

Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ  
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

УТВЕРЖДАЮ

Зав.кафедрой,

к. ф.-м. н.

\_\_\_\_\_ М. В. Огнева

**ОТЧЕТ О ПРАКТИКЕ**

студента 2 курса 242 группы факультета КНиИТ

Бурдавицына Артёма Андреевича

вид практики: учебная

кафедра: информатики и программирования

курс: 2

семестр: 4(5)

продолжительность: 2 нед., с 24.06.2018 г. по 05.07.2018 г.

Руководитель практики от университета,

\_\_\_\_\_

А. А. Казачкова

Тема практики: «Работа с данными в Python»

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ .....	4
1 История языка Python .....	5
2 Основы синтаксиса Python .....	6
2.1 Общие сведения .....	6
2.2 Условный оператор if-else .....	6
2.3 Циклы .....	7
2.3.1 Цикл с параметром (for) .....	7
2.3.2 Цикл с предусловием (while) .....	7
2.4 Функции .....	8
3 Структуры данных в Python .....	10
3.1 Строки .....	10
3.2 Списки .....	12
4 Библиотека Random .....	13
5 Библиотека NumPy .....	17
6 Библиотека MATPLOTLIB .....	19
7 Библиотека Pandas .....	21
8 Финальная задача .....	22
ЗАКЛЮЧЕНИЕ .....	27
Приложение А Вектор .....	28
Приложение Б Циклы .....	30
Приложение В Функции .....	31
Приложение Г Строки .....	32
Приложение Д Списки .....	33
Приложение Е Библиотека Random .....	34
Приложение Ж Библиотека NumPy .....	37
Приложение З Библиотека Mathplotlib .....	40
Приложение И Библиотека Pandas .....	42
Приложение К Финальная задача .....	44

## ВВЕДЕНИЕ

Python - высокоуровневый скриптовый объектно-ориентированный язык программирования, разработанный Гвидо ван Россумом и Python Software Foundation в 1991 году. Его отличительной особенностью является интуитивный синтаксис и быстрое действие.

Главной целью учебной практики является изучение синтаксиса Python и математических библиотек Numpy и Pandas.

Основные задачи практики:

- изучение синтаксиса и основных конструкций языка Python;
- изучение методов и возможностей библиотек NumPy, Pandas, Matplotlib.
- применение полученных знаний для решения поставленных задач, а также для самостоятельной обработки и анализа набора данных из открытых источников.

## 1 История языка Python

История языка программирования Python началась в конце 1980-х. Гвидо ван Россум задумал Python в 1980-х годах, а приступил к его созданию в декабре 1989 года в центре математики и информатики в Нидерландах. Язык Python был задуман как потомок языка программирования ABC, способный к обработке исключений и взаимодействию с операционной системой Амёба. Ван Россум является основным автором Python и по сей день продолжает выполнять центральную роль в принятии решений относительно развития языка.

Версия Python 2.0 была выпущена 16 октября 2000 года и включала в себя много новых крупных функций — таких как полный сборщик мусора и поддержка Unicode. Однако наиболее важным из всех изменений было изменение самого процесса развития языка и переход на более прозрачный процесс его создания. Первая обратно-несовместимая версия Python 3.0 была выпущена 3 декабря 2008 года после длительного периода тестирования. Многие её функции были портированы в обратно совместимые Python 2.6 и Python 2.7.

Python поддерживает несколько парадигм программирования: структурное, объектноориентированное, функциональное, императивное, аспектно-ориентированное. Основные архитектурные черты — динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений и удобные высокоуровневые структуры данных. Код в Python организовывается в функции и классы, которые могут объединяться в модули (они в свою очередь могут быть объединены в пакеты).

Python даёт возможность писать компактные и читабельные программы. Программы, написанные на Python отличаются большей краткостью чем эквиваленты на C, C++ или Java, по нескольким причинам:

- высокоуровневые типы данных позволяют вам выражать сложные операции в одной инструкции;
- группировка инструкций выполняется отступами, а не операторными скобками;
- нет необходимости в описании переменных или аргументов.

## 2 Основы синтаксиса Python

### 2.1 Общие сведения

Python - интерпретируемый объектно - ориентированный язык программирования. Его основным достоинством является нативный синтаксис и малое количество служебных слов. К основам Python можно отнести работу с условными операторами, циклами и функциями.

Основы синтаксиса Python:

- не требуется точка с запятой в конце строки;
- группировка отступов выполняется посредством отступов;
- не обязательно описывать типы переменных.

### 2.2 Условный оператор if-else

Условные операторы служат для определения поведения программы в зависимости от некоторых условий. Данный оператор является основным элементом выбора в языке Python.

Задача 1

Дана точка на плоскости с координатами  $(x, y)$ . Составить программу, которая выдает одно из сообщений «Да», «Нет», «На границе» в зависимости от того, лежит ли точка внутри заштрихованной области, вне заштрихованной области или на ее границе. Области задаются графически следующим образом:

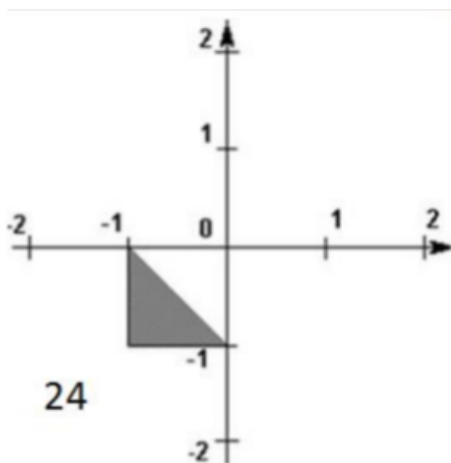


Рисунок 2.1 – Рисунок 1

В данном фрагменте задачи есть 2 условия. Если координаты точки удовлетворяют всем неравенствам, то точка находится внутри. Это описывается в if-части условного оператора. В elif-части условного оператора описываются условия принадлежности точки к пространству вне заштрихованной области.

Если координаты точки удовлетворяют этому условию, то точка не принадлежит заштрихованной области. В противном случае, при невыполнении ни одного из условий получается, что точка лежит на границе. Это описано в else-части.

Решение задачи:

```
.....
if (y<-x-1) and (x>-1) and (y>-1):
    print("Да");
elif not((y<=-x-1) and (x>=-1) and (y>=-1)):
    print("Нет");
else:
    print('На границе');
.....
```

Полный код программы приведен в приложении [А](#).

## 2.3 Циклы

В языке Python существуют 3 вида циклов: цикл с параметром (for), цикл с предусловием (while), цикл с постусловием (do while).

