

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ
ФЕДЕРАЦИИ

ФГБОУ ВО «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ «МЭИ»
ИНЖЕНЕРНО-ЭКОНОМИЧЕСКИЙ ИНСТИТУТ

ЛАБОРАТОРНАЯ РАБОТА

по дисциплине «Объектно-ориентированный анализ и программирование»
на тему:
«Классы»

Работу выполнил:
Студент группы ИЭс-160п-19
Зубков Д. Ю.

Принял:
Преподаватель Овсянникова М. Р.

Москва
2021

Оглавление

Условие задачи.....	3
Метод решения задачи.....	3
Алгоритм решения задачи	4
Наборы тестовых данных	4
Файл «cubes.csv»	4
Состав данных.....	5
Код программы	5
Код модуля «Cube.py»	5
Код модуля «menu.py».....	6
Код модуля «io_unit.py»	7
Код исполняемого файла «main.py»	9
Тестирование и отладка	10
1. Запуск программы.....	10
2. Просмотр кубов.....	10
3. Добавление куба.....	11
4. Удаление куба	12
5. Сравнение кубов.....	13
6. Сохранение в файл.....	14

Условие задачи

Вариант № 7

Разработать программу с использованием класса объектов.

Для класса объектов разработать подпрограммы (методы класса) для:

- создания объекта с заданными значениями,
- показать характеристики объекта класса,
- вычисления периметра геометрической фигуры,
- вычисления площади геометрической фигуры.

В программе:

- создать один объект класса и показать его характеристики;
- создать два объекта класса;
- сравнить вычисленные значения для двух объектов класса, по результатам каждого из сравнений вывести соответствующие сообщения.

Фигура: куб

Вычисляемые параметры: площадь поверхности, объем

Метод решения задачи

Задача будет решаться методом декомпозиции задач на составные части, применением математических расчетов.

Формула для решения задачи:

