



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

Институт Информационных Технологий
Кафедра Прикладной Математики

Практическая работа №1

по дисциплине
«Технологии и инструментарий анализа больших данных»

Студент группы ИКБО-04-22

Егоров Л.А.
(Ф.И.О. студента)

Принял

Царёв Р.Ю.
(Ф.И.О. преподавателя)

Москва 2025

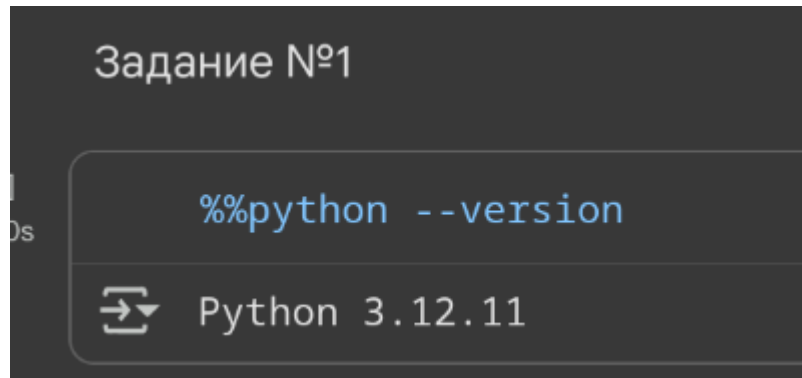
СОДЕРЖАНИЕ

| | |
|------------------------|----|
| 1 ОСНОВЫ PYTHON | 3 |
| 1.1 Задание №1 | 3 |
| 1.2 Задание №2 | 3 |
| 1.3 Задание №3 | 3 |
| 1.4 Задание №4 | 4 |
| 1.5 Задание №5 | 4 |
| 1.6 Задание №6 | 5 |
| 1.7 Задание №7 | 6 |
| 1.8 Задание №8 | 6 |
| 1.9 Задание №9 | 6 |
| 1.10 Задание №10 | 7 |
| 1.11 Задание №11 | 8 |
| 1.12 Задание №12 | 9 |
| 1.13 Задание №1* | 9 |
| 1.14 Задание №2* | 10 |
| 1.15 Задание №3* | 11 |

1 ОСНОВЫ PYTHON

1.1 Задание №1

Для выполнения практических работ используется Google Colab с предустановленным Python и некоторыми библиотеками (Рисунок 1.1).



```
Задание №1

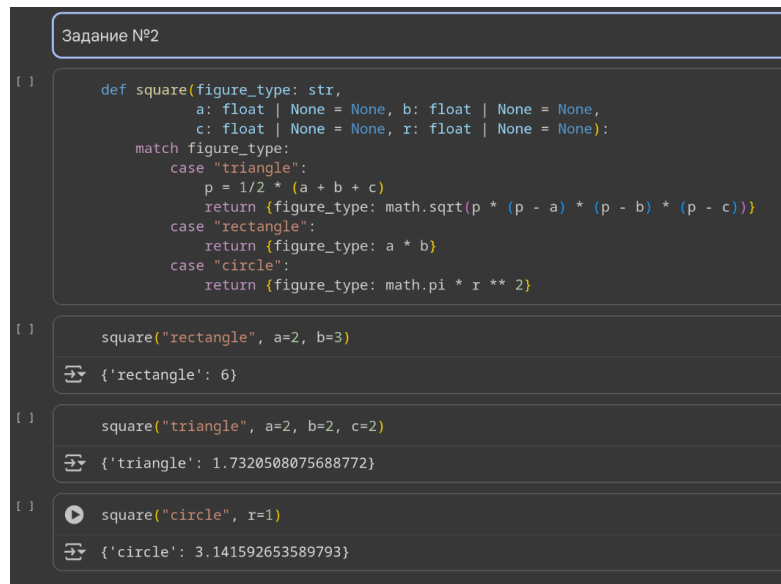
%%python --version

Python 3.12.11
```

Рисунок 1.1 — Установленная версия Python

1.2 Задание №2

Написана функция, позволяющая для разных фигур вычислять площадь (Рисунок 1.2).



```
Задание №2

def square(figure_type: str,
            a: float | None = None, b: float | None = None,
            c: float | None = None, r: float | None = None):
    match figure_type:
        case "triangle":
            p = 1/2 * (a + b + c)
            return {figure_type: math.sqrt(p * (p - a) * (p - b) * (p - c))}
        case "rectangle":
            return {figure_type: a * b}
        case "circle":
            return {figure_type: math.pi * r ** 2}

square("rectangle", a=2, b=3)
{'rectangle': 6}

square("triangle", a=2, b=2, c=2)
{'triangle': 1.7320508075688772}

square("circle", r=1)
{'circle': 3.141592653589793}
```

