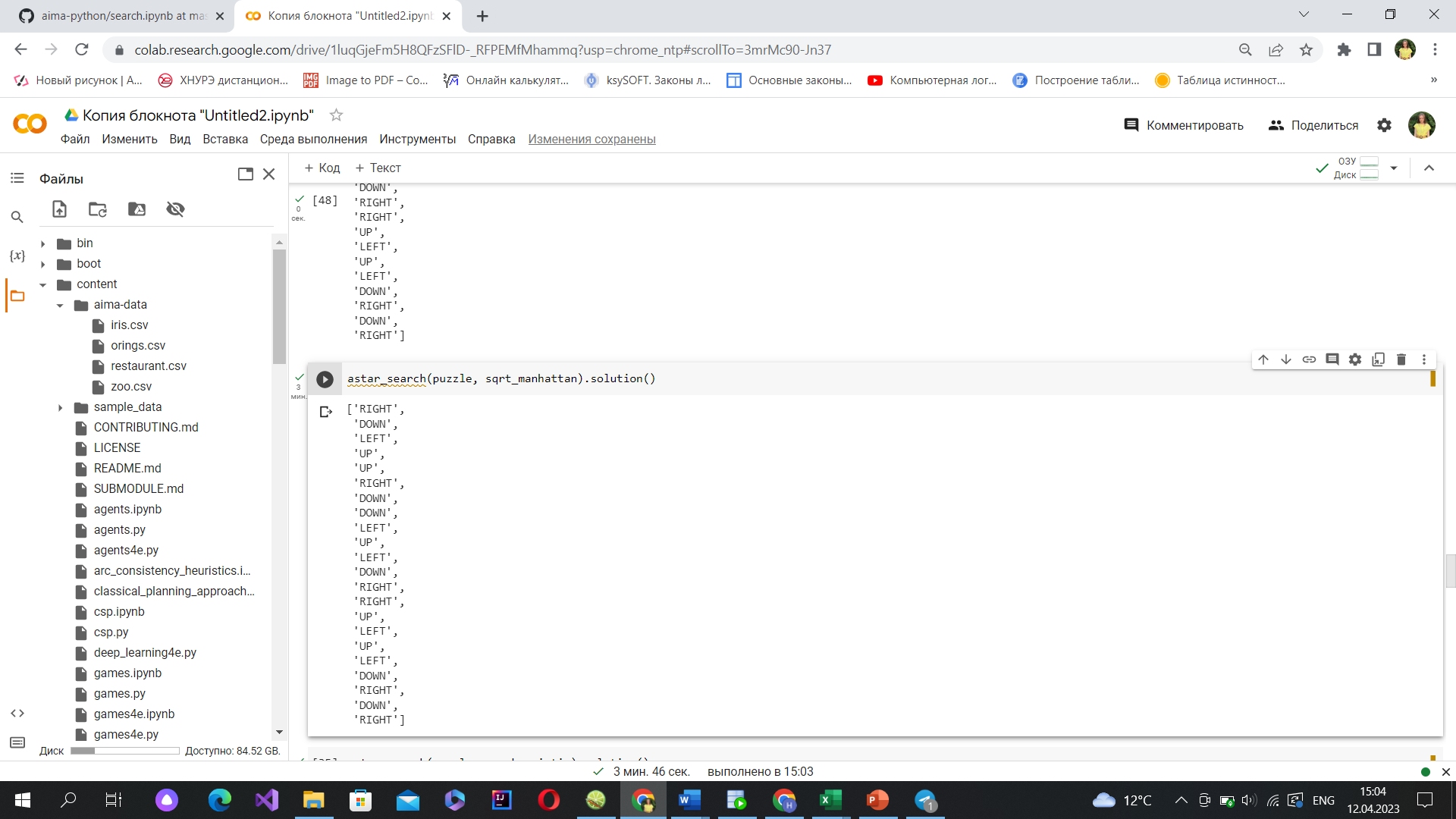
# Solving the puzzle

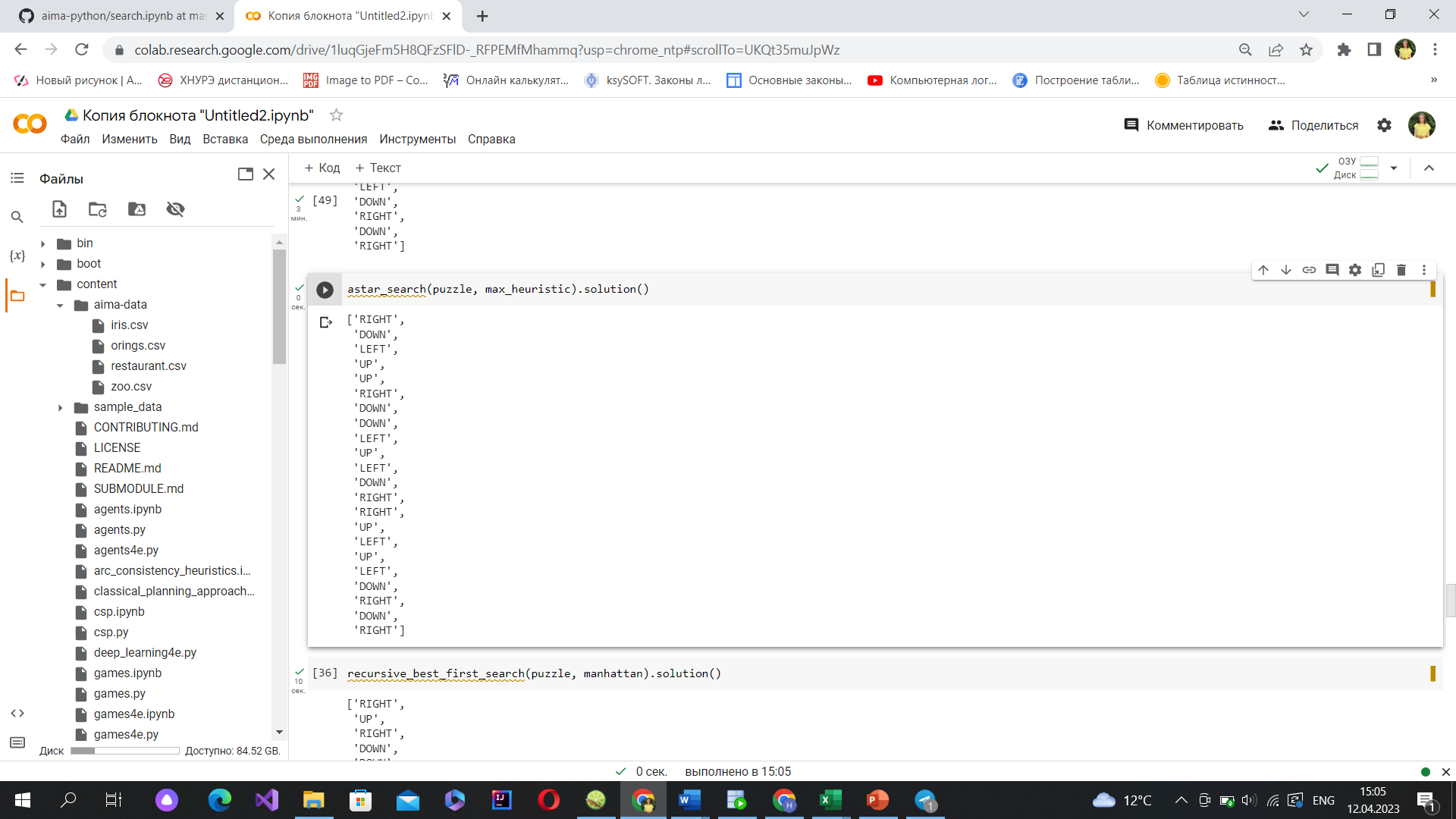