### 2.3.1 Цикл с параметром (for)

В цикле for указывается переменная и множество значений, по которому будет пробегать переменная. Множество значений может быть задано списком, кортежем, строкой или диапазоном. Как правило, циклы for используются либо для повторения какой-либо последовательности действий заданное число раз, либо для изменения значения переменной в цикле от некоторого начального значения до некоторого конечного. Для повторения цикла некоторое заданное число раз  $n$  можно использовать цикл for вместе с функцией range.

### 2.3.2 Цикл с предусловием (while)

Цикл while выполняется пока по логическое выражение в скобках (условие будет истинным). Этим данный оператор похож на оператор if, исполняемый много раз. Цикл завершает свою работу только тогда, когда логическое выражение в заголовке возвращает ложь, то есть условие выполнения цикла больше не соблюдается. После этого поток выполнения перемещается к выражениям, расположенным ниже всего цикла.

## Задача 2

Даны два отрезка А и В ( $A > B$ ). Не используя операции умножения и деления, определить, сколько отрезков В уместится в отрезке А.

Для решения задачи будем умножать число В на счетчик, являющийся натуральным числом. Если на данной итерации цикла число А всё еще больше числа В, то увеличиваем счетчик на единицу. Решение задачи:

```
.....
while (a>s*b):
    s+=1;

    if (s*b > a):
        s-=1;

print(s);
.....
```

Полный код программы приведен в приложении **Б**.

## 2.4 Функции

По своей сути функции в Python практически ничем не отличаются от функций из других языков программирования. Функцией называют именованный фрагмент программного кода, к которому можно обратиться из другого места вашей программы. Функции создаются для работы с данными, которые передаются ей в качестве аргументов, также функция может формировать некоторое возвращаемое значение. Для создания функции используется ключевое слово `def`, после которого указывается имя и список аргументов в круглых скобках. Тело функции выделяется также как тело условия (или цикла): четырьмя пробелами.

## Задача 3

Разработать функцию, которая по заданному  $n$  возвращает список  $n$  первых членов заданной последовательности:  $b_1 = 2.3$ ,  $b_2 = -5$ ,  $b_n = b_{n-2} + 2b_{n-1}$ .

Разработаем соответствующую функцию. Зададим начальные значения и положим их в возвращаемый список. В цикле `for` посчитаем по рекуррентной формуле и запишем в список следующие  $(n-2)$  члена заданной последовательности.



### Решение задачи:

```
.....  
def f(n):  
    a = list();  
  
    a.append(2.3);  
    a.append(-5);  
    b1 = 2.3;  
    b2 = -5;  
  
    for i in range(2,n):  
        ans = b1 + 2*b2;  
        a.append(ans);  
        b1 = b2;  
        b2 = ans;  
  
    return a;  
.....
```

Полный код программы приведен в приложении **В**.

## 3 Структуры данных в Python

### 3.1 Строки

Существует несколько литералов строк. Строки в апострофах и в кавычках - одно и то же. Причина наличия двух вариантов в том, чтобы позволить вставлять в литералы строк символы кавычек или апострофов, не используя экранирование. Пример:

```
.....  
S = 'spam"s'  
S = "spam's";  
.....
```

Существуют строки в тройных апострофах или кавычках. Главное достоинство строк в тройных кавычках в том, что их можно использовать для записи многострочных блоков текста. Внутри такой строки возможно присутствие кавычек и апострофов, главное, чтобы не было трех кавычек подряд. Пример:

```
.....  
>>> c = '''это очень большая  
... строка, многострочный  
... блок текста'''  
>>> c  
'это очень большая\nстрока, многострочный\nблок текста'  
>>> print(c)  
это очень большая  
строка, многострочный  
        блок текста  
.....
```

Основные операции для работы со строками представлены в таблице.

#### Задача 4

Удалить из строки слова, содержащие повторяющиеся символы

Решение задачи:

Введем строку. Удалим лишние символы, оставив только буквы и пробельные символы. Для этого воспользуемся функциями `split` и `join`. `Split` разобьет функцию согласно уловному выражению в скобках (`x for x in str if x.isalpha() or x == ' '`), оставив только пробельные символы и буквы. `Join` соединит слова в одну строку.

+	Конкатенация (сложение строк)
*	Умножение строки на число
[]	Получение элемента по индексу
[x:y:h]	Срез, где x - номер элемента, с которого начинается срез, y - номер первого элемента, не вошедшего в срез, h - шаг среза
len()	Длина строки
find(подстрока, [начало], [конец])	Возвращает индекс первого вхождения или -1.
rfind()	Возвращает индекс последнего вхождения или -1.
startswith(шаблон)	Начинается ли строка с с указанного шаблона;
endswith(шаблон)	Заканчивается ли строка шаблоном;
replace(шаблон, замена)	Замена шаблона
split(разделитель)	Разбиение строки по разделителю
join()	Обратная операция split()
isalpha()	Состоит ли строка из букв
isalnum()	Состоит ли строка из цифр и букв
islower()	Состоит ли строка из букв в нижнем регистре
isupper()	Состоит ли строка из букв в верхнем регистре
isspace()	Состоит ли строка из пробельных символов
istitle()	Начинаются ли слова в строке с заглавной буквы.
count(шаблон, [начало], [конец])	Подсчет непересекающихся вхождений шаблона в строку

```
.....
words = ''.join(x for x in str if x.isalpha() or x == ' ').split();
.....
```

В цикле Приведем все символы к нижнему регистру с помощью функции `lower()`. Положим их в `set`. Для каждого элемента сета будем проверять заданное условием задачи условие.

```
.....
for sub_str in words[:]:

    sub_set = set(sub_str.lower());
```

```

        if not(len(sub_set) == len(sub_str)):
            words.remove(sub_str);

words1 = ''.join(words);
.....

```

Полный код программы приведен в приложении **Г**.

### 3.2 Списки

Список – это структура данных для хранения объектов различных типов. Список очень похож на массив, только, в нем можно хранить объекты различных типов. Размер списка не статичен, его можно изменять. Список по своей природе является изменяемым типом данных. Переменная, определяемая как список, содержит ссылку на структуру в памяти, которая в свою очередь хранит ссылки на какие-либо другие объекты или структуры.

#### Задача 5

Дан одномерный массив чисел. После максимального из четных элементов вставить 0. Дан одномерный массив чисел. Найти/определить максимальный из элементов, имеющих четный индекс.

Решение задачи:

Запишем все числа в список. С помощью функции `max` и среза, оставляющего только элемента с чётным индексом (нумерация элементов массива с 0) выведем максимальный из элементов, имеющих четный индекс.

```

.....
a = list(map(int, input('Введите числа: ').split()));

print(max(a[::2]));
.....