Площадь поверхности: $12 a^2$

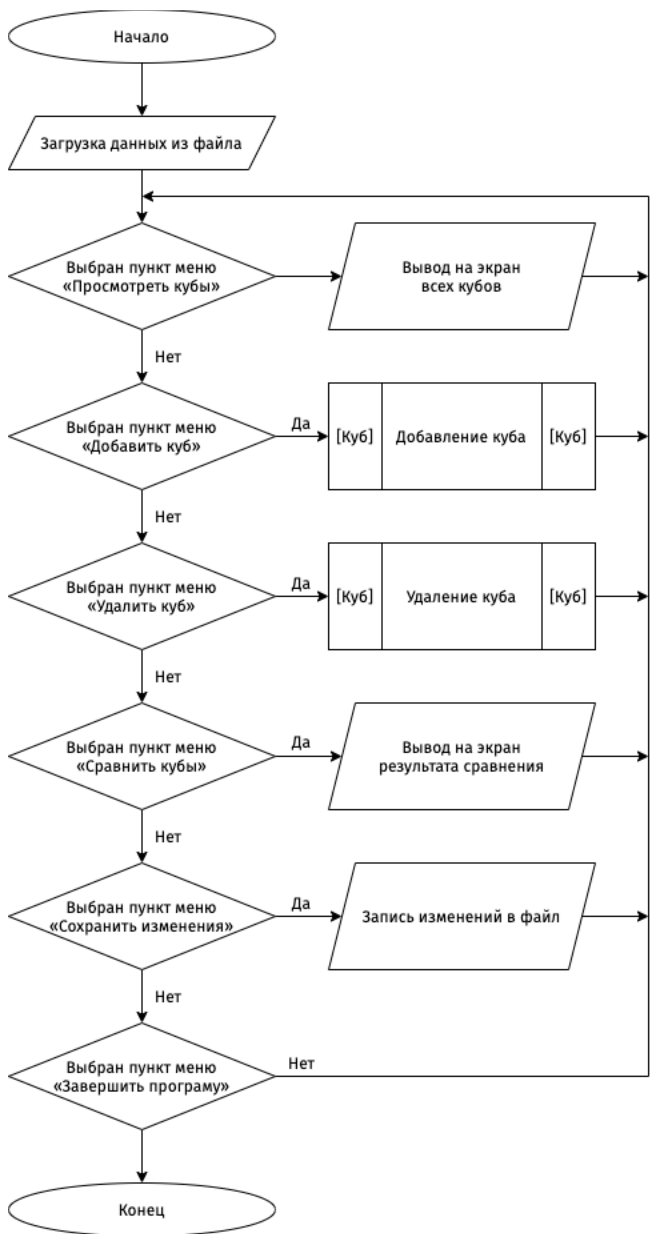
Объем: a^3

Поскольку, справедливо, что при сторонах a и b , где $a > b$, S_a и V_a будут соответственно больше S_b и V_b , сравнение будет проводиться только по стороне куба.

Техническое выполнение задания и тестирование будет проводиться в следующих условиях:

Язык программирования	Python 3.9
Среда разработки	JetBrains PyCharm Community 2020.3
Архитектура	Intel i386 (Core i9 9880H)
Операционная система	Apple macOS 11.2.1

Алгоритм решения задачи



Наборы тестовых данных

Тестовые данные представлены в виде трех .txt файлов.

Файл «cubes.csv»

Тестовые данные

2.24
3.20
6.10

Ожидаемый результат

№	Side	Perimeter	Area	Volume
1	2.14	25.68	27.48	9.80
2	3.20	38.40	61.44	32.77
3	6.10	73.20	223.26	226.98

Состав данных

Класс	Имя	Тип	Структура	Смысл
Входные данные	cubes	Класс	Одномерный массив	Исходный массив кубов
Выходные данные	cubes	Класс	Одномерный массив	Сохраняемый массив кубов
Промежуточные данные	cubes_to_compare	Класс	Одномерный массив	Массив кубов для сравнения

Код программы

Код модуля «Cube.py»

Класс Cube

```
1 class Cube:
2     def __init__(self, side):
3         self.side = abs(side)
4
5     def __str__(self):
6         return f'Side: {self.side:.2f}\n' \
7                f'Perimeter: {self.perimeter:.2f}\n' \
8                f'Area: {self.area:.2f}\n' \
9                f'Volume: {self.volume:.2f}\n'
10
11    def __gt__(self, other):
12        return self.side > other.side
13
14    def __eq__(self, other):
15        return self.side == other.side
16
17    def __ge__(self, other):
18        return self.side >= other.side
19
20    perimeter = property()
21    area = property()
22    volume = property()
23
24    @perimeter.getter
25    def perimeter(self):
26        return self.side * 12
27
28    @perimeter.setter
29    def perimeter(self, value):
30        self.side = value / 12
31
32    @area.getter
33    def area(self):
34        return self.side ** 2 * 6
35
36    @area.setter
37    def area(self, value):
38        self.side = (value / 6) ** (1 / float(2))
39
40    @volume.getter
41    def volume(self):
42        return self.side ** 3
43
```

```

44     @volume.setter
45     def volume(self, value):
46         self.side = value ** (1 / float(3))
47

```

Код модуля «menu.py»

