



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение  
высшего образования

**«МИРЭА – Российский технологический университет»**

**РТУ МИРЭА**

---

Институт информационных технологий (ИТ)

Кафедра прикладной математики

**ОТЧЁТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 1**

**по дисциплине «Технологии и инструментарий анализа больших  
данных»**

Выполнил студент группы ИКБО-20-21

Фомичев Р.А.

Проверил ассистент кафедры ПМ ИИТ

Тетерин Н.Н.

Москва 2024

## Задание 1

Установить Python, если это не было сделано ранее.

## Задание 2

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.

Код к заданию 2 представлен на рисунке 1.

```
def calc_triangle(h, b):  
    return 0.5 * h * b  
  
def calc_rectangle(a, b):  
    return a * b  
  
def calc_circle(r):  
    return math.pi * (r ** 2)  
  
def ex2():  
    print("Введите номер фигуры:")  
    print("(1) - треугольник")  
    print("(2) - прямоугольник")  
    print("(3) - круг")  
  
    figure = int(input())  
    if figure == 1:  
        print("Введите высоту и основание")  
        h, b = map(float, input().split())  
        print('Площадь треугольника: ', calc_triangle(h, b))  
    elif figure == 2:  
        print("Введите высоту и ширину")  
        a, b = map(float, input().split())  
        print('Площадь прямоугольника: ', calc_rectangle(a, b))  
  
    elif figure == 3:  
        print("Введите радиус")  
        r = float(input())  
        print('Площадь круга: ', calc_circle(r))
```

Рисунок 1 – Код к заданию 1

Пример выполнения представлен на рисунке 2.

```
Введите номер фигуры:
(1) - треугольник
(2) - прямоугольник
(3) - круг
2
Введите высоту и ширину
1 3
Площадь прямоугольника: 3.0
```

Рисунок 2 – Пример работы

### Задание 3

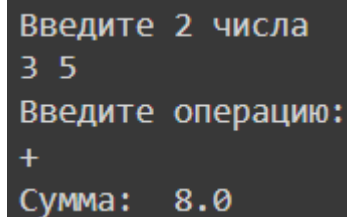
Написать программу, которая на вход получает два числа и операцию, которую к ним нужно применить. Должны быть реализованы следующие операции: +, -, /, //, abs – модуль, pow или \*\* – возведение в степень.

Код к заданию 3 представлен на рисунке 3.

```
def ex3():
    print("Введите 2 числа")
    a, b = map(float, input().split())
    print("Введите операцию: ")
    o = input()
    if o == '+':
        print("Сумма: ", a + b)
    elif o == '-':
        print("Разность: ", a - b)
    elif o == '/':
        print("Частное: ", a / b)
    elif o == '//':
        print("Целочисленное деление: ", a // b)
    elif o == 'abs':
        print("Модуль: ", abs(a), ' ', abs(b))
    elif o == 'pow' or o == '**':
        print("Степень ", a ** b)
```

Рисунок 3 – Код к заданию 3

Пример работы представлен на рисунке 4.



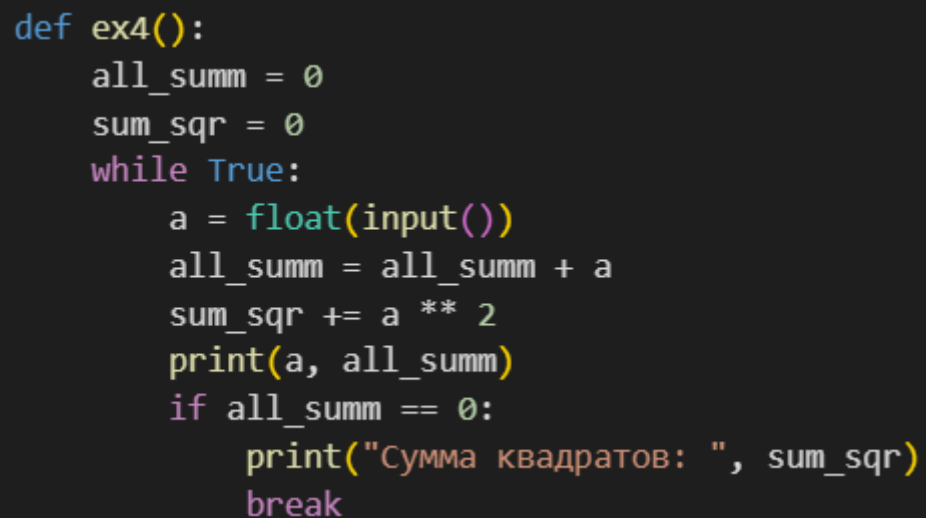
```
Введите 2 числа
3 5
Введите операцию:
+
Сумма: 8.0
```

Рисунок 4 – Пример работы

#### Задание 4

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.