```

Полный код программы приведен в приложении **Д**.

## 4 Библиотека Random

Модуль random предоставляет функции для генерации случайных чисел, букв, случайного выбора элементов последовательности. Описание функций этого модуля на русском языке можно посмотреть вот [здесь](#). Рассмотрим некоторые из них.

- `random.randrange(start, stop, step)` - возвращает случайно выбранное число из последовательности
- `random.randint(A, B)` - случайное целое число  $N$ ,  $A \leq N \leq B$
- `random.choice(sequence)` - случайный элемент непустой последовательности.
- `random.shuffle(sequence, [rand])` - перемешивает последовательность, изменяя ее, поэтому эта функция не работает для строк и кортежей
- `random.sample(population, k)` - список длиной  $k$  из последовательности `population`
- `random.random()` - случайное число от 0 до 1
- `random.uniform(A, B)` - случайное число с плавающей точкой,  $A < N < B$  (или  $B < N < A$ ).
- `random.triangular(low, high, mode)` - случайное число с плавающей точкой,  $low < N < high$ . Mode - распределение.

### Задача 6

`id` - случайное пятизначное число логин - случайная последовательность из 6 маленьких английских букв пароль - случайная последовательность 10 неповторяющихся больших и маленьких английских букв и цифр Создайте функцию генерации `id`, функцию генерации логина и функцию генерации пароля. С использованием этих трёх функций напишите функцию генерации списка из  $N$  троек вида (`id`, логин, пароль), `id`, логины и пароли в тройках не должны повторяться. При этом (гласными считаем: `aeiou`): Предпоследняя цифра `id` равна 2. В логине не больше 2 гласных букв. Пароль не должен заканчиваться цифрой, но хотя бы одна цифра в нём должна присутствовать.

Создадим функции, генерирующие `id`, логин и пароль.

`id` - пятизначное число, оканчивающееся цифрой 2. Для этого с помощью функции `random.randint(10000, 99999)` сгенерируем число в диапазоне от 10000 до 99999, т.е любое пятизначное. Будем генерировать это число до тех пор, пока оно не будет оканчиваться на цифру 2.

```

.....
def idGen():
    id = 0;
    while (id % 10 != 2):
        id = random.randint(10000, 99999)

    return id;
.....

```

Логин - случайная последовательность из 6 маленьких английских букв, содержащая не более 2х гласных букв. Для этого генерируем 6 случайных букв. При каждой итерации проверяем количество гласных и если оно превышает максимальное, то генерируем букву до тех пока, пока она не будет согласной.

```

.....
def loginGen():
    login = '';
    alphabet = ['q', 'w', 'e', 'r', 't', 'y', 'u', 'i', 'o', 'p', 'a', 's', 'd', 'f',
    vowels = ['a', 'e', 'i', 'o', 'u'];
    volwesCount = 0;

    for i in range (6):
        letter = random.choice(alphabet);

        if (not(letter in vowels) or volwesCount < 2):
            login+=letter;
            if (letter in vowels):
                volwesCount+=1;
        else:
            while (letter in vowels):
                letter = random.choice(alphabet);
            login+=letter;

    return login;
.....

```

Пароль - случайная последовательность 10 неповторяющихся больших и маленьких английских букв и цифр. Пароль не должен заканчиваться цифрой, но хотя бы одна цифра в нём должна присутствовать. Для этого генерируем случайную цифру от 0 до 9 и ее место в пароле - от 0 до 8, т.к она не может

быть последней согласно условию. В цикле генерируем 9 случайных букв, генерируем, будет она заглавной или строчной. Проверяем, является ли данное место местом цифры. Если является, то вставляем заранее сгенерированную цифру, иначе - букву.

```
.....
def passGen():
    password = '';
    alphabet = ['q', 'w', 'e', 'r', 't', 'y', 'u', 'i', 'o', 'p', 'a', 's', 'd', 'f', 'g', 'h', 'j', 'k', 'l', 'z', 'x', 'c', 'v', 'b', 'n', 'm'];
    characters = ['q', 'w', 'e', 'r', 't', 'y', 'u', 'i', 'o', 'p', 'a', 's', 'd', 'f', 'g', 'h', 'j', 'k', 'l', 'z', 'x', 'c', 'v', 'b', 'n', 'm'];
    digits = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9'];
    digit = random.choice(digits);
    digitPlace = random.randint(0,8);

    for i in range (10):

        if (i != digitPlace):
            letter = random.choice(alphabet);
            ifLower = random.randint(0,1);
            if (ifLower == 0):
                password += letter.upper();
            else:
                password += letter;

        else:
            password+=digit;

    return password;
.....
```

Создадим функцию generate, возвращающую список из необходимого количества уникальных связей id-логин-пароль.

```
.....
def generate(n):
    a = list();

    for i in range(n):
        a1 = list();
        a1.append(idGen());
        a1.append(loginGen());
```

```
        a1.append(passGen());

        if (isUnique(a, a1)):
            a.append(a1);

    return a;
.....
```

Полный код программы приведен в приложении **E**.



## 5 Библиотека NumPy

NumPy — это библиотека языка Python, добавляющая поддержку больших многомерных массивов и матриц, вместе с большой библиотекой высокоуровневых (и очень быстрых) математических функций. Python библиотека NumPy незаменима для работы с числовыми массивами, векторами и матрицами, а также позволяет строить графики.

**ЗАДАЧА 6** Написать программу, реализующую алгоритм из индивидуального задания с использованием list, замерить время выполнения и сравнить с готовой реализацией алгоритма из библиотеки NumPy 10x10, 100x100, 500x500  
24. Создать единичную матрицу.

Решение задачи:

Используем библиотеку time для замеров времени работы матриц и экземпляров библиотеки NumPy.

Произведем замеры для матрицы 10x10. В библиотеке NumPy существует специальная функция для создания единичной матрицы - np.eye(n), где n - размерность матрицы.

```
.....
time1 = time.time();
a1 = np.eye(n);
time2 = time.time();
print ((time2 - time1)*1000, 'ms by NumPY');
.....
```

Для матрицы типа list используем вложенный цикл. Если координаты равны, то в матрице ставится 1, иначе - 0.

```
.....
time1 = time.time();

a = list();
for i in range(n):
    a2 = list();
    for j in range(n):
        if (i == j):
            a2.append(1);
        else:
            a2.append(0);
```

```
a.append(a2);  
  
time2 = time.time();  
print ((time2 - time1)*1000, 'ms by list');  
.....
```

Аналогично для матриц размерности 100x100 и 500x500.  
Полный код программы приведен в приложении Ж.