Классы MenuItem, Menu

```

1  import os
2
3
4  class MenuItem:
5      def __init__(self, text, func, params=None):
6          self.text = text
7          self.func = func
8          self.params = params
9
10
11 class Menu:
12     def __init__(self, items: [MenuItem]):
13         self.items = items
14
15     def show(self, text='', info_msg='', avoid_clr=False):
16         act_n = 0
17         while act_n == 0:
18             os.system('clear')
19             if text != '' and avoid_clr:
20                 print(text)
21             for index, item in enumerate(self.items):
22                 print(f'{index + 1}: {item.text}')
23             if info_msg == '':
24                 info_msg = '\n'
25             print(info_msg)
26             usr_inp = input("Type number of action and press return: ")
27             try:
28                 act_n = int(usr_inp)
29             except Exception:
30                 info_msg = f'\nERROR: {usr_inp} -- wrong input'
31                 act_n = 0
32
33             if act_n > len(self.items) or act_n < 1:
34                 info_msg = f'\nERROR: {usr_inp} -- wrong input'
35                 act_n = 0
36             else:
37                 if self.items[act_n - 1].params is not None:
38                     act_n, info_msg = self.items[act_n - 1].func(*self.items[act_n - 1].params)
39                 else:
40                     act_n, info_msg = self.items[act_n - 1].func()
41             if act_n == -1:
42                 act_n = 0
43             return act_n, info_msg
44

```

Код модуля «io_unit.py»

```
1  import csv
2  import os
3  from Cube import Cube
4  from menu import Menu, MenuItem
5
6
7  def load_cubes_from_csv(filename):
8      cubes = []
9      if os.path.exists(filename) and os.path.isfile(filename):
10         with open(filename, encoding='utf-8') as r_file:
11             try:
12                 file_reader = csv.reader(r_file, delimiter=",")
13                 for row in file_reader:
14                     cube = Cube(float(row[0]))
15                     cubes.append(cube)
16             except Exception:
17                 pass
18         return cubes
19
20
21 def save_to_csv(filename, cubes: [Cube]):
22     with open(filename, mode='w', encoding='utf-8') as w_file:
23         f_writer = csv.writer(w_file, delimiter=',', quotechar='"',
24                                quoting=csv.QUOTE_MINIMAL)
25         for cube in cubes:
26             print(cube.side)
27             f_writer.writerow([cube.side])
28         return 0, f'\nSaved to {filename}'
29
30 def show_arg_val_table(table_data, annotations):
31     res_str = ' '
32     col_width = get_max_value_length(table_data, annotations)
33     res_str += ' '.join([f'{a:<{col_width[index]}}' for index, a in
34                          enumerate(annotations)]) + '\n'
35     res_str += ('-' * len(res_str) + '\n')
36     for i in range(len(table_data[0])):
37         res_str += ' '
38         for index, table in enumerate(table_data):
39             if table[i] < 0:
40                 res_str = res_str[:-1]
41             if table[i] == int(table[i]):
42                 res_str += f'{table[i]:<{col_width[index]}.0f}'
43             else:
44                 res_str += f'{table[i]:<{col_width[index]}.2f}'
45             if table[i] < 0:
46                 res_str += ' '
47         res_str += '\n'
48     return res_str
49
50 def get_max_value_length(table_data, annotations):
51     n = [max(5, len(f'{a} ')) for a in annotations]
52     for i in range(len(table_data[0])):
53         n = [max(n[index], len(f'{(abs(table[i])):.2f} ')) for index, table
54              in enumerate(table_data)]
55     return n
56
```

```

57 def present_cubes(cubes: [Cube]):
58     annotations = ['№', 'Side', 'Perimeter', 'Area', 'Volume']
59     nums, sides, perimeters, areas, volumes = [], [], [], [], []
60     for index, cube in enumerate(cubes):
61         nums.append(index + 1)
62         sides.append(cube.side)
63         perimeters.append(cube.perimeter)
64         areas.append(cube.area)
65         volumes.append(cube.volume)
66     table_data = [nums, sides, perimeters, areas, volumes]
67     return show_arg_val_table(table_data, annotations)
68
69
70 def show_cubes(cubes: [Cube]):
71     text = present_cubes(cubes)
72     menu = Menu([
73         MenuItem("Back", lambda x, y: (x, y), (-1, ''))
74     ])
75     return menu.show(avoid_clr=True, text=text)
76
77
78 def add_cube(cubes: [Cube]):
79     os.system('clear')
80     side = input('Type side of cube: ')
81     try:
82         side_f = float(side)
83         if side_f > 0:
84             cubes.append(Cube(side_f))
85             msg = f'\nAdded cube with side {side}'
86         else:
87             msg = '\nERROR: can't add cube, side must be > 0'
88     except ValueError:
89         msg = '\nERROR: can't add cube, wrong input'
90     return 0, msg
91
92
93 def delete_cube(cubes: [Cube]):
94     os.system('clear')
95     print(present_cubes(cubes))
96     pos = input('Type № of cube to delete: ')
97     try:
98         index = int(pos)
99         if index - 1 in range(len(cubes)):
100             cubes.pop(index - 1)
101             msg = f'\nDeleted cube at № {index}'
102         else:
103             msg = '\nERROR: can't delete cube, wrong index'
104     except ValueError:
105         msg = '\nERROR: can't delete cube, wrong input'
106     return 0, msg
107
108
109 def compare_cubes(cubes: [Cube]):
110     os.system('clear')
111     print(present_cubes(cubes))
112     pos = input('Type №№ of cubes to compare divided by space: ')
113     str_arr = pos.strip().split(' ')
114     try:
115         index1, index2 = int(str_arr[0]), int(str_arr[1])
116         if index1 - 1 in range(len(cubes)) and index2 - 1 in
range(len(cubes)):

