|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА − Российский технологический университет»**  **РТУ МИРЭА** |

**Институт информационных технологий (ИИТ)**

**Кафедра прикладной математики (ПМ)**

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ**

по дисциплине «Технологии и инструментарий анализа больших данных»

**Практическая работа № 1**

|  |  |  |  |
| --- | --- | --- | --- |
| Студент группы ИНБО-01-17 | *ИКБО-06-22, Ляхов Т. А.* | (подпись) | |
| Проверил | *Старший преподаватель,*  *Приходько Н.А.* | (подпись) | |
| Отчет представлен | «\_\_\_»\_\_\_\_\_\_\_\_2025 г. | |  | |

Москва 2025 г.

**Задание №1**

Условие: *установить Python, если это не было сделано ранее.*

Ранее уже был установлен язык программирования Python, путь до него был указан в переменной окружения PATH, а в качестве среды разработке используется программное обеспечение от JetBrains PyCharm.

**Задание №2**

Условие: *написать программу, которая вычисляет площадь фигуры, параметры которой подаются на вход. Фигуры, которые подаются на вход: треугольник, прямоугольник, круг. Результатом работы является словарь, где ключ – это название фигуры, а значение – это площадь.*

Для решения задачи был реализовать интерфейс Figure, который реализует общий для каждой фигуры функционал, а именно – вывод площади в формате словаря с ключами и значениями, не противоречащими условию задачи. Далее были реализованы три класса: Circle, Rectangle и Triangle соответственно. Так же для реализации условия задачи о вводе фигуры и ее параметров была реализована функция create\_figure, которая отвечает за инициализацию класса, переданного в потоковый ввод в консоли.

Демонстрация работы программы изображена на рисунке 1.

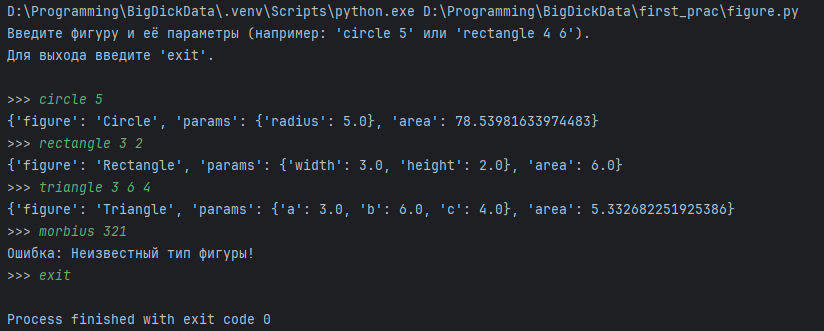


Рисунок 1 – Демонстрация работы приложения.

Код программы показан на листинге 1.

Листинг 1 – Код программы.

|  |
| --- |
| import math  class Figure:  def \_\_init\_\_(self):  self.my\_area = {}  def area(self):  raise NotImplementedError("Это виртуальный класс!")  def calculate(self):  self.my\_area = {  "figure": self.\_\_class\_\_.\_\_name\_\_,  "params": self.\_\_dict\_\_.copy(),  "area": self.area()  }  self.my\_area["params"].pop("my\_area", None)  return self.my\_area  class Rectangle(Figure):  def \_\_init\_\_(self, width, height):  super().\_\_init\_\_()  self.width = width  self.height = height  def area(self):  return self.width \* self.height  class Circle(Figure):  def \_\_init\_\_(self, radius):  super().\_\_init\_\_()  self.radius = radius  def area(self):  return self.radius \*\* 2 \* math.pi  class Triangle(Figure):  def \_\_init\_\_(self, a, b, c):  super().\_\_init\_\_()  self.a = a  self.b = b  self.c = c  def perimetr(self):  return self.a + self.b + self.c  def area(self):  p = self.perimetr() / 2  return math.sqrt(p \* (p - self.a) \* (p - self.b) \* (p - self.c))  def create\_figure(fig\_type, \*args):  figures = {  "circle": Circle,  "rectangle": Rectangle,  "triangle": Triangle  }  if fig\_type not in figures:  raise ValueError("Неизвестный тип фигуры!") |

Продолжение листинга 1.

|  |
| --- |
| return figures[fig\_type](\*args)  if \_\_name\_\_ == "\_\_main\_\_":  print("Введите фигуру и её параметры (например: 'circle 5' или 'rectangle 4 6').")  print("Для выхода введите 'exit'.\n")  while True:  line = input(">>> ").strip().lower()  if line == "exit":  break  if not line:  continue  try:  parts = line.split()  fig\_type = parts[0]  numbers = list(map(float, parts[1:]))  fig = create\_figure(fig\_type, \*numbers)  print(fig.calculate())  except Exception as e:  print(f"Ошибка: {e}") |

