

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. Продолжение»

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»	10
Тестирование и отладка	11
1. Запуск программы.....	11
2. Просмотр кубов.....	12
3. Добавление куба.....	11
4. Удаление куба	13
5. Сравнение кубов.....	14
6. Сохранение в файл.....	14

Условие задачи

Вариант № 7

Разработать программу с использованием класса объектов.

В дополнение к условиям задания № 3

1. Создать массив объектов класса. Количество элементов массива пользователь вводит с клавиатуры.
2. Добавить метод класса – вывод характеристик объектов на экран дисплея в табличном виде. Таблица имеет заголовок и имена столбцов.
3. Сохранить сведения об объектах класса в типизированном файле.
4. Изменить характеристики третьей фигуры. Внести соответствующие изменения в типизированный файл.
5. Вывести на экран сведения обо всех фигурах, хранящихся в файле.

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: $12 a^2$

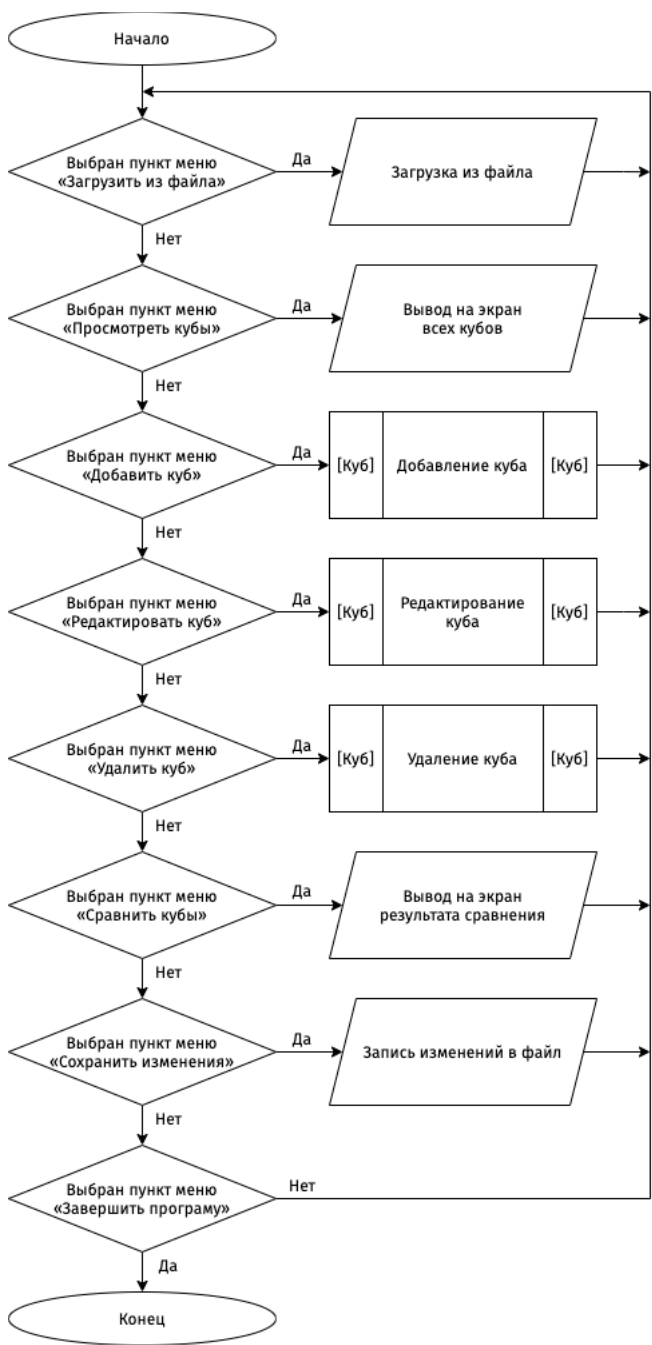
Объем: a^3

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

3.15
3.41
5.45

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
48     @staticmethod
49     def show_annotations(col_widths):
50         annotations = ['M', 'Side', 'Perimeter', 'Area', 'Volume']
51         return ''.join([f'a:<{col_widths[index]}' for index, a in
enumerate(annotations))]) + '\n'
52
53     def present_table(self, number, col_widths):
54         columns = [self.side, self.perimeter, self.area, self.volume]
55         res_str = ''.join(f'{number:<{col_widths[0]}.0f}')
56         for index, param in enumerate(columns):
57             if param == int(param):
58                 res_str += f'{param:<{col_widths[index + 1]}.0f}'
59             else:
60                 res_str += f'{param:<{col_widths[index + 1]}.2f}'
61         res_str += '\n'
62         return res_str
63

```

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]
39             .func(*self.items[act_n - 1].params)
40         else:
41             act_n, info_msg = self.items[act_n - 1].func()
42     if act_n == -1:
43         act_n = 0
44     return act_n, info_msg

```

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(cubes: [Cube]):
8      filename = input('Type path to file, or press return: ')
9      os.system('clear')
10     if os.path.exists(filename) and os.path.isfile(filename):
11         with open(filename, encoding='utf-8') as r_file:
12             try:
13                 file_reader = csv.reader(r_file, delimiter=",")
14                 new_cubes = []
15                 for row in file_reader:
16                     cube = Cube(float(row[0]))
17                     new_cubes.append(cube)
18                 if new_cubes:
19                     msg = f'\nSuccessfully loaded {len(new_cubes)} cubes'
20                 else:
21                     msg = 'No one cube loaded'
22                 del cubes[0:len(cubes)]
23                 cubes += new_cubes
24                 return 0, msg
25             except Exception:
26                 msg = f'\nERROR: Can't load cubes, file error'
27                 return 0, msg
28     msg = f'\nERROR: Can't load cubes, file not exist'
29     return 0, msg
30
31
32 def save_to_csv(cubes: [Cube]):
33     filename = input('Type name of file: ')
34     with open(filename, mode='w', encoding='utf-8') as w_file:
35         f_writer = csv.writer(w_file, delimiter=',', quotechar='"',
36                               quoting=csv.QUOTE_MINIMAL)
37         for cube in cubes:
38             f_writer.writerow([cube.side])
39     return 0, f'\nSaved to {filename}'
40
41 def get_max_value_length(cubes: [Cube], annotations):
42     nums, sides, perimeters, areas, volumes = [], [], [], [], []
43     for index, cube in enumerate(cubes):
44         nums.append(index + 1)

```

```

45         sides.append(cube.side)
46         perimeters.append(cube.perimeter)
47         areas.append(cube.area)
48         volumes.append(cube.volume)
49         table_data = [nums, sides, perimeters, areas, volumes]
50
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     col_widths = get_max_value_length(cubes, annotations)
60     res_str = ''.join([f'a:<{{col_widths[index]}}' for index, a in
61 enumerate(annotations)]) + '\n'
62     res_str += '-' * len(res_str) + '\n'
63     for index, cube in enumerate(cubes):
64         res_str += cube.present_table(index + 1, col_widths)
65     return res_str
66
67 def show_cubes(cubes: [Cube]):
68     text = present_cubes(cubes)
69     menu = Menu([
70         MenuItem("Back", lambda x, y: (x, y), (-1, ''))
71     ])
72     return menu.show(avoid_clr=True, text=text)
73
74
75 def add_cubes(cubes: [Cube]):
76     os.system('clear')
77     num = input('Type number of cubes to add: ')
78     msg = ''
79     bef_cubes_count = len(cubes)
80     try:
81         num = int(num)
82     except ValueError:
83         msg = '\nERROR: can't add cubes, wrong input'
84     for i in range(num):
85         code, msg = add_cube(cubes, msg)
86         print(msg)
87     cubes_added = len(cubes) - bef_cubes_count
88     os.system('clear')
89     print(present_cubes(cubes))
90     input('Press return')
91     msg = f'\nSuccessfully added {cubes_added} cubes'
92     return 0, msg
93
94
95 def add_cube(cubes: [Cube], msg=''):
96     os.system('clear')
97     print(present_cubes(cubes))
98     if msg:
99         print(msg)
100     side = input('Type side of cube: ')
101     try:
102         side_f = float(side)
103         if side_f > 0:

```