```



```

117         cubes_to_compare = [cubes[index1 - 1], cubes[index2 - 1]]
118         os.system('clear')
119         menu = Menu([
120             MenuItem("Back", lambda x, y: (x, y), (-1, ''))
121         ])
122         msg = (present_cubes(cubes_to_compare))
123         if cubes_to_compare[0] < cubes_to_compare[1]:
124             msg += f'\nCube №{index1} less than cube №{index2}'
125         elif cubes_to_compare[0] > cubes_to_compare[1]:
126             msg += f'\nCube №{index1} greater than cube №{index2}'
127         else:
128             msg += f'\nCube №{index1} equal cube №{index2}'
129         return menu.show(avoid_clr=True, text=msg)
130     else:
131         msg = '\nERROR: can't compare cubes, wrong index'
132     except ValueError:
133         msg = '\nERROR: can't compare cubes, wrong input'
134     return 0, msg
135

```

Код исполняемого файла «main.py»

```

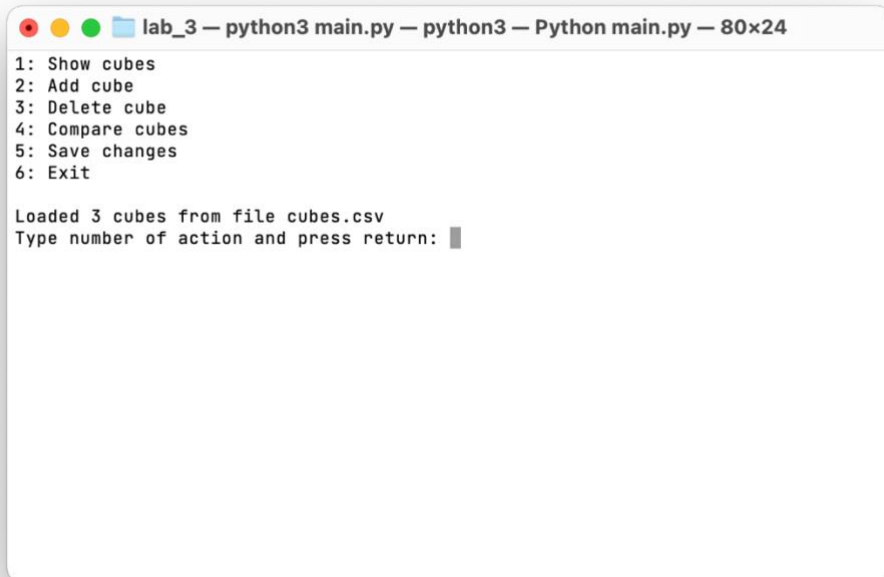
1  from Cube import Cube
2  import os
3  from menu import Menu, MenuItem
4  import io_unit as iou
5
6
7  if __name__ == "__main__":
8      filename = 'cubes.csv'
9      cubes = iou.load_cubes_from_csv('cubes.csv')
10     if cubes:
11         text = (f'Loaded {len(cubes)} cubes from file {filename}')
12     else:
13         text = 'No one cube loaded'
14     main_menu = Menu([
15         MenuItem('Show cubes', iou.show_cubes, (cubes,)),
16         MenuItem('Add cube', iou.add_cube, (cubes,)),
17         MenuItem('Delete cube', iou.delete_cube, (cubes, )),
18         MenuItem('Compare cubes', iou.compare_cubes, (cubes, )),
19         MenuItem('Save changes', iou.save_to_csv, ('cubes.csv', cubes)),
20         MenuItem('Exit', exit, (0,))
21     ])
22     main_menu.show(avoid_clr=True, info_msg=f'\n{text}')
23

