**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Текстовое поле

**Домашнее задание № 1**

**по дисциплине:**

**«Разработка веб-приложений»**

|  |  |
| --- | --- |
| Группа | 231-3211 |
| Студент  Преподаватель | Косарева Светлана Александровна  Речинский Владислав Алексеевич |

Москва 2025 г.

Чтобы запустить тесты, следуем следующим шагам:

1. Создайте виртуальное окружение:

python -m venv ve

2. Активируйте виртуальное окружение:

- Для Windows используйте команду: ve\Scripts\activate

Если файлы находятся в другой папке, укажите путь к ней. Например:

cd C:\WEB

3. Установите pytest:

pip install pytest

4. Запустите тесты:

pytest test.py

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

1) Скажи "Hello, World!" с помощью Python

Выведите "Hello, World!" на стандартный поток вывода.

Название файла: hello.py

Изображение выглядит как снимок экрана, текст, Мультимедийное программное обеспечение, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

2) Python If-Else

Получив целое число (n), выполните следующие условные действия:

\* Если n нечетное значение, выведите "Weird";

\* Если n четно и находится в диапазоне от 2 до 5 включительно, выведите "Not Weird";

\* Если n четно и находится в диапазоне от 6 до 20 включительно, выведите "Weird";

\* Если n четно и больше 20, выведите "Not Weird".

Формат ввода:

Строка, содержащая положительное целое число (n).

Ограничения:

1⩽n⩽100

Название файла: python\_if\_else.py

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

**3) Арифметические операторы**

Считайте два целых числа из стандартного потока ввода (a и b), и напишите код для вывода трех строк, где:

1. Первая строка содержит сумму двух чисел.

2. Вторая строка содержит разницу двух чисел (первое - второе).

3. Третья строка содержит произведение двух чисел.

**Формат ввода:**

Первая строка содержит первое целое число (a), вторая строка -- второе целое число (b).

**Ограничения:**

1⩽a⩽10101⩽a⩽1010

1⩽b⩽10101⩽b⩽1010

**Название файла:** arithmetic\_operators.py

Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

**4) Деление**

Считайте два целых числа a и b из стандартного потока ввода.

Реализуйте логику для печати двух строк. Первая строка должна содержать результат целочисленного деления,

(a // b). Вторая строка должна содержать результат деления с плавающей точкой  (a/b).

Округление или форматирование не требуется. Ваша программа должна корректно отбрабатывать деление на ноль.

**Формат ввода:**

Первая строка содержит первое целое число (a), вторая строка -- второе целое число (b).

**Название файла:** division.py

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

**5)  Циклы**

Считайте целое число n. Для всех неотрицательных целых чисел  i<ni<n выведите i2i2.

**Формат ввода:**

Строка, содержащая целое число (n).

**Ограничения:**

1⩽n⩽201⩽n⩽20

**Название файла:** loops.py

Изображение выглядит как текст, Мультимедийное программное обеспечение, программное обеспечение, Графическое программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

**6) Функция вывода**

Считайте целое число n из стандартного потока ввода.

**Формат ввода:**

Строка, содержащая целое число (n).

**Ограничения:**

1⩽n⩽201⩽n⩽20

**Название файла:** print\_function.py

Изображение выглядит как текст, снимок экрана, Мультимедийное программное обеспечение, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

**7) Узнай, кто занял второе место!**

Учитывая таблицу результатов участников вашей университетской спартакиады, вам необходимо указать результат, занявший второе место. Вам даны быллы n участников. Запишите их в список и найдите оценку занявшего второе место.

**Формат ввода:**

Первая строка содержит n. Вторая строка содержит массив A[] из n целых чисел, разделённых пробелом.

**Формат вывода:**

Выведите результат, занявший второе место.

**Название файла:** second\_score.py

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

**8) Вложенные списки**

Учитывая имена и оценки каждого учащегося в классе, сохраните их во вложенном списке и выведите имя (имена) любого учащегося, имеющего вторую по величине оценку.

Примечание: Если есть несколько учеников со вторыми по величине оценками, расположите их имена в алфавитном порядке и выведите каждое имя с новой строки.

Например:

records = [['chi', 20.0], ['beta', 50.0], ['alpha', 50.0]]

Упорядоченный список оценок - [20.0,50.0] , поэтому второй по величине балл - 50.0. Есть два студента с таким баллом: ['beta','alpha']. Имена в алфавитном порядке напечатаны как:

alpha

beta

**Формат ввода:**

Первая строка содержит целое число (N) -- количество учащихся.

В 2N последующих строках каждый учащийся описывается в 2 строках:

- В первой строке указано имя учащегося.

- Во второй строке указана его оценка.

**Формат вывода:**

Выведите имя (имена) любого ученика (ов), имеющего вторую по величине оценку. Если учеников несколько, расположите их имена в алфавитном порядке и выведите каждое с новой строки.

**Ограничения:**

2⩽N⩽52⩽N⩽5

Во вводимых данных всегда должен присутствовать хотя бы один учащийся со второй по величине оценкой.

**Название файла:** nested\_list.py

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

**9) Cписки**

Рассмотрим список (arr = []). Вы можете выполнить следующие команды:

1. insert i e: вставить целое число e в позицию i.

2. print: вывести список.