## 6 Библиотека MATPLOTLIB

Библиотека `matplotlib` - это библиотека двумерной графики для языка программирования Python, с помощью которой можно создавать высококачественные рисунки различных форматов. `Matplotlib` представляет собой модуль-пакет для Python.

Главной единицей при работе с `matplotlib` является рисунок (`Figure`). Любой рисунок в `matplotlib` имеет вложенную структуру.

Рисунок является объектом самого верхнего уровня, на котором располагаются одна или несколько областей рисования (`Axes`), элементы рисунка `Artists` (заголовки, легенда и т.д.) и основа-холст (`Canvas`). На рисунке может быть несколько областей рисования `Axes`, но данная область рисования `Axes` может принадлежать только одному рисунку `Figure`.

Область рисования является объектом среднего уровня, который является главным объектом работы с графикой `matplotlib` в объектно-ориентированном стиле. Это то, что ассоциируется со словом "plot" это часть изображения с пространством данных. Каждая область рисования `Axes` содержит две (или три в случае трёхмерных данных) координатных оси (`Axis` объектов), которые упорядочивают отображение данных.

Координатная ось является объектом среднего уровня, которые определяют область изменения данных, на них наносятся деления `ticks` и подписи к делениям `ticklabels`. Расположение делений определяется объектом `Locator`, а подписи делений обрабатывает объект `Formatter`. Конфигурация координатных осей заключается в комбинировании различных свойств объектов `Locator` и `Formatter`.

Элементы рисунка `Artists` являются как бы красной линией для всех иерархических уровней. Практически всё, что отображается на рисунке является элементом рисунка (`Artist`), даже объекты `Figure`, `Axes` и `Axis`. Элементы рисунка `Artists` включают в себя такие простые объекты как текст (`Text`), плоская линия (`Line2D`), фигура (`Patch`) и другие. Когда происходит отображение рисунка (`figure rendering`), все элементы рисунка `Artists` наносятся на основу-холст (`Canvas`). Большая часть из них связывается с областью рисования `Axes`. Также элемент рисунка не может совместно использоваться несколькими областями `Axes` или быть перемещён с одной на другую.

**ЗАДАЧА 7** Постройте несколько функций на одном графике различными

цветами. Для построенного графика сделайте сетку и легенду.

Решение задачи:

Построим 3 графика различных функций. Зададим диапазон значений  $x$  с помощью функции `np.linspace`. Задаем выражение функции через переменную  $y$ . Построим график этой функции, задав цвет линии и подпись для легенды

```
.....  
x = np.linspace(-2 * np.pi, 2 * np.pi)  
  
y = 1/2*np.cos(2*x+1)  
  
plt.plot(x, y, color='#000000', label = '0.5cos(2x+1)')  
.....
```

Аналогично для двух других графиков функций. Для ввода на экран графика, сетки и легенды выполним следующие команды:

```
.....  
plt.legend()  
plt.grid()  
plt.show()  
.....
```

Полный код программы приведен в приложении **3**.

## 7 Библиотека Pandas

Pandas это высокоуровневая Python библиотека для анализа данных. Высокоуровневая, потому что построена она поверх более низкоуровневой библиотеки NumPy, что является большим плюсом в производительности. Pandas является наиболее продвинутой и быстроразвивающейся библиотекой для обработки и анализа данных.

**ЗАДАЧА 8** Создать 2 файла .csv с товарами: номер склада, наименование товара, количество, вес, хрупкость(да/нет), требуется хранить в холодильнике(да/нет), страна производитель. В первом файле товары одного склада, во втором другого. Объединить данные из этих файлов в один DataFrame. 24.Найти процент товаров количеством < 10 среди товаров из Германии.

Решение задачи:

Создадим 2 файла и объединим их в один.

```
.....  
df1 = pd.read_csv('input1.csv', delimiter=',')  
df2 = pd.read_csv('input2.csv', delimiter=',')  
  
df1 = df1.append(df2)  
.....
```

Подсчитаем и выведем необходимое число

```
.....  
print(df1[(df1['number'] < 10) & (df1['country'] == 'Germany')].shape[0] / df1[df1['country'] == 'Germany'].shape[0])  
.....
```

Полный код программы приведен в приложении **И**.

## 8 Финальная задача

Найти готовый файл с данными.

Для выполнения задания был взят файл с данными о футбольных матчах между национальными сборными с 2018 года по настоящее время

```
.....
sys.stdout = open('output.txt', 'w')
df = pd.read_csv('results.csv', delimiter=',')
.....
```

Посчитать статистические показатели: - медиана, - мода, - средняя, - минимум,

Сделаем это с помощью стандартных функций библиотеки по полям `homescore` и `awayscore`

```
.....
#2
print('Median score:')
print(df[['home_score', 'away_score']].median())
print()

print('Mode score:')
print(df[['home_score', 'away_score']].mode())
print()

print('Average score:')
print(df[['home_score', 'away_score']].mean())
print()

print('Min score:')
print(df[['home_score', 'away_score']].min())
print()

print('Max score:')
print(df[['home_score', 'away_score']].max())
print()
.....
```

Разбить данные на несколько блоков (не менее 3, приблизительно равных по размеру) по некоторому критерию (критерий определить самостоятельно).

Разобьем данные на 3 части: домашние и гостевые матчи сборных России, Англии и Германии.

```
.....
#3
df_rus = df.query("home_team == 'Russia' or away_team == 'Russia'")
df_rus.to_csv('Russia.csv')
df_eng = df.query("home_team == 'England' or away_team == 'England'")
df_eng.to_csv('England.csv')
df_ger = df.query("home_team == 'Germany' or away_team == 'Germany'")
df_ger.to_csv('Germany.csv')
.....
```

Привести круговую диаграмму для проделанного разбиения.

```
.....
#4
data = [df_rus.shape[0], df_eng.shape[0], df_ger.shape[0]]
labels = ["Matches with Russia", "Matches with England", "Matches with Germany"]

#5
pylab.pie(data, labels=labels)
plt.savefig('pic1.png')
pylab.show()
.....
```

В зависимости от особенностей набора данных для каждого блока построить график (гистограмму, ...).

Для матчей сборной России построим гистограмму по городам, в которых были сыграны матчи. Цифры будут означать количество матчей в определенном городе.

```
.....
#6
plt.hist(df_rus["city"])
plt.savefig('pic2.png')
plt.show()
.....
```

Для матчей сборной Англии подсчитаем количество раз, в который команда забила на выезде то или иное количество мячей.