Рисунок 1.2 — Функция вычисления площади

1.3 Задание №3

Написана программа-калькулятор (Рисунок 1.3).

```
Задание №3

[ ] def calculator(operation: str, operand1: float, operand2: float | None = None):
    match operation:
        case '+':
            return operand1 + operand2
        case '-':
            return operand1 - operand2
        case '*':
            return operand1 * operand2
        case '/':
            return operand1 // operand2
        case '/':
            return operand1 / operand2
        case 'abs':
            return abs(operand1)
        case 'pow':
            return operand1 ** operand2

[ ] calculator("+", 77, 33)
    110

[ ] calculator("pow", 2, 10)
    1024

[ ] calculator("abs", -3)
    3
```

Рисунок 1.3 — Программа-калькулятор

1.4 Задание №4

Написана программа, 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 (Рисунок 1.4).

```
Задание №4

[4] def square_sum():
    s = int(input())
    result = s ** 2
    while s != 0:
        n = int(input())
        s += n
        result += n ** 2
    return result

[5] print("Result: ", square_sum())
    1
    2
    -3
    Result: 14
```

Рисунок 1.4 — Считывание чисел из пользовательского ввода

1.5 Задание №5

Написана программа, которая выводит последовательность чисел, длиною N, где каждое число повторяется столько раз, чему оно равно (Рисунок 1.5).

Задание №5

```
def print_repeating_numbers(n: int):  
    i = 1  
    result = []  
    while n > 0:  
        result.extend([i] * min(i, n))  
        n -= i  
        i += 1  
    print(*result)
```

```
print_repeating_numbers(13)
```

```
➞ 1 2 2 3 3 3 4 4 4 4 5 5 5
```

Рисунок 1.5 — Вывод последовательности чисел

1.6 Задание №6

Написана программа, где требуется 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 (Рисунок 1.6).

Задание №6

```
def accumulate(a: list[int], b: list[str]) -> dict[str, int]:  
    if len(a) != len(b): raise Exception("Не совпадает длина списков")  
  
    result_dict = {}  
    for i in range(len(b)):  
        result_dict[b[i]] = result_dict.get(b[i], 0) + a[i]  
    return result_dict
```

```
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(accumulate(a, b))
```

```
➞ {'a': 17, 'b': 11, 'c': 17}
```

Рисунок 1.6 — Создание словаря

1.7 Задание №7

Выгружены данные о стоимости домов в Калифорнии (Рисунок 1.7).

Задание №7

```
from sklearn.datasets import fetch_california_housing
data = fetch_california_housing(as_frame=True)
```

Рисунок 1.7 — Стоимость домов в Калифорнии

1.8 Задание №8

Выведена информация о датасете (Рисунок 1.8)

Задание №8

[]

```
data.frame.info()
```



```
<class 'pandas.core.frame.DataFrame'>
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
```

Рисунок 1.8 — Информация о датасете

1.9 Задание №9

Пропущенных значений в датасете не обнаружено (Рисунок 1.9).