```

Тестирование и отладка

1. Запуск программы

Ожидаемый результат: отображение меню программы.



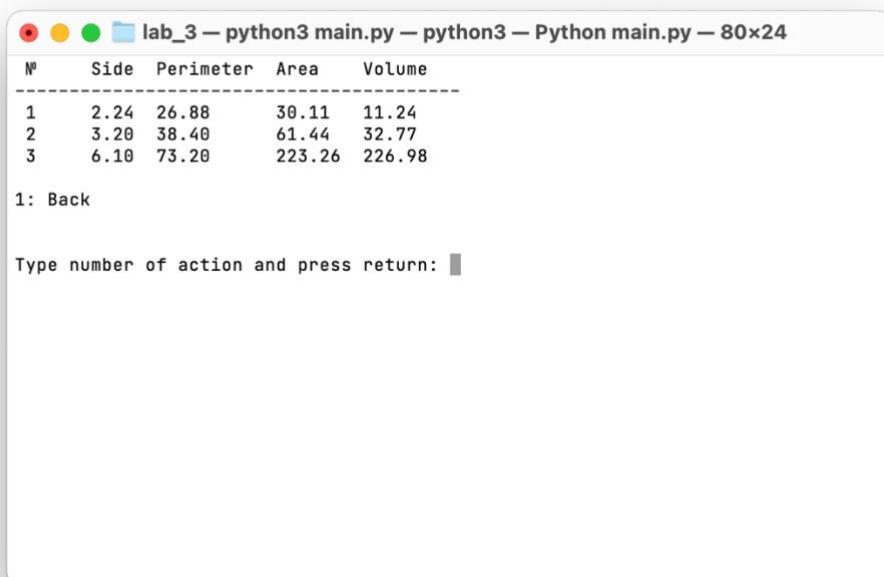
```
lab_3 — python3 main.py — python3 — Python main.py — 80x24
1: Show cubes
2: Add cube
3: Delete cube
4: Compare cubes
5: Save changes
6: Exit

Loaded 3 cubes from file cubes.csv
Type number of action and press return: █
```

Вывод: файл «cubes.csv» загружен, данные считаны. Тест пройден.

2. Просмотр кубов.