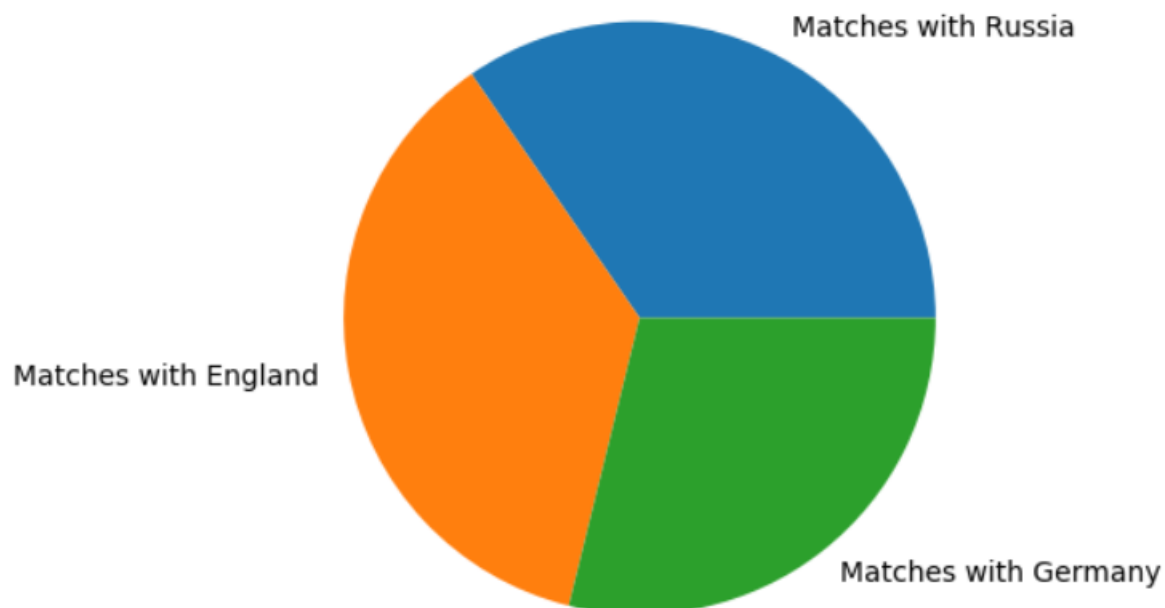


Рисунок 8.1 – Круговая диаграмма разбиения - pic1.png.

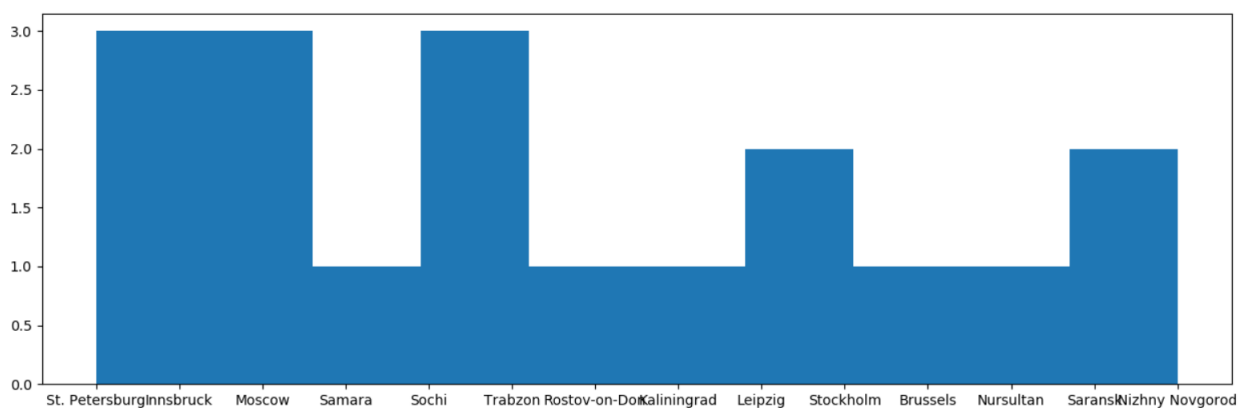


Рисунок 8.2 – Разбиение городов, в которых проводились матчи сборной России - pic2.png.

```
.....
plt.hist(df_eng['away_score'])
plt.savefig('pic3.png')
plt.show()
.....
```

Для матчей сборной Германии определим, в каких городах команда играла в том или ином турнире



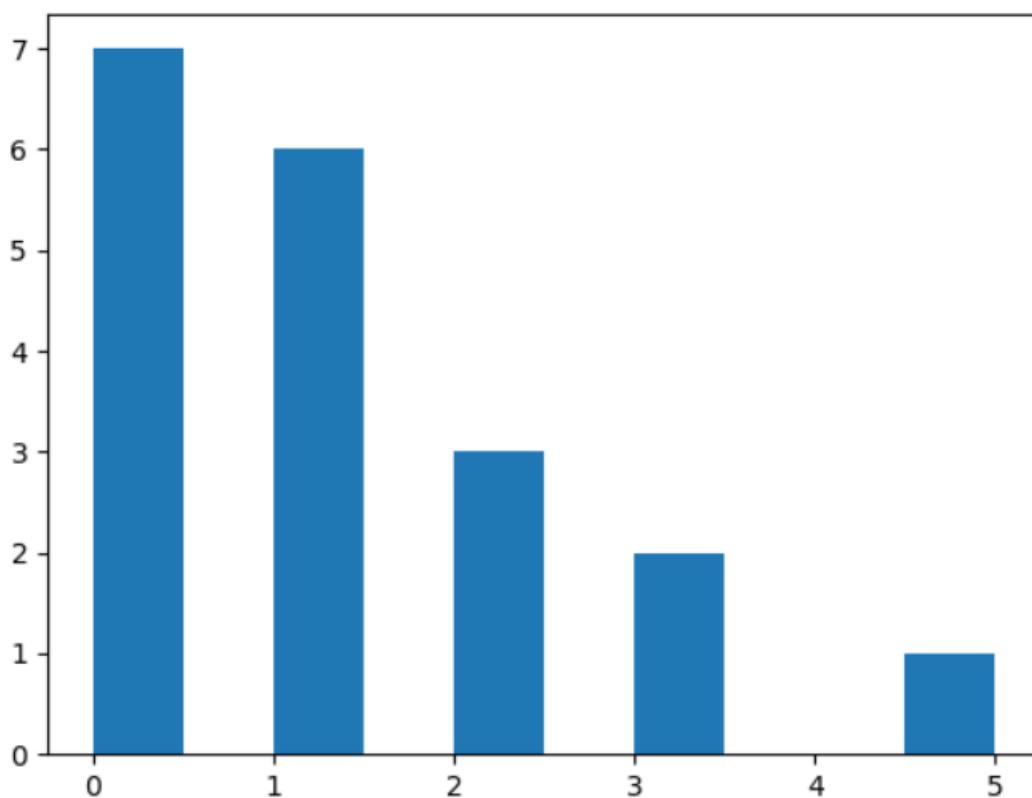


Рисунок 8.3 – Количество мячей и количество раз, сколько из забила сборная Англии на выезде - pic3.png.

```
.....
plt.plot(df_ger['city'], df_ger['tournament'])
plt.savefig('pic4.png')
plt.show()
.....
```