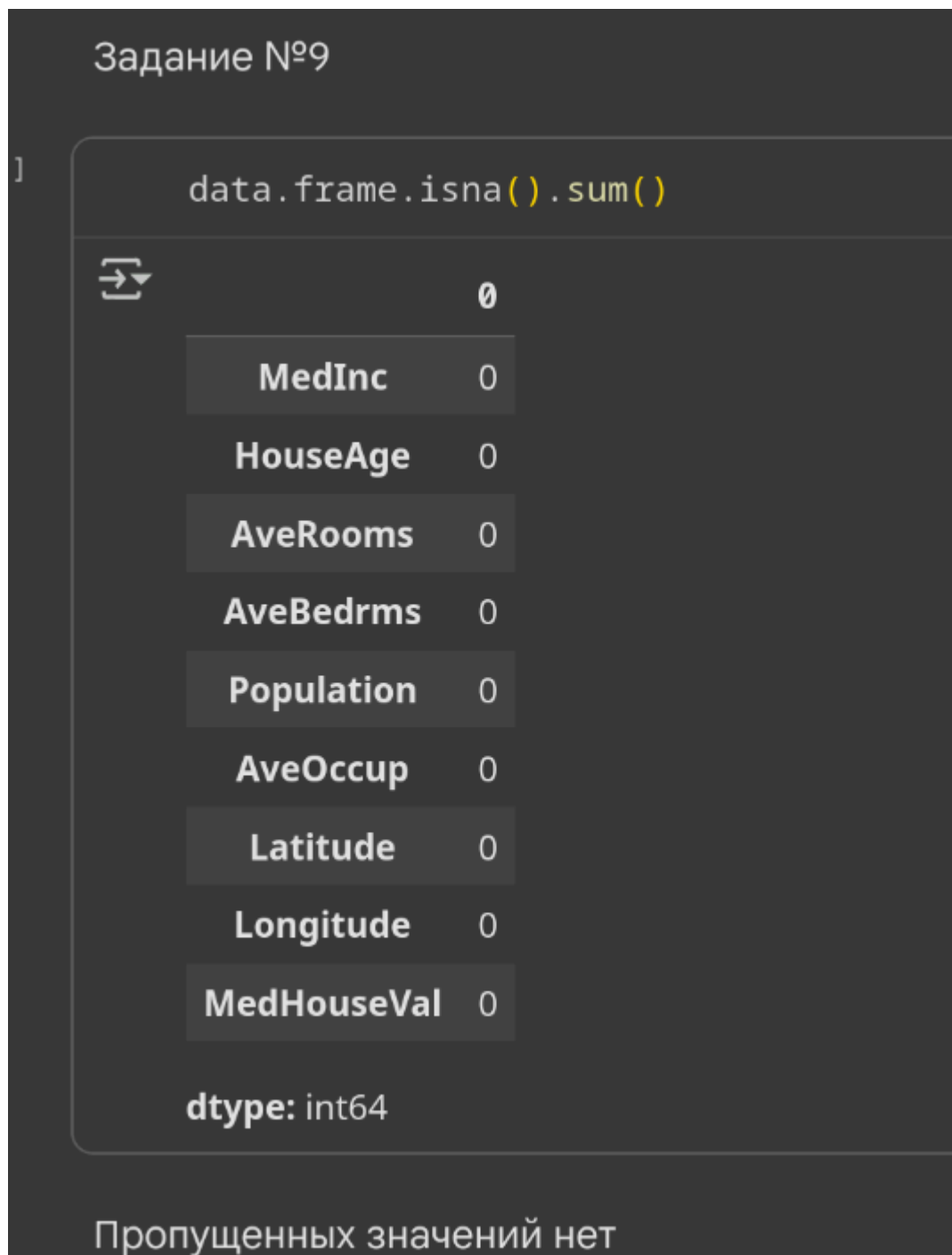


Рисунок 1.9 — Поиск пропущенных значений

1.10 Задание №10

Выведены записи, где средний возраст домов больше 50 лет и население более 2500 человек (Рисунок 1.10).

Задание №10

```
df = data.frame
df.loc[(df.HouseAge > 50) & (df.Population > 2500)]
```

| | MedInc | HouseAge | AveRooms | AveBedrms | Population | AveOccup | Latitude | Longitude | MedHouseVal |
|-------|--------|----------|----------|-----------|------------|------------|----------|-----------|-------------|
| 460 | 1.4012 | 52.0 | 3.105714 | 1.060000 | 3337.0 | 9.534286 | 37.87 | -122.26 | 1.75000 |
| 4131 | 3.5349 | 52.0 | 4.646119 | 1.047945 | 2589.0 | 5.910959 | 34.13 | -118.20 | 1.93600 |
| 4440 | 2.6806 | 52.0 | 4.806283 | 1.057592 | 3062.0 | 4.007853 | 34.08 | -118.21 | 1.53000 |
| 5986 | 1.8750 | 52.0 | 4.500000 | 1.206349 | 2688.0 | 21.333333 | 34.10 | -117.71 | 2.12500 |
| 7369 | 3.1901 | 52.0 | 4.730942 | 1.017937 | 3731.0 | 4.182735 | 33.97 | -118.21 | 1.67600 |
| 8227 | 2.3305 | 52.0 | 3.488860 | 1.170380 | 3018.0 | 3.955439 | 33.78 | -118.20 | 1.62500 |
| 13034 | 6.1359 | 52.0 | 8.275862 | 1.517241 | 6675.0 | 230.172414 | 38.69 | -121.15 | 2.25000 |
| 15634 | 1.8295 | 52.0 | 2.628169 | 1.053521 | 2957.0 | 4.164789 | 37.80 | -122.41 | 2.43800 |
| 15652 | 0.9000 | 52.0 | 2.237474 | 1.053535 | 3260.0 | 2.237474 | 37.80 | -122.41 | 5.00001 |
| 15657 | 2.5166 | 52.0 | 2.839075 | 1.184049 | 3436.0 | 1.621520 | 37.79 | -122.41 | 2.75000 |
| 15659 | 1.7240 | 52.0 | 2.278566 | 1.082348 | 4518.0 | 1.780142 | 37.79 | -122.41 | 2.25000 |
| 15795 | 2.5755 | 52.0 | 3.402576 | 1.058776 | 2619.0 | 2.108696 | 37.77 | -122.42 | 3.25000 |
| 15868 | 2.8135 | 52.0 | 4.584329 | 1.041169 | 2987.0 | 3.966799 | 37.76 | -122.41 | 2.60300 |

Рисунок 1.10 — Поиск по условию

1.11 Задание №11

Выведены максимальное и минимальное значения медианной стоимости домов (Рисунок 1.11).