**Задание №3**

Условие: *написать программу, которая на вход получает два числа и операцию, которую к ним нужно применить. Должны быть реализованы следующие операции: +, -, /, //, abs – модуль, pow или \*\* – возведение в степень.*

Для реализации данной программы был создан класс Calculator, который, при инициализации, принимает в себя два числа и одну строковую переменную, отвечающую за операцию с числами. Внутри класса были реализованы методы, отвечающие за исполнение операций с числами.

Демонстрация работы программы изображена на рисунке 2.

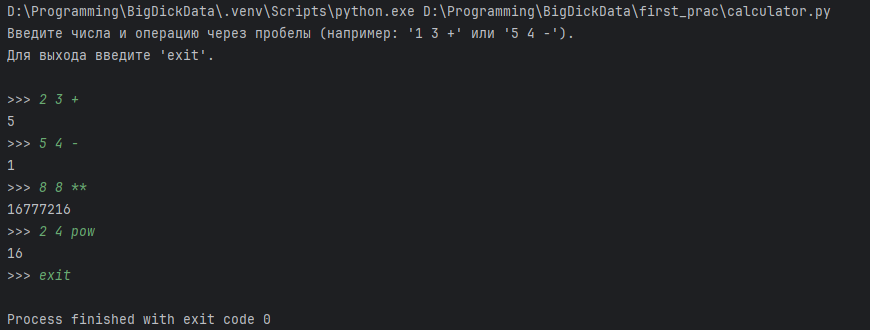


Рисунок 2 – Демонстрация работы калькулятора.

Код программы представлен в листинге 2.

Листинг 2 – Код программы-калькулятора.

|  |
| --- |
| class Calculator:  def \_\_init\_\_(self, num1, num2, optype):  self.num1 = int(num1)  self.num2 = int(num2)  self.optype = optype  @property  def plus(self):  return self.num1 + self.num2  @property  def minus(self):  return self.num1 - self.num2  @property  def multiply(self):  return self.num1 \* self.num2  @property  def divide(self):  if self.num2 == 0:  raise ValueError("Division by zero")  return self.num1 / self.num2  @property  def floordiv(self):  if self.num2 == 0:  raise ValueError("Division by zero")  return self.num1 // self.num2  @property  def absolute(self):  return f"{abs(self.num1)}, {abs(self.num2)}"  @property  def power(self):  return self.num1 \*\* self.num2  def calculate(self):  operations = {  '+': self.plus,  '-': self.minus,  '\*': self.multiply,  '/': self.divide,  '//': self.floordiv,  'abs': self.absolute,  'pow': self.power,  '\*\*': self.power,  }  if self.optype in operations:  return operations[self.optype]  else:  raise ValueError(f"Unknown operation: {self.optype}")  if \_\_name\_\_ == '\_\_main\_\_':  print("Введите числа и операцию через пробелы (например: '1 3 +' или '5 4 -').")  print("Для выхода введите 'exit'.\n")  while True: |

Продолжение листинга 2.

|  |
| --- |
| line = input(">>> ").strip().lower()  if line == "exit":  break  if not line:  continue  try:  inp = line.split()  calculator = Calculator(inp[0], inp[1], inp[2])  print(calculator.calculate())  except Exception as e:  print(f"Ошибка: {e}") |

**Задние №4**

Условие: *напишите программу, которая считывает с консоли числа (по одному в строке) до тех пор, пока сумма введённых чисел не будет равна 0 и после этого выводит сумму квадратов всех считанных чисел.*

Для этого задания была реализована простейшая программа, реализующая потоковые ввод в цикле, проверяющем, равна ли нулю сумма всех введенных чисел. Параллельно вводу, числа добавляются в массив, числа из которого, по окончании работы, возводятся в квадрат и суммируются.

Демонстрация работы программы изображена на рисунке 3.

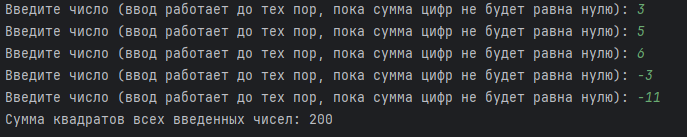


Рисунок 3 – Демонстрация работы программы сумматора

Код программы представлен на листинге 3.