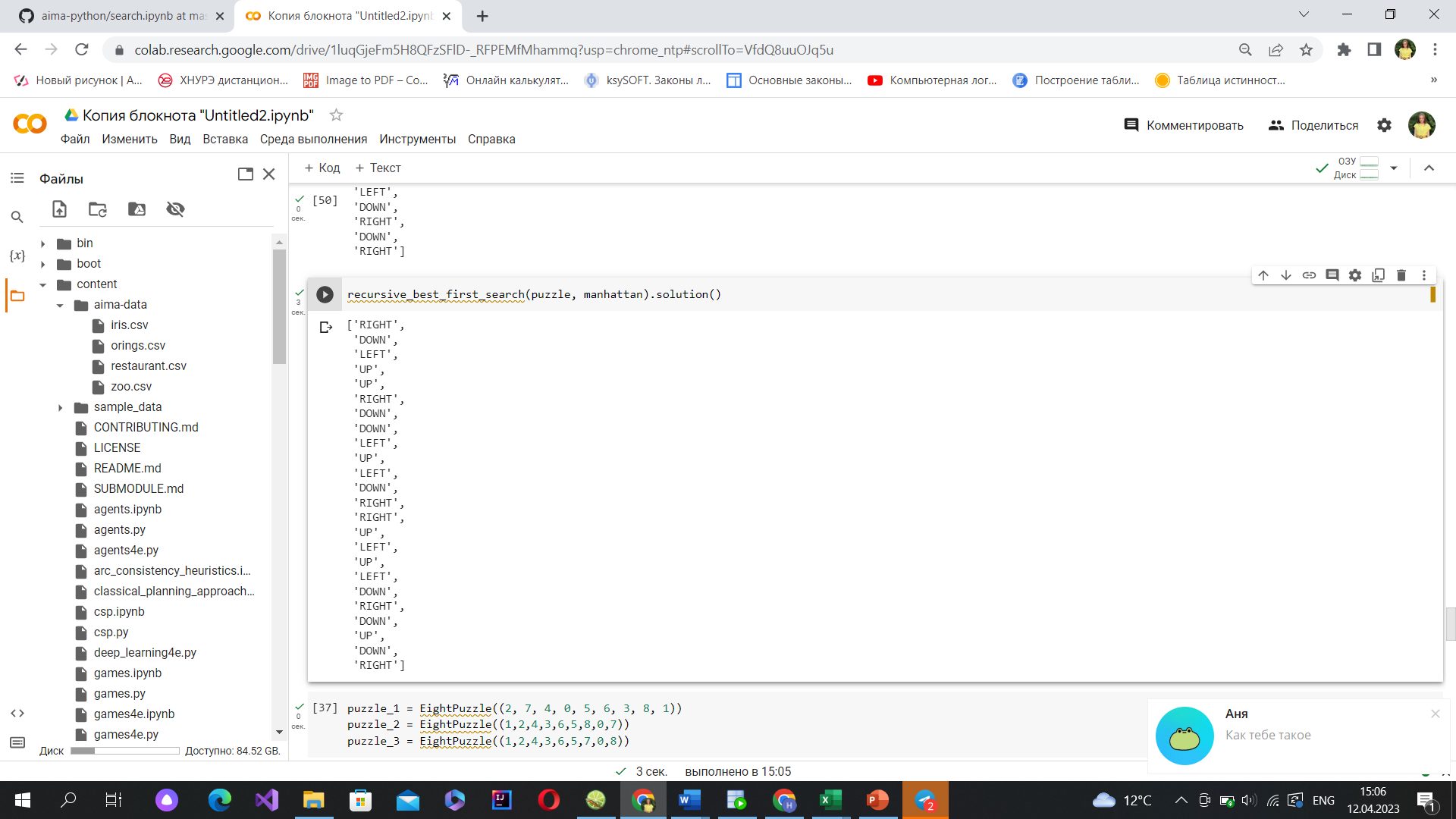
puzzle = EightPuzzle((2, 7, 4, 6, 0, 5, 1, 8, 3))