Задание №11

| |
|-----------------------------------|
| <code>df.MedHouseVal.max()</code> |
| ⇒ 5.00001 |
| <code>df.MedHouseVal.min()</code> |
| ⇒ 0.14999 |

Рисунок 1.11 — Максимальная и минимальная стоимость медианной стоимости

1.12 Задание №12

Написана функция для вычисления среднего арифметического и применена к датасету с помощью метода `.apply()` (Рисунок 1.12).

```
Задание №12

def mean_info(x):
    print(x.name, x.mean())
    return x.mean()

df.apply(mean_info, axis=0)
```

MedInc 3.8706710029069766
HouseAge 28.639486434108527
AveRooms 5.428999742190376
AveBedrms 1.096675149606208
Population 1425.4767441860465
AveOccup 3.0706551594363742
Latitude 35.63186143410853
Longitude -119.56970445736432
MedHouseVal 2.068558169089147

| | 0 |
|-------------|-------------|
| MedInc | 3.870671 |
| HouseAge | 28.639486 |
| AveRooms | 5.429000 |
| AveBedrms | 1.096675 |
| Population | 1425.476744 |
| AveOccup | 3.070655 |
| Latitude | 35.631861 |
| Longitude | -119.569704 |
| MedHouseVal | 2.068558 |

dtype: float64

Рисунок 1.12 — Применение метода `.apply()`

1.13 Задание №1*

Написана функция, переводящая сообщение в код Морзе (Рисунок 1.13).

Задание №1*

```
def morze_encode(s: str):
    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': '- - . .'}

    result = []
    for word in s.lower().split():
        encoded_word = []
        for symbol in word:
            encoded_word.append(morze[symbol])
        result.append(' '.join(encoded_word))
    return '\n'.join(result)

print(morze_encode("Ignition sequence start"))
```

⌂

```
.. - . - . . - . . - - .
. . . . - . . . . . . .
. . - . - . . . -
```

Рисунок 1.13 — Код Морзе

1.14 Задание №2*

Написана программа, проверяющая вводимые имена пользователей на уникальность (Рисунок 1.14).

```
[ ] def check_unique_names(n: int):
    database = {}
    for _ in range(n):
        name = input()
        if name not in database:
            print('OK')
        else:
            print(name + str(database[name]))
            database[name] = database.get(name, 0) + 1

[ ] check_unique_names(3)

⇒ b
OK
b
b1
b
b2

[ ] check_unique_names(10)

⇒ fpqh fouqlddravpjttarh
OK
fpqh fouqlddravpjttarh
fpqh fouqlddravpjttarh1
fpqh fouqlddravpjttarh
fpqh fouqlddravpjttarh2
fpqh fouqlddravpjttarh
fpqh fouqlddravpjttarh3
fpqh fouqlddravpjttarh
fpqh fouqlddravpjttarh4
fpqh fouqlddravpjttarh
fpqh fouqlddravpjttarh5
jmv lplnrmba
OK
fpqh fouqlddravpjttarh
fpqh fouqlddravpjttarh6
jmv lplnrmba
jmv lplnrmba1
fpqh fouqlddravpjttarh
fpqh fouqlddravpjttarh7
```

Рисунок 1.14 — Проверка имён на уникальность

1.15 Задание №3*

Написана программа, проверяющая доступ к выполнению операций над различными файлами (Рисунок 1.15).

Задание №3*

```
def manage_files():
    mode_dict = {"w": [], "r": [], "x": []}
    n = int(input())
    for _ in range(n):
        _file, *modes = input().split()
        for mode in modes:
            mode_dict[mode].append(_file)

    query_count = int(input())
    for _ in range(query_count):
        action, _file = input().split()
        mode = {"read": "r", "write": "w", "execute": "x"}[action]
        if _file not in mode_dict[mode]:
            print('Access denied')
        else:
            print('OK')
```

▶ manage_files()

```
⇒ 3
python.exe x
book.txt r w
notebook.exe r w x
5
read python.exe
Access denied
read book.txt
OK
write notebook.exe
OK
execute notebook.exe
OK
write book.txt
OK
```

Рисунок 1.15 — Управление файлами