Листинг 3 – Код программы сумматора.

|  |
| --- |
| nums = [] while sum(nums) != 0 or not nums:  nums.append(int(input("Введите число (ввод работает до тех пор, пока сумма цифр не будет равна нулю): "))) print("Сумма квадратов всех введенных чисел: " + str(sum(x\*x for x in nums))) |

**Задание №5**

Условие: *Напишите программу, которая выводит последовательность чисел, длинною N, где каждое число повторяется столько раз, чему оно равно. На вход программе передаётся неотрицательное целое число N. Например, если N = 7, то программа должна вывести 1 2 2 3 3 3 4. Вывод элементов списка через пробел – print(\*list).*

Для выполнения задания была написана простая программа, которая получает на вход число, которое является количеством итераций цикла, заполняющего массив с результирующими числами. Вывод осуществляется до соответствующего индекса в массиве. Демонстрация работы программы изображена на рисунке 4.

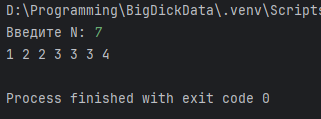


Рисунок 4 – Результат работы программы с последовательностями.

Код программы представлен на листинге 4.

Листинг 4 – Код программы с последовательностями.

|  |
| --- |
| N = int(input("Введите N: "))  result = [] num = 1  while len(result) < N:  result.extend([num] \* num)  num += 1  print(\*result[:N]) |

**Задание №6**

Условие: *Даны два списка: А = [1, 2, 3, 4, 2, 1, 3, 4, 5, 6, 5, 4, 3, 2] В = [‘a’, ’b’, ’c’, ’c’, ’c’, ’b’, ’a’, ’c’, ’a’, ’a’, ’b’, ’c’, ’b’, ’a’] Создать словарь, в котором ключи – это содержимое списка В, а значения для ключей словаря – это сумма всех элементов списка А в соответствии с буквой, содержащийся на той же позиции в списке В. Пример результата программы: {‘a’ : 10, ‘b’ : 15, ‘c’ : 6}.*

Для реализации программы были скопированы списки из условия и сформировано лямбда выражение, для сортировки и суммирования массивов.

Результат работы программы представлен на рисунке 5.

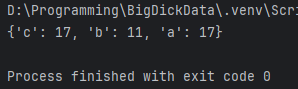


Рисунок 5 – Работа программы сортировщика массивов.

Код для работы программы сортировщика показан на листинге 5.

Листинг 5 – Код программы сортировщика массивов.

|  |
| --- |
| A = [1, 2, 3, 4, 2, 1, 3, 4, 5, 6, 5, 4, 3, 2] B = ['a', 'b', 'c', 'c', 'c', 'b', 'a', 'c', 'a', 'a', 'b', 'c', 'b', 'a'] print({k: sum(a for a, b in zip(A, B) if b == k) for k in set(B)}) |

**Задание №7–12**

Условие: скачать и загрузить данные о стоимости домов в Калифорнии, используя библиотеку sklearn. Использовать метод info(). Узнать, есть ли пропущенные значения, используя isna().sum(). Вывести записи, где средний возраст домов в районе более 50 лет и население более 2500 человек, используя метод loc(). Узнать максимальное и минимальное значения медианной стоимости дома. Используя метод apply(), вывести на экран название признака и его среднее значение.

Для выполнения задачи была написана программа, выполняющая все указанные в заданиях пункты.

Результаты работы программы представлены на рисунках 6-7.