Ожидаемый результат: отображение списка кубов



```
lab_3 — python3 main.py — python3 — Python main.py — 80x24
№      Side  Perimeter  Area   Volume
-----
1       2.24   26.88      39.11  11.24
2       3.20   38.40      61.44  32.77
3       6.10   73.20     223.26 226.98

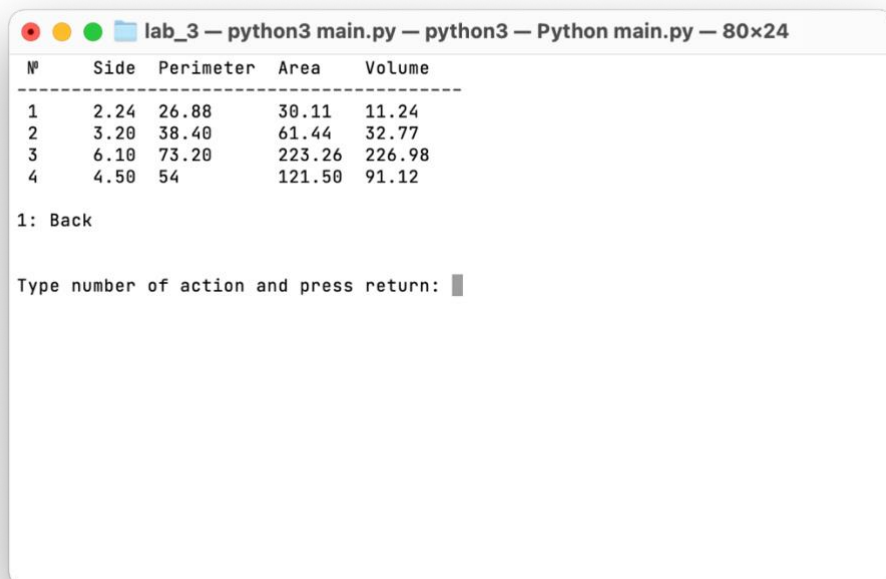
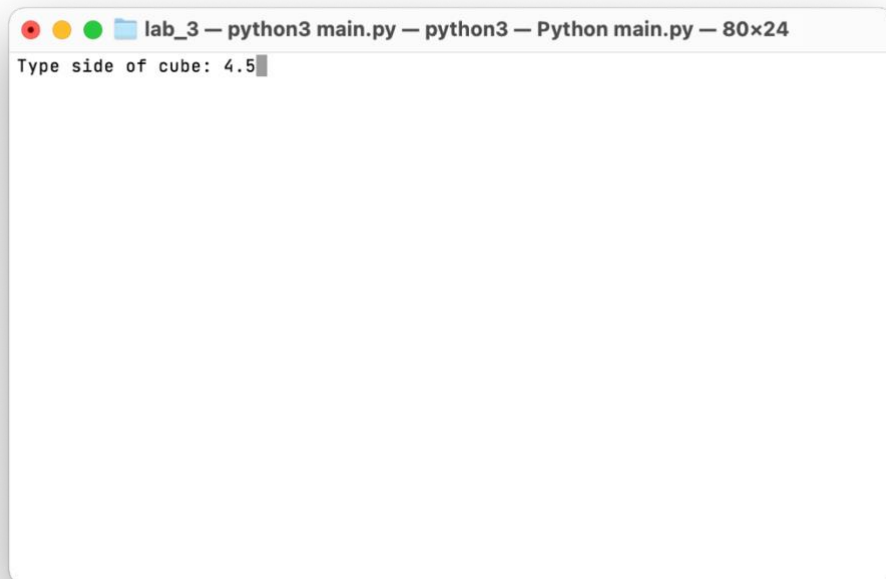
1: Back

Type number of action and press return: █
```

Вывод: кубы из файла отображаются в табличном виде. Тест пройден.

3. Добавление куба.