```

104         cubes.append(Cube(side_f))
105         msg = f'\nAdded cube with side {side}'
106     else:
107         msg = '\nERROR: can't add cube, side must be > 0'
108 except ValueError:
109     msg = '\nERROR: can't add cube, wrong input'
110 return 0, msg
111
112
113 def edit_cube(cubes: [Cube]):
114     os.system('clear')
115     print(present_cubes(cubes))
116     pos = input('Type № of cube to edit: ')
117     try:
118         index = int(pos)
119         if index - 1 in range(len(cubes)):
120             side = input('Type side of cube: ')
121             try:
122                 side_f = float(side)
123                 if side_f > 0:
124                     cubes[index - 1].side = side_f
125                     os.system('clear')
126                     print(present_cubes(cubes))
127                     msg = f'\nSuccessfully edited'
128                     input('Press return')
129             else:
130                 msg = '\nERROR: can't edit cube, side must be > 0'
131         except ValueError:
132             msg = '\nERROR: can't edit cube, wrong input'
133     else:
134         msg = '\nERROR: can't edit cube, wrong index'
135 except ValueError:
136     msg = '\nERROR: can't edit cube, wrong input'
137 return 0, msg
138
139
140 def delete_cube(cubes: [Cube]):
141     os.system('clear')
142     print(present_cubes(cubes))
143     pos = input('Type № of cube to delete: ')
144     try:
145         index = int(pos)
146         if index - 1 in range(len(cubes)):
147             cubes.pop(index - 1)
148             os.system('clear')
149             print(present_cubes(cubes))
150             msg = f'\nSuccessfully deleted'
151             input('Press return')
152         else:
153             msg = '\nERROR: can't edit cube, wrong index'
154     except ValueError:
155         msg = '\nERROR: can't edit cube, wrong input'
156 return 0, msg
157
158
159 def compare_cubes(cubes: [Cube]):
160     os.system('clear')
161     print(present_cubes(cubes))
162     pos = input('Type №№ of cubes to compare divided by space: ')
163     str_arr = pos.strip().split(' ')

```

```

164     try:
165         index1, index2 = int(str_arr[0]), int(str_arr[1])
166         if index1 - 1 in range(len(cubes)) and index2 - 1 in
range(len(cubes)):
167             cubes_to_compare = [cubes[index1 - 1], cubes[index2 - 1]]
168             os.system('clear')
169             menu = Menu([
170                 MenuItem("Back", lambda x, y: (x, y), (-1, ''))
171             ])
172             msg = (present_cubes(cubes_to_compare))
173             if cubes_to_compare[0] < cubes_to_compare[1]:
174                 msg += f'\nCube №{index1} less than cube №{index2}'
175             elif cubes_to_compare[0] > cubes_to_compare[1]:
176                 msg += f'\nCube №{index1} greater than cube №{index2}'
177             else:
178                 msg += f'\nCube №{index1} equal cube №{index2}'
179             return menu.show(avoid_clr=True, text=msg)
180         else:
181             msg = '\nERROR: can't compare cubes, wrong index'
182     except ValueError:
183         msg = '\nERROR: can't compare cubes, wrong input'
184     return 0, msg
185

```

Код исполняемого файла «main.py»