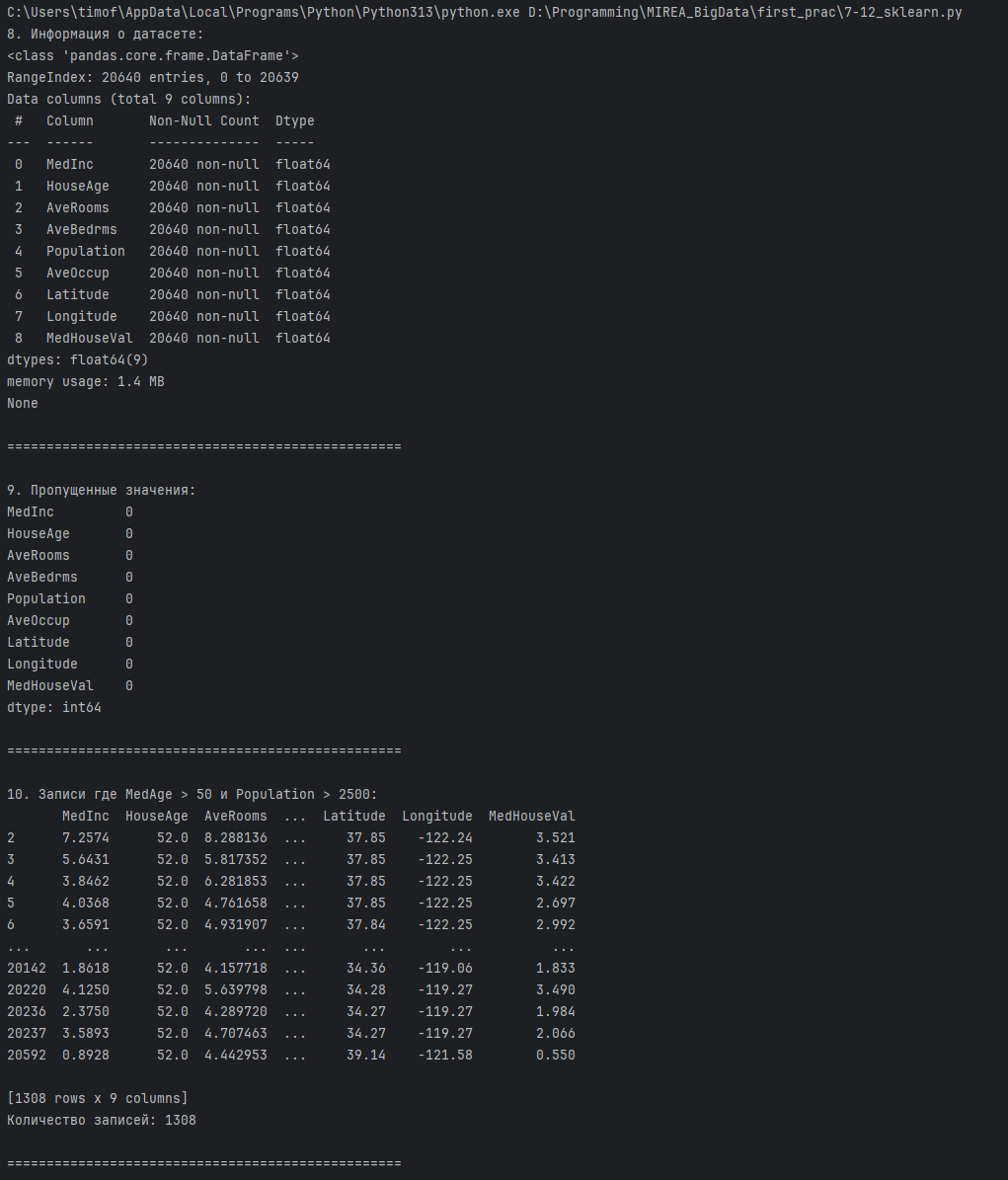


Рисунок 6 – Результат работы программы с использованием sklearn.

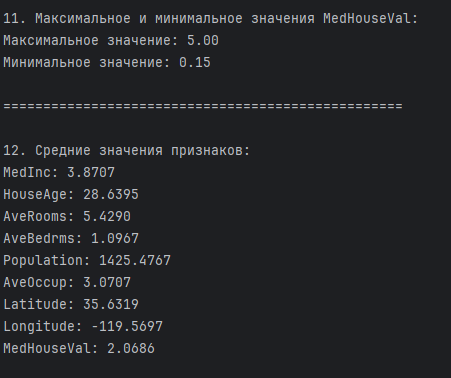


Рисунок 7 – Продолжение результата работы программы.

Код программы показан на листинге 6.