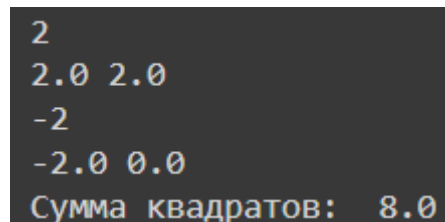
Код к заданию 4 представлен на рисунке 5.



```
def ex4():
    all_summ = 0
    sum_sqr = 0
    while True:
        a = float(input())
        all_summ = all_summ + a
        sum_sqr += a ** 2
        print(a, all_summ)
        if all_summ == 0:
            print("Сумма квадратов: ", sum_sqr)
            break
```

Рисунок 5 – Код к заданию 4

Пример работы представлен на рисунке 6.



```
2
2.0 2.0
-2
-2.0 0.0
Сумма квадратов: 8.0
```

Рисунок 6 – Пример работы

### Задание 5

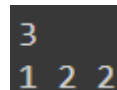
Напишите программу, которая выводит последовательность чисел, длиною N, где каждое число повторяется столько раз, чему оно равно. На вход программе передаётся неотрицательное целое число N. Например, если  $N = 7$ , то программа должна вывести 1 2 2 3 3 3 4. Вывод элементов списка через пробел – `print(*list)`.

Код к заданию 5 представлен на рисунке 7.

```
def ex5():
    n = int(input())
    sequence = []
    for i in range(1, n + 1):
        sequence.extend([i] * i)
    print(*sequence[:n])
```

Рисунок 7 – Код к заданию 5

Пример работы представлен на рисунке 8.



```
3
1 2 2
```

Рисунок 8 – Пример работы

### Задание 6

$A = [1, 2, 3, 4, 2, 1, 3, 4, 5, 6, 5, 4, 3, 2]$   $B = ['a', 'b', 'c', 'c', 'c', 'b', 'a', 'c', 'a', 'a', 'b', 'c', 'b', 'a']$  Создать словарь, в котором ключи – это содержимое списка B, а значения для ключей словаря – это сумма всех элементов списка A в соответствии с буквой, содержащийся на той же позиции в списке B. Пример результата программы: `{'a' : 10, 'b' : 15, 'c' : 6}`.

Код к заданию 6 представлен на рисунке 9.

```
def ex6():
    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']

    result = {}

    for a, b in zip(A, B):
        if b in result:
            result[b] += a
        else:
            result[b] = a

    print(result)
```

Рисунок 9 – Код к заданию 6

Пример работы представлен на рисунке 10.

```
{'a': 17, 'b': 11, 'c': 17}
```

Рисунок 10 – Пример работы

### Задание 7

Скачать и загрузить данные о стоимости домов в калифорнии, используя библиотеку sklearn.

### Задание 8

Использовать метод info().

### Задание 9

Узнать, есть ли пропущенные значения, используя isna().sum().

### Задание 10

Вывести записи, где средний возраст домов в районе более 50 лет и население более 2500 человек, используя метод loc().

### Задание 11

Узнать максимальное и минимальное значения медианной стоимости дома.

### Задание 12

Используя метод `apply()`, вывести на экран название признака и его среднее значение.