```

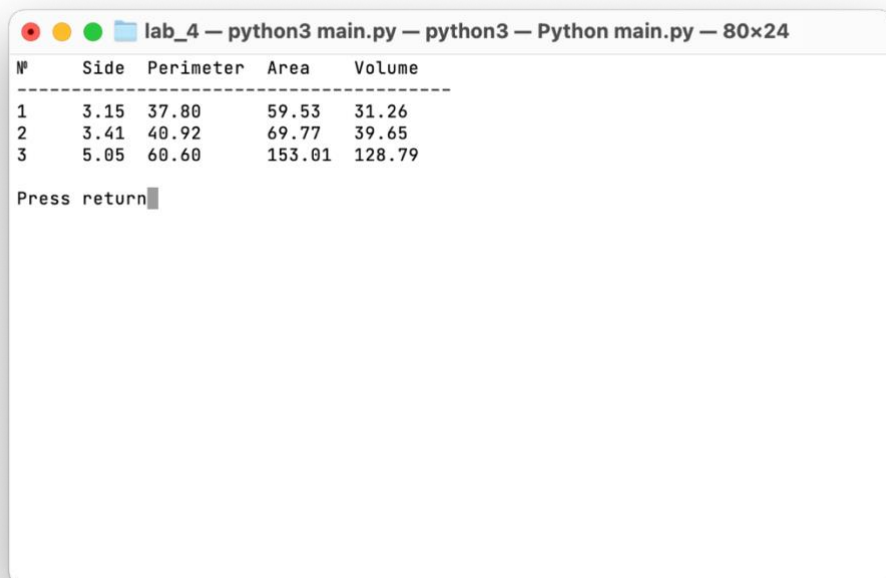
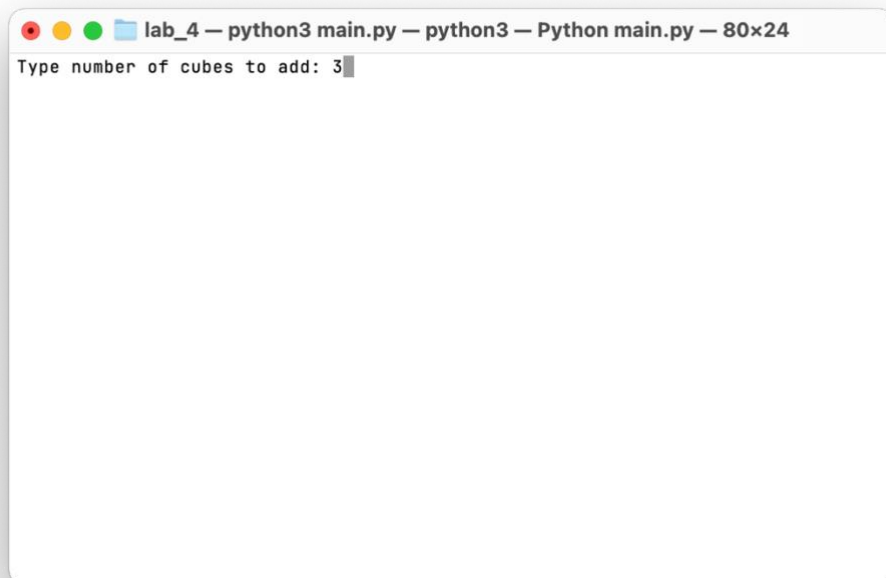
1  from Cube import Cube
2  from menu import Menu, MenuItem
3  import io_unit as iou
4
5  if __name__ == "__main__":
6      cubes: [Cube] = []
7      msg = ''
8
9      main_menu = Menu([
10         MenuItem('Load cubes from file', iou.load_cubes_from_csv, (cubes, )),
11         MenuItem('Show cubes', iou.show_cubes, (cubes,)),
12         MenuItem('Add cubes', iou.add_cubes, (cubes,)),
13         MenuItem('Edit cube', iou.edit_cube, (cubes, )),
14         MenuItem('Delete cube', iou.delete_cube, (cubes,)),