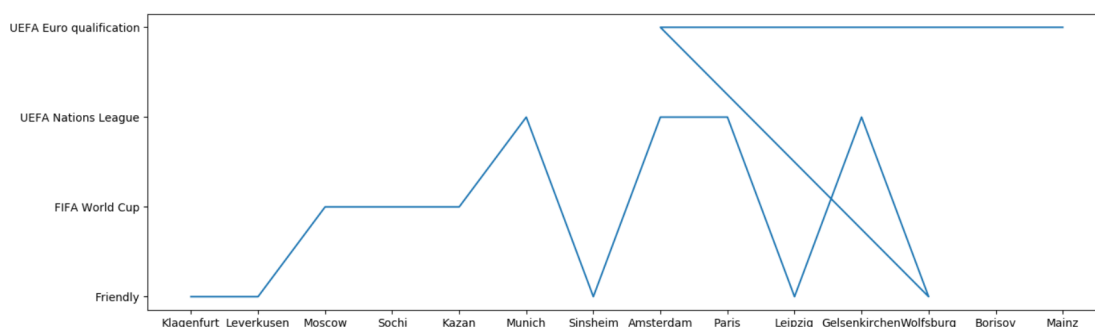


Рисунок 8.4 – Города, в которых сборная Германия играла в разных турнирах - pic4.png.

Сгруппировать данные (GroupBy) по некоторому признаку и сохранить результаты в новые таблицы. Для каждой новой таблицы провести сортировку по сложному ключу, состоящему из нескольких признаков. Для каждой

таблицы своя сортировка.

Сгруппируем изначальные данные по городу, в котором проводился матч и забитым/пропущенным мячам командами, игравшими дома/на выезде.

```
.....  
dfgroup = df.groupby('city')[['home_score', 'away_score']].sum()  
dfgroup.to_csv('city_groupby.csv')  
.....
```

Полный код программы приведен в приложении **К**.

## **ЗАКЛЮЧЕНИЕ**

В ходе практики был изучен синтаксис, базовые конструкции и структуры языка Python. В частности, были изучены методы и возможности популярных библиотек Numpy, Pandas, Matplotlib. Применение этих библиотек существенно упростило и ускорило процесс решения задач, из чего следует необходимость использования расширения языка Python.

## ПРИЛОЖЕНИЕ А

### Вектор

**ЗАДАЧА 1** Дана точка на плоскости с координатами  $(x, y)$ . Составить программу, которая выдает одно из сообщений «Да», «Нет», «На границе» в зависимости от того, лежит ли точка внутри заштрихованной области, вне заштрихованной области или на ее границе. Области задаются графически следующим образом:

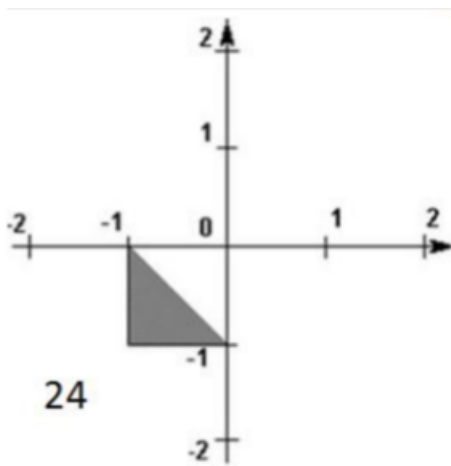


Рисунок А.1 – Рисунок 1

Решение задачи:

```
.....
x = float(input('x = '));
y = float(input('y = '));

#y=-x-1 and x<=0 and y<=0
#x=-1 and y>=0 and y<=-1
#y=-1 and x>=0 and x<=-1

if (y<-x-1) and (x>-1) and (y>-1):
    print("Да");
elif not((y<=-x-1) and (x>=-1) and (y>=-1)):
    print("Нет");
else:
    print('На границе');
.....
```

Пример:

Вход	Выход
1 2	Нет
-0.5 -0.5	На границе
-0.75 -0.75	Да

## ПРИЛОЖЕНИЕ Б

### Циклы

**ЗАДАЧА 2** Даны два отрезка А и В ( $A > B$ ). Не используя операции умножения и деления, определить, сколько отрезков В уместится в отрезке А.

Решение задачи:

```
.....  
a = float(input('a = '));  
b = float(input('b = '));  
  
s = 0.0;  
  
while (a>s*b):  
    s+=1;  
  
if (s*b > a):  
    s-=1;  
  
print(s);  
  
.....
```

Пример:

Вход	Выход
2 1	1
15 5	3
30 20	1

## ПРИЛОЖЕНИЕ В

### Функции

**ЗАДАЧА 3** Разработать функцию, которая по заданному  $n$  возвращает список  $n$  первых членов заданной последовательности:  $b_1 = 2.3$ ,  $b_2 = -5$ ,  $b_n = b_{n-2} + 2b_{n-1}$ .

Решение задачи:

```
.....  
a = float(input('a = '));  
b = float(input('b = '));  
  
s = 0.0;  
  
while (a>s*b):  
    s+=1;  
  
if (s*b > a):  
    s-=1;  
  
print(s);  
  
.....
```

Пример:

Вход	Выход
4	[2.3, -5, -7.7, -20.4]
5	[2.3, -5, -7.7, -20.4, -48.5]
2	[2.3, -5]

## ПРИЛОЖЕНИЕ Г

### Строки

**ЗАДАЧА 4** Удалить из строки слова, содержащие повторяющиеся символы.

Решение задачи:

```
.....
str = input('String: ');
i=0;
str_ans='';
words = ''.join(x for x in str if x.isalpha() or x == ' ').split();

for sub_str in words[:]:

    sub_set = set(sub_str.lower());

    if not(len(sub_set) == len(sub_str)):
        words.remove(sub_str);

words1 = ' '.join(words);
print(words1);

.....
```

Пример:

Вход	Выход
absAcaDa abc	abc
amdkvndkn. dflkbvm, ekfsnv!	dflkbvm ekfsnv
a	a



## ПРИЛОЖЕНИЕ Д

### Списки

**ЗАДАЧА 5** Дан одномерный массив чисел. Найти/определить максимальный из элементов, имеющих четный индекс.

Решение задачи:

```
.....  
a = list(map(int, input('Введите числа: ').split()));  
  
print(max(a[::2]));  
.....
```

Пример:

Вход	Выход
1 2 3 4	3
-1 2 -3 4	-1
0	0

## ПРИЛОЖЕНИЕ Е

### Библиотека Random

ЗАДАЧА 5 id - случайное пятизначное число логин - случайная последовательность из 6 маленьких английских букв пароль - случайная последовательность 10 неповторяющихся больших и маленьких английских букв и цифр Создайте функцию генерации id, функцию генерации логина и функцию генерации пароля. С использованием этих трёх функций напишите функцию генерации списка из N троек вида (id, логин, пароль), id, логины и пароли в тройках не должны повторяться. При этом (гласными считаем: aeіou): Предпоследняя цифра id равна 2. В логине не больше 2 гласных букв. Пароль не должен заканчиваться цифрой, но хотя бы одна цифра в нём должна присутствовать.