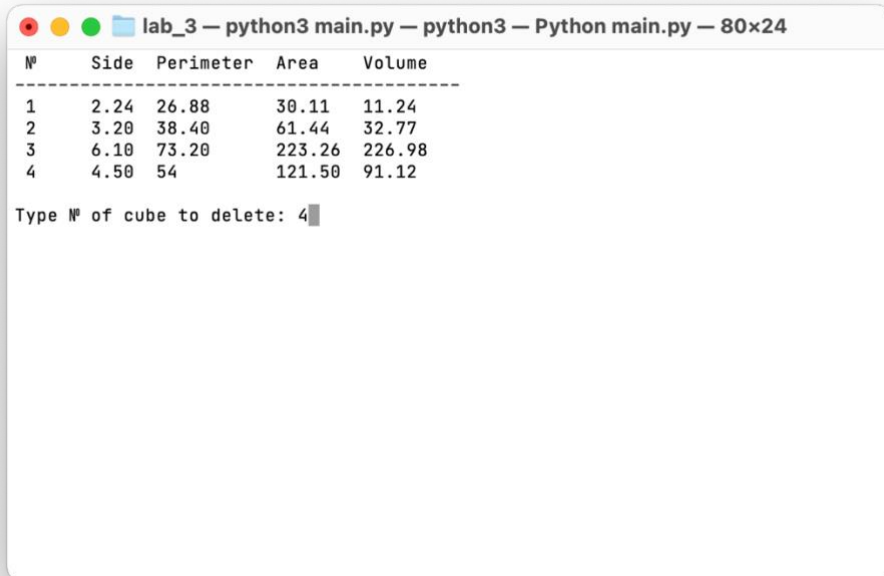
Ожидаемый результат: добавление куба в массив кубов. Отображение в списке кубов.



Вывод: куб добавлен, отображается в списке кубов. Тест пройден.

4. Удаление куба

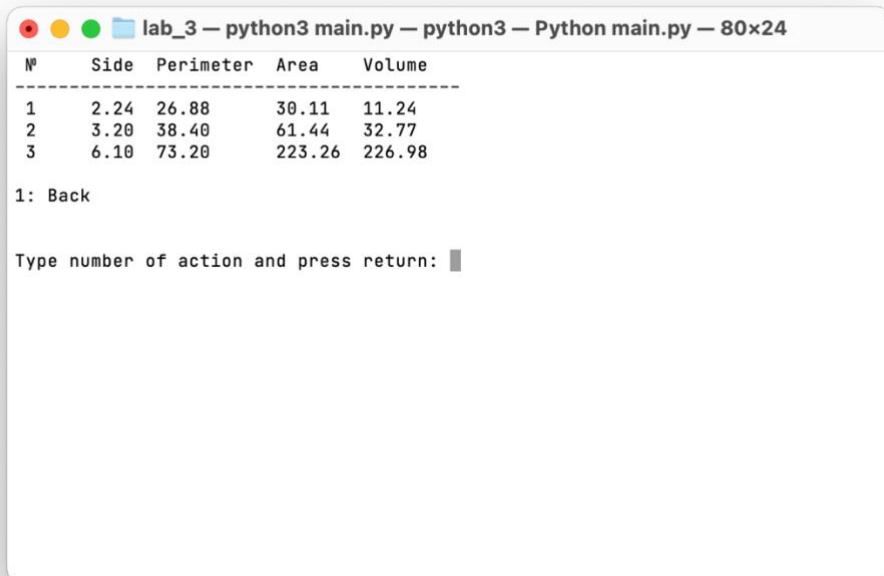
Ожидаемый результат: удаление куба из массива кубов. Отсутствие в списке кубов.



```
lab_3 — python3 main.py — python3 — Python main.py — 80x24
```

№	Side	Perimeter	Area	Volume
1	2.24	26.88	30.11	11.24
2	3.20	38.40	61.44	32.77
3	6.10	73.20	223.26	226.98
4	4.50	54	121.50	91.12

Type № of cube to delete: 4



```
lab_3 — python3 main.py — python3 — Python main.py — 80x24
```

№	Side	Perimeter	Area	Volume
1	2.24	26.88	30.11	11.24
2	3.20	38.40	61.44	32.77
3	6.10	73.20	223.26	226.98

1: Back

Type number of action and press return:

Вывод: куб удален, не отображается в списке кубов. Тест пройден.