15         MenuItem('Compare cubes', iou.compare_cubes, (cubes,)),
16         MenuItem('Save changes', iou.save_to_csv, (cubes, )),
17         MenuItem('Exit', exit, (0,))
18     ])
19     main_menu.show(avoid_clr=True, info_msg=f'\n{msg}')
20

```

Тестирование и отладка

1. Добавление массива кубов

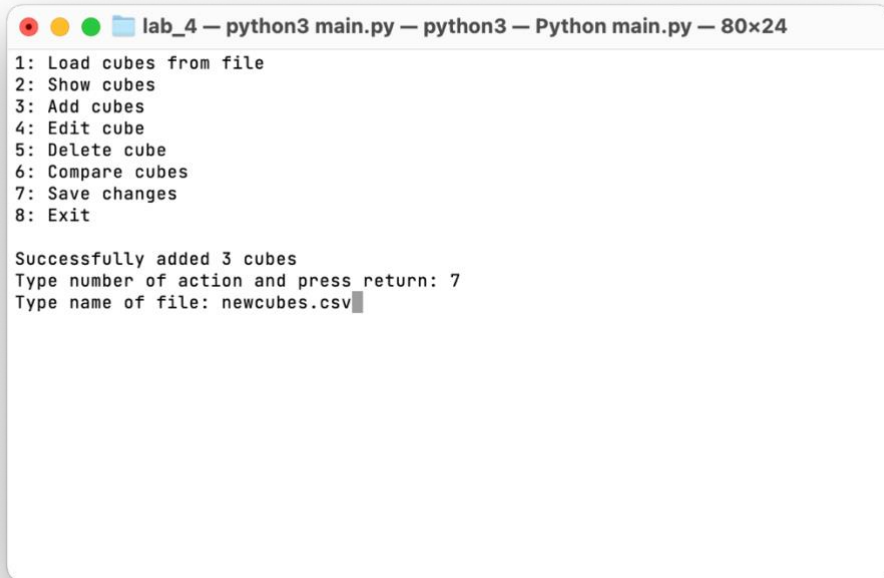
Ожидаемый результат: добавление трех кубов в массив и их отображение в табличном виде.



Вывод: кубы добавлены в массив и отображаются в табличном виде. Тест пройден.

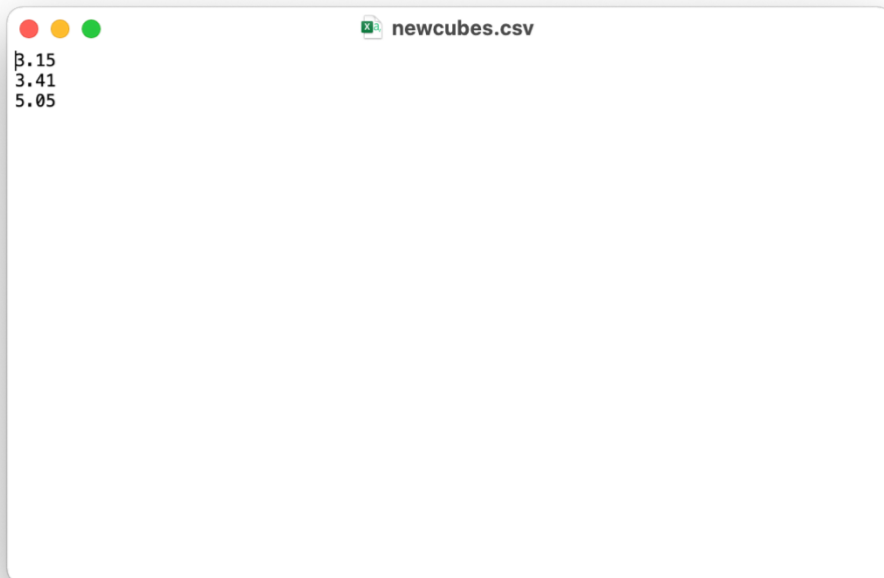
2. Сохранение кубов в файл

Ожидаемый результат: массив кубов записан в файл