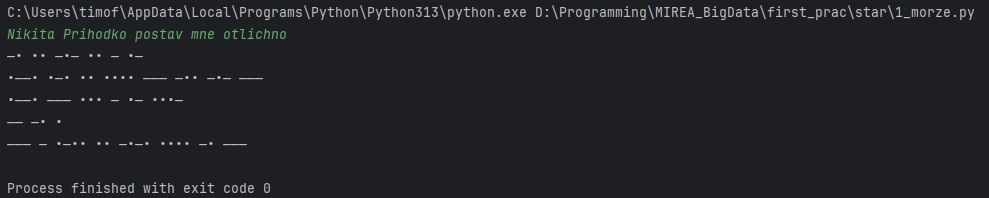
Листинг 6 – Код программы с sklearn

|  |
| --- |
| import pandas as pd  import numpy as np  from sklearn.datasets import fetch\_california\_housing  california\_housing = fetch\_california\_housing()  data = pd.DataFrame(california\_housing.data, columns=california\_housing.feature\_names)  data['MedHouseVal'] = california\_housing.target  print("8. Информация о датасете:")  print(data.info())  print("\n" + "="\*50 + "\n")  print("9. Пропущенные значения:")  print(data.isna().sum())  print("\n" + "="\*50 + "\n")  print("10. Записи где MedAge > 50 и Population > 2500:")  filtered\_data = data[(data['HouseAge'] > 50) & (data['Population'] < 2500)]  print(filtered\_data)  print(f"Количество записей: {len(filtered\_data)}")  print("\n" + "="\*50 + "\n")  print("11. Максимальное и минимальное значения MedHouseVal:")  print(f"Максимальное значение: {data['MedHouseVal'].max():.2f}")  print(f"Минимальное значение: {data['MedHouseVal'].min():.2f}")  print("\n" + "="\*50 + "\n")  print("12. Средние значения признаков:")  mean\_values = data.apply(lambda x: x.mean() if np.issubdtype(x.dtype, np.number) else None)  for feature, mean\_val in mean\_values.items():  if mean\_val is not None:  print(f"{feature}: {mean\_val:.4f}") |

**Задание №1\***

Условие: Дан текст на английском языке. Необходимо закодировать его с помощью азбуки Морзе, где каждой букве соответствует последовательность точек и тире. Например, буква «g» превратится в строку «--.». В переменной morze для удобства хранится словарь соответствия латинских букв коду Морзе. morze = {'a': '.-', 'b': '-…', 'c': '-.-.', 'd': '-..', 'e': '.', 'f': '..-.', 'g': '--.', 'h': '….', 'i': '..', 'j': '.---', 'k': '-.-', 'l': '.-..', 'm': '--', 'n': '-.', 'o': '---', 'p': '.--.', 'q': '--.-', 'r': '.-.', 's': '…', 't': '-', 'u': '..-', 'v': '…-', 'w': '.--', 'x': '-..-', 'y': '-.--', 'z': '--..'}.

Для решения данной задачи был скопирован словарь Морзе, реализована функция считывания текста и последующее сопоставление буквы с набором символов из словаря. Результат работы программы продемонстрирован на рисунке 8.

Рисунок 8 – Результат работы программы-переводчика.

Код программы представлен на листинге 7.

Листинг 7 – Код программы с последовательностями.

|  |
| --- |
| morze = {'a': '•—', 'b': '—•••', 'c': '—•—•', 'd': '—••',  'e': '•', 'f': '••—•', 'g': '——•', 'h': '••••',  'i': '••', 'j': '•———', 'k': '—•—', 'l': '•—••',  'm': '——', 'n': '—•', 'o': '———', 'p': '•——•',  'q': '——•—', 'r': '•—•', 's': '•••', 't': '—',  'u': '••—', 'v': '•••—', 'w': '•——', 'x': '—••—',  'y': '—•——', 'z': '——••'}  text = input().lower().split()  print(\*[' '.join(map(morze.get, word)) for word in text], sep='\n') |

**Задание №2\***

Условие: В некотором городе открывается новая служба по доставке электронных писем. Необходимо наладить систему регистрации новых пользователей. Регистрация должна работать следующим образом: если новый пользователь хочет зарегистрироваться на сайте, то он должен послать системе запрос name со своим именем. Система должна определить, существует ли уже такое имя в базе данных. Если такого имени не существует, то оно заносится в базу данных системы и пользователю возвращается ответ "ОК", подтверждающий успешную регистрацию. А если пользователь с таким именем уже существует, то система должна сформировать новое имя и выдать его пользователю в качестве подсказки, при этом сама подсказка также добавляется в базу данных. Новое имя формируется следующим образом: к name последовательно приписываются числа, начиная с 1 (name1, name2 и так далее), и среди них находят такое наименьшее i, что namei еще не содержится в системе.

Для решения задачи было реализовано три функции, каждая из которых запускается последовательно. Первоначально запускается цикл, считывающий количество повторений ввода имен. Далее запускается проверка на то, есть ли такое имя уже в словаре. Если нет – вернется сообщение “ОК”, в противном случае будет вызван алгоритм, проверяющий, сколько таких имен уже было. В зависимости от результата новому пользователю будет присвоено имя с порядковым номером встреченной комбинации символов. Результат работы программы представлен на рисунке 9.