Решение задачи:

```
.....
import random
import string

def idGen():
    id = 0;
    while (id % 10 != 2):
        id = random.randint(10000, 99999)

    return id;

def loginGen():
    login = '';
    alphabet = ['q', 'w', 'e', 'r', 't', 'y', 'u', 'i', 'o', 'p', 'a', 's', 'd', 'f',
    vowels = ['a', 'e', 'i', 'o', 'u'];
    volwesCount = 0;

    for i in range (6):
        letter = random.choice(alphabet);

        if (not(letter in vowels) or volwesCount < 2):
            login+=letter;
            if (letter in vowels):
                volwesCount+=1;
        else:
```

```

        while (letter in vowels):
            letter = random.choice(alphabet);
        login+=letter;

    return login;

def passGen():
    password = '';
    alphabet = ['q', 'w', 'e', 'r', 't', 'y', 'u', 'i', 'o', 'p', 'a', 's', 'd', 'f', 'g', 'h', 'j', 'k', 'l', 'z', 'x', 'c', 'v', 'b', 'n', 'm'];
    characters = ['q', 'w', 'e', 'r', 't', 'y', 'u', 'i', 'o', 'p', 'a', 's', 'd', 'f', 'g', 'h', 'j', 'k', 'l', 'z', 'x', 'c', 'v', 'b', 'n', 'm'];
    digits = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9'];
    digit = random.choice(digits);
    digitPlace = random.randint(0,8);

    for i in range (10):

        if (i != digitPlace):
            letter = random.choice(alphabet);
            ifLower = random.randint(0,1);
            if (ifLower == 0):
                password += letter.upper();
            else:
                password += letter;

        else:
            password+=digit;

    return password;

def isUnique(a, a1):
    for i in range(len(a)):
        if (a[i] == a1[i]):
            return False;
    return True;

def generate(n):
    a = list();

    for i in range(n):
        a1 = list();
        a1.append(idGen());

```

```

        a1.append(loginGen());
        a1.append(passGen());

        if (isUnique(a, a1)):
            a.append(a1);

    return a;

n = int(input('n = '));
a = generate(n);

for i in range (len(a)):
    print(a[i]);

.....

```

Пример:

Вход	Выход
5	[45082, 'vxcoen', 'JxG1FinVeX'] [43492, 'fjekes', '9DpjqMkEcd'] [35852, 'jadudn', 'ypnHrKPV4p'] [69632, 'dlccmx', 'X9JjyUfMYf'] [46802, 'zkmlhz', '0X04jSVqUS']
0	
-1	

## ПРИЛОЖЕНИЕ Ж

### Библиотека NumPy

ЗАДАЧА 6 Написать программу, реализующую алгоритм из индивидуального задания с использованием list, замерить время выполнения и сравнить с готовой реализацией алгоритма из библиотеки NumPy 10x10, 100x100, 500x500

24. Создать единичную матрицу.

Решение задачи:

```
.....
import numpy as np
import random
import time

#10x10
n = 10;
print('for', n, 'elements:');

time1 = time.time();

a = list();
for i in range(n):
    a2 = list();
    for j in range(n):
        if (i == j):
            a2.append(1);
        else:
            a2.append(0);
    a.append(a2);

time2 = time.time();
print ((time2 - time1)*1000, 'ms by list');

time1 = time.time();
a1 = np.eye(n);
time2 = time.time();
print ((time2 - time1)*1000, 'ms by NumPY');
print('');

#100x100
n = 100;
print('for',n,'elements:');
```

```

time1 = time.time();

a = list();
for i in range(n):
    a2 = list();
    for j in range(n):
        if (i == j):
            a2.append(1);
        else:
            a2.append(0);
    a.append(a2);

time2 = time.time();
print ((time2 - time1)*1000, 'ms by list');

time1 = time.time();
a1 = np.eye(n);
time2 = time.time();
print ((time2 - time1)*1000, 'ms by NumPY');
print('');

#500x500
n = 500;
print('for',n,'elements:');

time1 = time.time();

a = list();
for i in range(n):
    a2 = list();
    for j in range(n):
        if (i == j):
            a2.append(1);
        else:
            a2.append(0);
    a.append(a2);

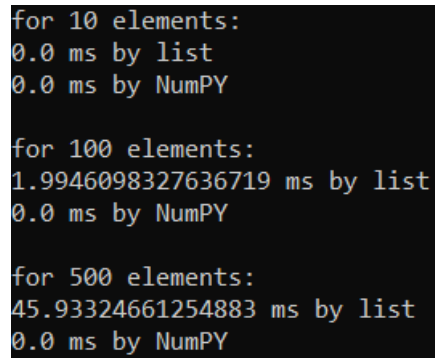
time2 = time.time();
print ((time2 - time1)*1000, 'ms by list');

```

```
time1 = time.time();
a1 = np.eye(n);
time2 = time.time();
print ((time2 - time1)*1000, 'ms by NumPY');

.....
```

Пример:



```
for 10 elements:
0.0 ms by list
0.0 ms by NumPY

for 100 elements:
1.9946098327636719 ms by list
0.0 ms by NumPY

for 500 elements:
45.93324661254883 ms by list
0.0 ms by NumPY
```

Рисунок Ж.1 – Сравнение времени работы матриц и экземпляра библиотеки NumPy

## ПРИЛОЖЕНИЕ 3

### Библиотека Mathplotlib

**ЗАДАЧА 7** Постройте несколько функций на одном графике различными цветами. Для построенного графика сделайте сетку и легенду.