```
lab_4 — python3 main.py — python3 — Python main.py — 80x24
1: Load cubes from file
2: Show cubes
3: Add cubes
4: Edit cube
5: Delete cube
6: Compare cubes
7: Save changes
8: Exit

Successfully added 3 cubes
Type number of action and press return: 7
Type name of file: newcubes.csv
```

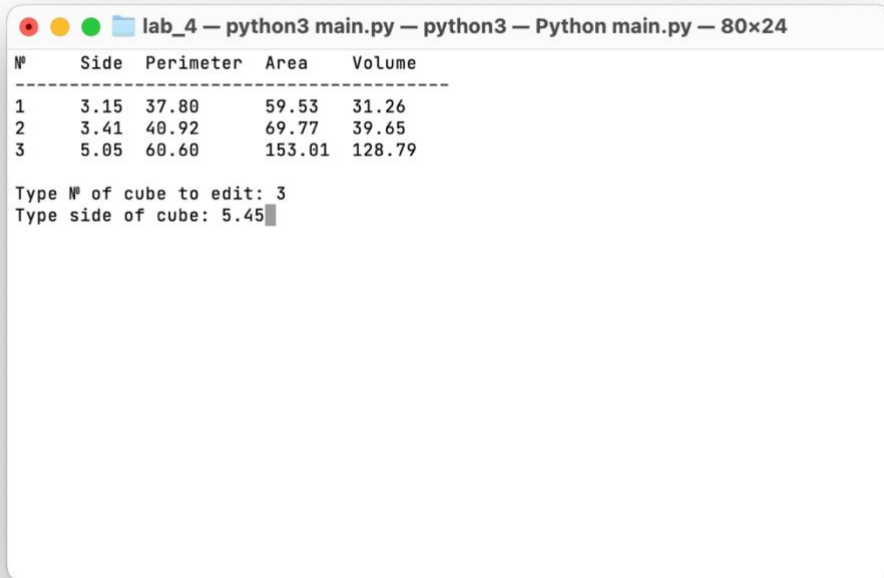


```
newcubes.csv
3.15
3.41
5.05
```

Вывод: массив кубов сохранен в файл. Тест пройден.

3. Изменение куба

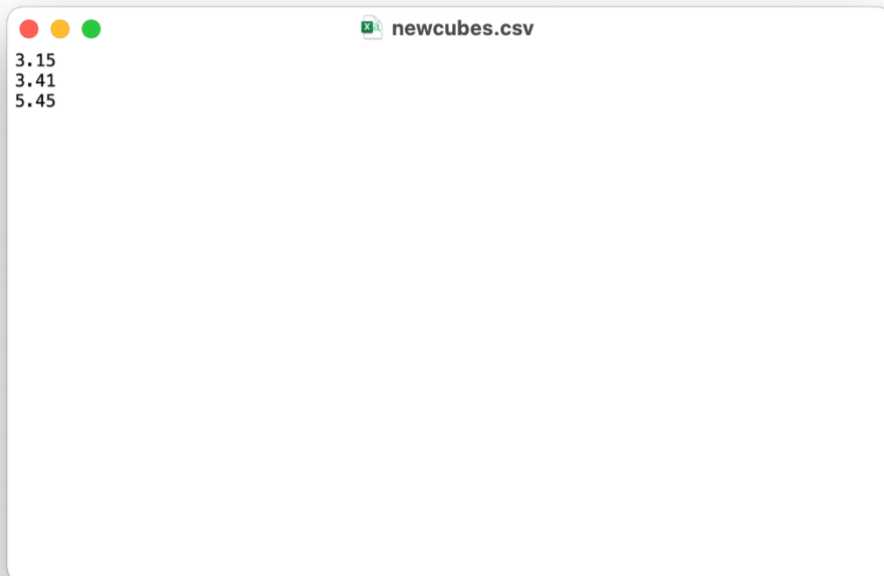
Ожидаемый результат: изменение третьего куба. Запись в файл.



```
lab_4 — python3 main.py — python3 — Python main.py — 80x24
```

№	Side	Perimeter	Area	Volume
1	3.15	37.80	59.53	31.26
2	3.41	40.92	69.77	39.65
3	5.05	60.60	153.01	128.79

Type № of cube to edit: 3
Type side of cube: 5.45



```
newcubes.csv
```

```
3.15  
3.41  
5.45
```

Вывод: куб изменен, изменения записаны в файл. Тест пройден.

4. Чтение из файла

Ожидаемый результат: загрузка из файла, вывод на экран.

Для проведения данного теста необходимо перезапустить приложение, массив кубов в оперативной памяти будет обнулен.

```
lab_4 — python3 main.py — python3 — Python main.py — 80x24
1: Load cubes from file
2: Show cubes
3: Add cubes
4: Edit cube
5: Delete cube
6: Compare cubes
7: Save changes
8: Exit

Successfully loaded 3 cubes
Type number of action and press return: █
```

```
lab_4 — python3 main.py — python3 — Python main.py — 80x24
№      Side  Perimeter  Area    Volume
-----
1       3.15   37.80     59.53   31.26
2       3.41   40.92     69.77   39.65
3       5.45   65.40     178.22  161.88

1: Back

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

Вывод: кубы считаны из файла. Тест пройден.