5. Сравнение кубов

Ожидаемый результат: сравнение двух кубов из массива. Отображение результата.

```
lab_3 — python3 main.py — python3 — Python main.py — 80x24
```

№	Side	Perimeter	Area	Volume
1	2.24	26.88	30.11	11.24
2	3.20	38.40	61.44	32.77
3	6.10	73.20	223.26	226.98

Type №№ of cubes to compare divided by space: 1 3

```
lab_3 — python3 main.py — python3 — Python main.py — 80x24
```

№	Side	Perimeter	Area	Volume
1	2.24	26.88	30.11	11.24
2	6.10	73.20	223.26	226.98

Cube №1 less than cube №3
1: Back

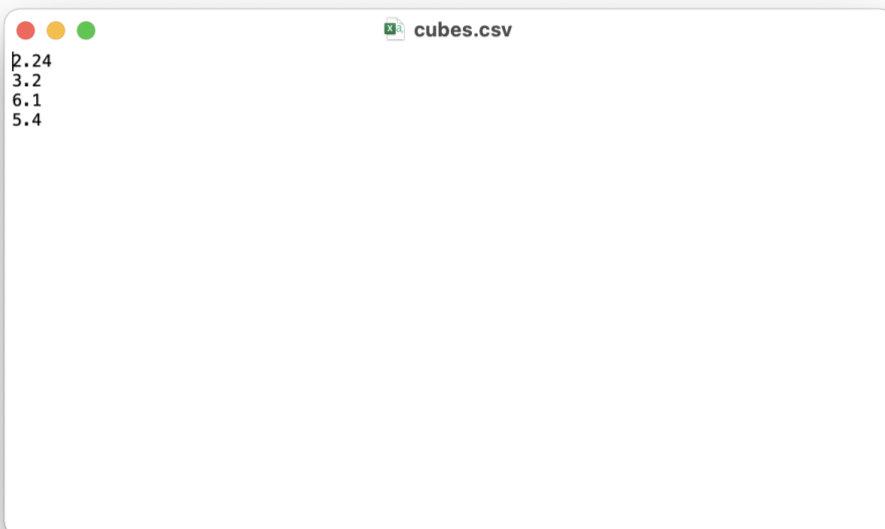
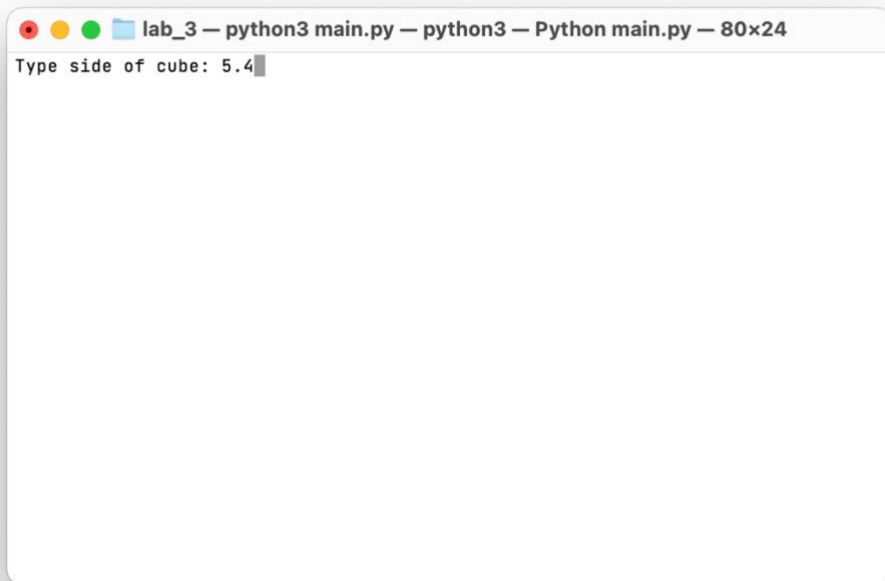
Type number of action and press return:

Вывод: куб удален, не отображается в списке кубов. Тест пройден.

6. Сохранение в файл

Ожидаемый результат: запись данных в файл.

Для теста добавим куб со стороной 5.4



Вывод: куб добавлен в файл. Тест пройден.