puzzle.check\_solvability((2, 7, 4, 6, 0, 5, 1, 8, 3)) # checks whether the initialized configuration is solvable or not



puzzle\_1 = EightPuzzle((2, 7, 4, 6, 0, 5, 1, 8, 3))

puzzle\_2 = EightPuzzle((1, 2, 3, 4, 6, 5, 8, 0, 7))

puzzle\_3 = EightPuzzle((1, 2, 3, 4, 6, 5, 7, 0, 8))

**1**

%%timeit

astar\_search(puzzle\_1)

astar\_search(puzzle\_2)

astar\_search(puzzle\_3)

Результат:

23.4 ms ± 19.04 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)

**2**

%%timeit

astar\_search(puzzle\_1, linear)

astar\_search(puzzle\_2, linear)

astar\_search(puzzle\_3, linear)

Результат:

42.7 ms ± 4.17 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)

**3**

%%timeit

astar\_search(puzzle\_1, manhattan)

astar\_search(puzzle\_2, manhattan)

astar\_search(puzzle\_3, manhattan)

Результат:

30.18 ms ± 167 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)

**4**

%%timeit

astar\_search(puzzle\_1, sqrt\_manhattan)

astar\_search(puzzle\_2, sqrt\_manhattan)

astar\_search(puzzle\_3, sqrt\_manhattan)

Результат:

57.34 ms ± 5.08 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)

**5**

%%timeit

astar\_search(puzzle\_1, max\_heuristic)

astar\_search(puzzle\_2, max\_heuristic)

astar\_search(puzzle\_3, max\_heuristic)

Результат:

55.8 ms ± 3.05 s per loop (mean ± std. dev. of 7 runs, 10 loops each)

**6**

%%timeit

recursive\_best\_first\_search(puzzle\_1, linear)

recursive\_best\_first\_search(puzzle\_2, linear)

recursive\_best\_first\_search(puzzle\_3, linear)

Результат:

31.3 ms ± 8.23 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)

**Висновок:** у рамках даної роботи були здобуті навички роботи з евристичними функціями на прикладі задачі гри у вісім, що дозволяє найкраще продемонструвати характерні особливості всіх евристичних функцій в цілому. Відбулося дослідження особливості використання генетичних алгоритмів для вирішення задачі гри в вісім.