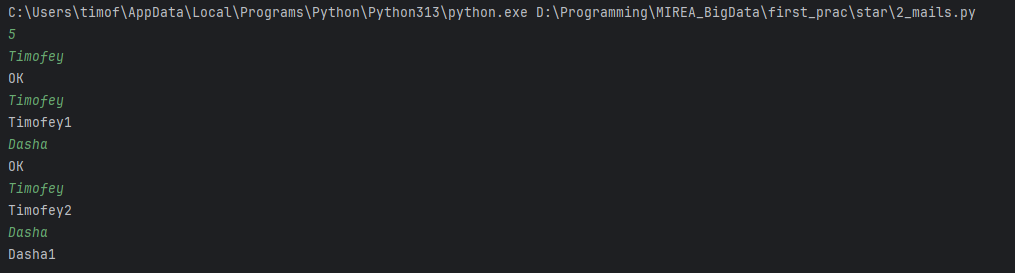


Рисунок 9 – Результат работы программы генератора имен.

Код программы представлен на листинге 8.

Листинг 8 – Код программы с последовательностями.

|  |
| --- |
| names = {}   def main(name):  if name not in names.keys(): |

Продолжение листинга 8.

|  |
| --- |
| names[name] = 1  return 'OK'  else:  return gen\_name(name)   def gen\_name(old\_name):  new\_name = f"{old\_name}{names[old\_name]}"  names[old\_name] += 1  if new\_name in names.keys():  names[new\_name] += 1  else:  names[new\_name] = 1  return new\_name   if \_\_name\_\_ == '\_\_main\_\_':  n = int(input())  for \_ in range(n):  name = input()  print(main(name)) |

**Задание №3\***

Условие: необходимо создать программу обработки запросов пользователей к файловой системе компьютера. Над каждым файлом можно производить следующие действия: запись – w ("write"), чтение – r ("read"), запуск – x ("execute").

Для реализации упрощенной системы проверки уровня доступа был реализован код, разбивающий первые n строк на массив. Первой значение, являющееся именем файла, становилось ключом, а последующие – его значением, т. е. массивом из букв “r”, “w” или “x”. После этого последующие t строк разбивались на операцию и имя файла. Операция проходила через словарь-переводчик, а потом по имени файла сравнивалась, с его уровнем доступа. Результатом было либо сообщение “OK”, либо “Access denied”. Результат работы программы показан на рисунке 10.

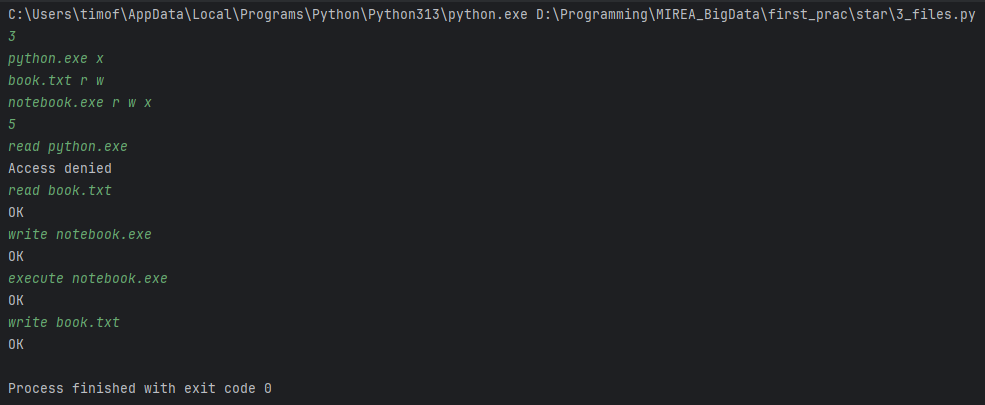


Рисунок 10 – Результат работы программы файловой системы.

Код программы представлен на листинге 9.

Листинг 9 – Код программы файловой системы.

|  |
| --- |
| os = {}  modes = {'read': 'r',  'write': 'w',  'execute': 'x', }  def gen\_files(string):  data = string.split()  name = data.pop(0)  mode = data  os[name] = mode  return os  def try\_use(string):  op, file = string.split()  if modes[op] in os[file]:  print('OK')  else:  print('Access denied')  def main():  for \_ in range(int(input())):  gen\_files(input())  for \_ in range(int(input())):  try\_use(input())  if \_\_name\_\_ == '\_\_main\_\_':  main() |

**Вывод:**

В ходе практической работы были выполнены поставленные задачи, повторены знания о языке Python и сформирован отчет.