Решение задачи:

```
.....
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import math

x = np.linspace(-2 * np.pi, 2 * np.pi)

y = 1/2*np.cos(2*x+1)
f = np.cos(2*x+1)*x;
g = -1/5*np.sin(3*x-1)

plt.plot(x, y, color='#000000', label = '0.5cos(2x+1)')

plt.plot(x, f, color='#FF4500', label = 'xcos(2x+1)')

plt.plot(x, g, color='#20B2AA',label = '-0.2sin(3x-1)')

plt.legend()
plt.grid()
plt.show()
.....
```

Пример:



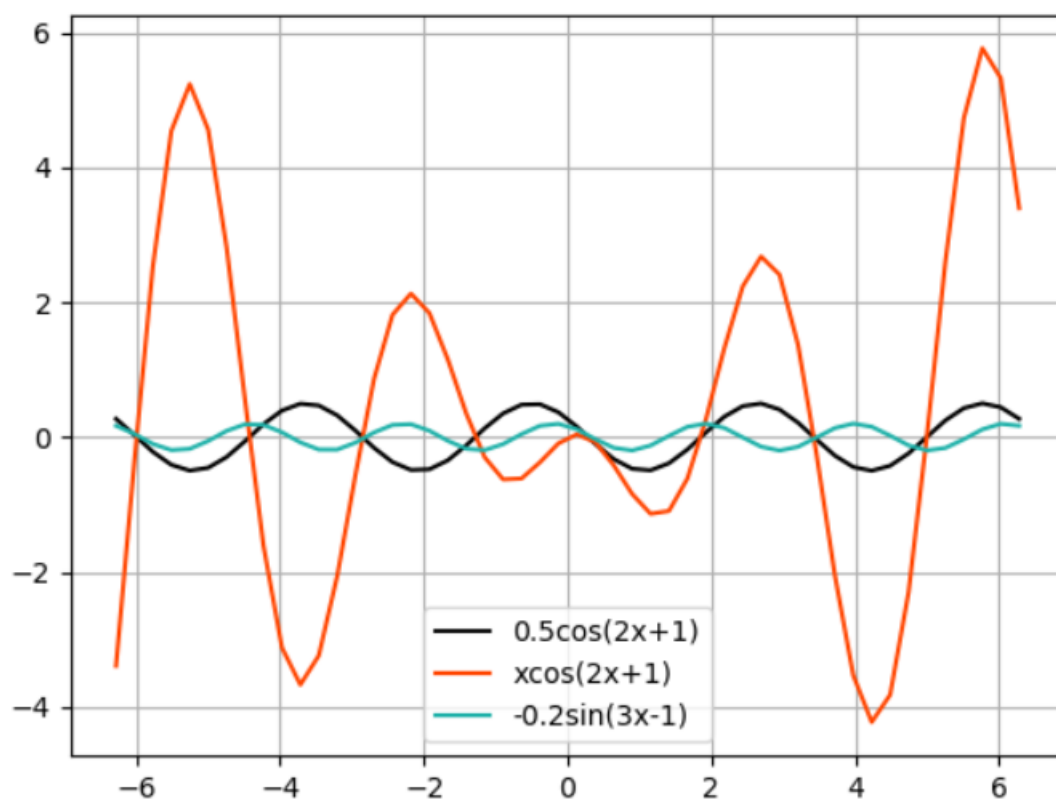


Рисунок 3.1 – Графики функций, легенда и сетка

## ПРИЛОЖЕНИЕ И

### Библиотека Pandas

**ЗАДАЧА 8** Создать 2 файла .csv с товарами: номер склада, наименование товара, количество, вес, хрупкость(да/нет), требуется хранить в холодильнике(да/нет), страна производитель. В первом файле товары одного склада, во втором другого. Объединить данные из этих файлов в один DataFrame. 24.Найти процент товаров количеством < 10 среди товаров из Германии.

Решение задачи

```
.....
import pandas as pd
import sys

sys.stdout = open('output.txt', 'w')

df1 = pd.read_csv('input1.csv', delimiter=',')
df2 = pd.read_csv('input2.csv', delimiter=',')

df1 = df1.append(df2)

print(df1[(df1['number'] < 10) & (df1['country'] == 'Germany')].shape[0] / df1[df1['country'] == 'Germany'].shape[0])
.....
```

Пример:

input1.csv:

```
.....
warehouse number,product,number,weight,fragility,in refrigerator,country
1,glass,12,21,Yes,No,Germany
1,phone,144,12,No,Yes,China
1,plastic,2,99,No,No,Japan
1,beer,5,4,Yes,No,Germany
1,tank,15,294,No,No,Russia
1,coffee,22,512,Yes,No,Indonesia
.....
```

input2.csv:

```
.....
warehouse number,product,number,weight,fragility,in refrigerator,country
2,fur,52,13,Yes,No,Russia
```

```
2,meat,1,56,No,No,Germany
2,bread,2,41,No,No,Russia
2,bread,45,12,Yes,No,USA
2,oil,11,22,No,Yes,Italy
2,juice,12,10,No,Yes,Germany
.....
```

**output.txt:**

```
.....
0.5
.....
```

## ПРИЛОЖЕНИЕ К

### Финальная задача

```
.....
import pandas as pd
import sys
import matplotlib.pyplot as plt
import pylab

#1
sys.stdout = open('output.txt', 'w')
df = pd.read_csv('results.csv', delimiter=',')

#2
print('Median score:')
print(df[['home_score', 'away_score']].median())
print()

print('Mode score:')
print(df[['home_score', 'away_score']].mode())
print()

print('Average score:')
print(df[['home_score', 'away_score']].mean())
print()

print('Min score:')
print(df[['home_score', 'away_score']].min())
print()

print('Max score:')
print(df[['home_score', 'away_score']].max())
print()

#3
df_rus = df.query("home_team == 'Russia' or away_team == 'Russia'")
df_rus.to_csv('Russia.csv')
df_eng = df.query("home_team == 'England' or away_team == 'England'")
df_eng.to_csv('England.csv')
df_ger = df.query("home_team == 'Germany' or away_team == 'Germany'")
df_ger.to_csv('Germany.csv')
```

```

#4
data = [df_rus.shape[0], df_eng.shape[0], df_ger.shape[0]]
labels = ["Matches with Russia", "Matches with England", "Matches with Germany"]

#5
pylab.pie(data, labels=labels)
plt.savefig('pic1.png')
pylab.show()

#6
plt.hist(df_rus["city"])
plt.savefig('pic2.png')
plt.show()

plt.hist(df_eng['away_score'])
plt.savefig('pic3.png')
plt.show()

plt.plot(df_ger['city'], df_ger['tournament'])
plt.savefig('pic4.png')
plt.show()

dfgroup = df.groupby('city')[['home_score', 'away_score']].sum()
dfgroup.to_csv('city_groupby.csv')
.....

```