Код к заданиям 7 – 12 представлен на рисунке 11.

```
def ex7():
    # Загрузка данных
    housing_data = fetch_california_housing(as_frame=True)

    # Извлечение DataFrame и целевой переменной
    df = housing_data.frame

    # 8. Получение информации о DataFrame
    df.info()

    # 9. Проверка на пропущенные значения
    missing_values = df.isna().sum()
    print("\nПропущенные значения в каждом столбце:\n", missing_values)

    # 10. Вывод записей, где средний возраст домов > 50 лет и население > 2500 чел
    filtered_housing = df.loc[(df['HouseAge'] > 50) & (df['Population'] > 2500)]
    print("\nЗаписи с средним возрастом домов более 50 лет и населением более 2500:\n", filtered_housing)

    # 11. Узнать максимальное и минимальное значения медианной стоимости дома
    max_value = df['MedHouseVal'].max() # Используем 'MedHouseVal' для медианной стоимости
    min_value = df['MedHouseVal'].min()
    print("\nМаксимальная медианная стоимость дома:", max_value)
    print("Минимальная медианная стоимость дома:", min_value)

    # 12. Названия признаков и их средние значения
    mean_values = df.apply(lambda x: (x.name, x.mean()))
    print("\nНазвания признаков и их средние значения:\n", mean_values)
```

Рисунок 11 – Код к заданиям 7-12

Пример работы представлен на рисунках 12, 13.

```

RangeIndex: 20640 entries, 0 to 20639
Data columns (total 9 columns):
#   Column      Non-Null Count  Dtype
---  -
0   MedInc      20640 non-null  float64
1   HouseAge    20640 non-null  float64
2   AveRooms    20640 non-null  float64
3   AveBedrms   20640 non-null  float64
4   Population  20640 non-null  float64
5   AveOccup    20640 non-null  float64
6   Latitude    20640 non-null  float64
7   Longitude   20640 non-null  float64
8   MedHouseVal 20640 non-null  float64
dtypes: float64(9)
memory usage: 1.4 MB

Пропущенные значения в каждом столбце:
MedInc      0
HouseAge    0
AveRooms    0
AveBedrms   0
Population  0
AveOccup    0
Latitude    0
Longitude   0
MedHouseVal 0
dtype: int64

```

Рисунок 12 – Пример работы



```

Записи с средним возрастом домов более 50 лет и населением более 2500:
      MedInc   HouseAge   AveRooms   AveBedrms   Population   AveOccup   \
460      1.4012      52.0   3.105714   1.060000      3337.0      9.534286
4131     3.5349      52.0   4.646119   1.047945      2589.0      5.910959
4440     2.6806      52.0   4.806283   1.057592      3062.0      4.007853
5986     1.8750      52.0   4.500000   1.206349      2688.0     21.333333
7369     3.1901      52.0   4.730942   1.017937      3731.0      4.182735
8227     2.3305      52.0   3.488860   1.170380      3018.0      3.955439
13034    6.1359      52.0   8.275862   1.517241      6675.0    230.172414
15634    1.8295      52.0   2.628169   1.053521      2957.0      4.164789
15652    0.9000      52.0   2.237474   1.053535      3260.0      2.237474
15657    2.5166      52.0   2.839075   1.184049      3436.0      1.621520
15659    1.7240      52.0   2.278566   1.082348      4518.0      1.780142
15795    2.5755      52.0   3.402576   1.058776      2619.0      2.108696
15868    2.8135      52.0   4.584329   1.041169      2987.0      3.966799

      Latitude   Longitude   MedHouseVal
460         37.87    -122.26      1.75000
4131        34.13    -118.20      1.93600
4440        34.08    -118.21      1.53000
5986        34.10    -117.71      2.12500
7369        33.97    -118.21      1.67600
8227        33.78    -118.20      1.62500
13034       38.69    -121.15      2.25000
15634       37.80    -122.41      2.43800
15652       37.80    -122.41      5.00001
15657       37.79    -122.41      2.75000
15659       37.79    -122.41      2.25000
15795       37.77    -122.42      3.25000
15868       37.76    -122.41      2.60300

Максимальная медианная стоимость дома: 5.00001
Минимальная медианная стоимость дома: 0.14999

Названия признаков и их средние значения:
      MedInc   HouseAge   AveRooms   AveBedrms   Population   AveOccup   Latitude   \
0   MedInc   HouseAge   AveRooms   AveBedrms   Population   AveOccup   Latitude
1  3.870671  28.639486      5.429     1.096675  1425.476744  3.070655  35.631861

      Longitude   MedHouseVal
0   Longitude   MedHouseVal
1 -119.569704      2.068558

```

Рисунок 13 – Пример работы

### Задание 1\*

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': '--..'] }

На входе: В одной строке вам дан текст, который состоит из латинских букв и пробелов.

На выходе: Выведите каждое слово исходного текста, закодированное азбукой Морзе. Количество строк в ответе должно совпадать с количеством слов в исходном тексте. Между закодированными буквами ставится ровно один пробел. Например, слово «Help» превратится в «.... .-.. ---». Строчные и заглавные буквы кодируются одинаково.

Код программы представлен на рисунке 14.

```
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': '--..' }  
}  
  
def encode_morse(text):  
    words = text.lower().split()  
    morse_code = []  
  
    for word in words:  
        encoded_word = ' '.join(morze[char] for char in word if char in morze)  
        morse_code.append(encoded_word)  
  
    return morse_code  
  
def dopex1():  
    input_text = "Ignition sequence start"  
    output = encode_morse(input_text)  
  
    for line in output:  
        print(line)
```

Рисунок 14 – Код к заданию 1\*

Пример работы представлен на рисунке 15.



Рисунок 15 – Пример работы

### Задание 2\*

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.

Входные данные В первой строке входных данных задано число  $n$  ( $1 \leq n \leq 100000$ ). Следующие  $n$  строк содержат запросы к системе. Каждый запрос представляет собой непустую строку длиной не более 32 символов, состоящую только из строчных букв латинского алфавита.

Выходные данные В выходных данных должно содержаться  $n$  строк – ответы системы на запросы: "ОК" в случае успешной регистрации, или подсказка с новым именем, если запрашиваемое уже занято

Код к заданию 2\* представлен на рисунке 16.

```

def register_users(n, queries):
    database = {}
    results = []

    for name in queries:
        if name not in database:
            database[name] = 0
            results.append("OK")
        else:
            database[name] += 1
            new_name = f"{name}{database[name]}"
            while new_name in database:
                database[name] += 1
                new_name = f"{name}{database[name]}"
            database[new_name] = 0
            results.append(new_name)

    return results

def dopex2():
    n = int(input().strip())
    queries = [input().strip() for _ in range(n)]

    results = register_users(n, queries)

    for result in results:
        print(result)

```

Рисунок 16 – Код к заданию 2\*

Пример работы представлен на рисунке 17.

```

3
b
b
b
OK
b1
b2

```

Рисунок 17 – Пример работы

### **Задание 3\***

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").

**Входные данные** На вход программе подаются следующие параметры: число  $n$  – количество файлов в файловой системе. В следующих  $n$  строках содержится информация с именами файлов и допустимыми действиями (w, x, r), разделенных пробелами. Далее идет число  $m$  – количество запросов к файлам вида «операция файл» (обозначение операции: "write", "read", "execute").

**Выходные данные** Для каждого допустимого запроса программа должна возвращать ОК, для недопустимого – Access denied.

Код к заданию 3\* представлен на рисунке 18.

```

def process_file_requests(n, file_permissions, m, requests):
    # Словарь для хранения разрешений
    file_access = {}

    # Заполнение
    for line in file_permissions:
        parts = line.split()
        filename = parts[0]
        permissions = set(parts[1:])
        file_access[filename] = permissions

    results = []

    for request in requests:
        operation, filename = request.split()
        if operation == 'write':
            operation_let = 'w'
        elif operation == 'read':
            operation_let = 'r'
        elif operation == 'execute':
            operation_let = 'x'
        else:
            operation_let = 'none'

        if filename in file_access and operation_let in file_access[filename]:
            results.append("OK")
        else:
            results.append("Access denied")

    return results

def dopex3():
    n = int(input().strip())
    file_permissions = [input().strip() for _ in range(n)]
    m = int(input().strip())
    requests = [input().strip() for _ in range(m)]

    results = process_file_requests(n, file_permissions, m, requests)

    for result in results:
        print(result)

```

Рисунок 18 – Код к заданию 3\*

Пример работы представлен на рисунке 19.

```

3
python.exe x
book.txt r w
notebook.exe r w x
2
read python.exe
read book.txt
Access denied
OK

```

Рисунок 19 – Пример работы