3. remove e: удалить первое вхождение целого числа e.

4. append e: вставить целое число e в конце списка.

5. sort: сортировать список.

6. pop: удалить последний элемент из списка.

7. reverse: перевернуть список.

Инициализируйте свой список и считайте из стандартного потока ввода значение n, за которым следуют n строк команд, где каждая команда будет соответствовать одному из приведённых выше 7 типов. Пройдитесь по командам в порядке их ввода и выполните соответствующую операцию в вашем списке.

**Формат ввода:**

Первая строка содержит целое число n,  обозначающее количество команд.

Каждая строка i из n последующих строк содержит одну из команд, описанных выше.

**Ограничения:**

\* Элементы, добавляемые в список, должны быть целыми числами.

**Формат вывода:**

Для каждой команды типа print выведите список с новой строки.

**Название файла:** lists.py

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

**10) Пример обмена**

Вам дается строка, и ваша задача - поменять регистры местами. Другими словами, преобразуйте все строчные буквы в прописные и наоборот.

**Формат ввода:**

Строка (s), содержащая некоторую последовательность символов.

**Ограничения:**

0<len(s)⩽10000<len(s)⩽1000

**Название файла:** swap\_case.py

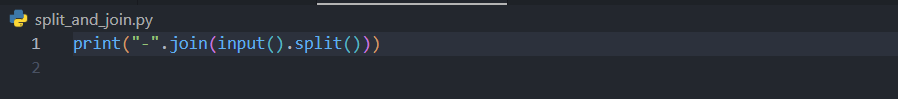
Изображение выглядит как текст, Мультимедийное программное обеспечение, программное обеспечение, Графическое программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

**11) Разделение и объединение строк**

Вам через стандартный поток ввода будет предоставлена строка. Разделите строку по пробелу и соедините получившиеся фрагменты с помощью дефиса.

**Название файла:** split\_and\_join.py



**12) Поиск слова максимальной длины**

Напишите программу, которая считывает содержимое файла example.txt (приложен к заданию) и выводит на стандартный поток вывода слово максимальной длины, встречающееся в этом файле. Если таких слов несколько, программа должна вывести все. При подсчёте длины слова не должны учитываться знаки препинания, кавычки и прочие спецсимволы.

**Формат вывода:**

Строка, содержащая слово максимальной длины (если таких слов несколько -- каждое на отдельной строке в порядке их встречаемости в тексте).

**Название файла:** max\_word.py

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Контент, сгенерированный ИИ, может содержать ошибки.

**13) Агрегирование данных**

В файле products.csv (приложен к заданию) приведены данные ежемесячных трат на продукты для различных категорий потребителей. Напишите программу, которая считывает данные из файла и вычисляет суммарные затраты для каждой категории потребителей (взрослый, пенсионер, ребёнок). Полученные значения выведите на стандартный поток вывода.

**Формат вывода:**

Три числа разделённых пробелами -- суммарные затраты для каждой категории потребителей. Каждое число должно быть округлено до двух знаков после десятичного разделителя.

**Название файла:** price\_sum.py

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

**14) Анаграммы**

Даны две строки. Определите, можно ли получить одну из другой перестановкой букв.

**Формат ввода:**

Программа получает на вход две строки, содержащие только ASCII-символы, не содержащие пробелы.

**Формат вывода:**

Программа должна вывести слово YES, если одна строка может быть получена из другой перестановкой букв или NO.

**Название файла:** anagram.py

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

**15) Пассажиры в метро**

Для изучения пассажиропотока в метро было записано время входа и время выхода в метро каждого пассажира. На основании этих данных определите, сколько пассажиров было в метро в некоторый заданный момент времени T.

**Формат ввода:**

Программа получает на вход число пассажиров N.

Далее в N строках записано время входа (AiAi) и время выхода (BiBi) каждого пассажира (Ai<BiAi<Bi). Время задаётся в виде целого числа -- количества минут от начала работы метрополитена. Значения времени входа и времени выхода разделены пробелом.

В следующей строке дано время T.

**Формат вывода:**

Выведите одно число: количество пассажиров в момент времени T. Если какой-то пассажир в момент T входит или выходит, то его тоже необходимо посчитать.

**Название файла:** metro.py

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Контент, сгенерированный ИИ, может содержать ошибки.

**16) Игра миньонов**

Кевин и Стюарт хотят сыграть в «Игру миньонов».

**Правила игры**

Обоим игрокам даётся одна и та же строка. Оба игрока должны составить подстроки, используя буквы строки. Стюарт должен составить слова, начинающиеся с согласных. Кевину нужно составить слова, начинающиеся с гласных. Игра заканчивается, когда оба игрока составили все возможные подстроки.

**Подсчет очков**

Игрок получает +1 очко за каждое вхождение подстроки в строку.

**Формат ввода:**

Одна строка ввода, содержащая строку S.

Примечание. Строка будет содержать только буквы верхнего регистра: [A-Z].

**Формат вывода:**

Одна строка, содержащая имя победителя (Кевин или Стюарт) и набранное им количество очков (значения должны быть разделены пробелом).

**Ограничения:**

0<len(S)⩽1060<len(S)⩽106

**Название файла:** minion\_game.py

Изображение выглядит как текст, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

**17) Високосный год**

Почти каждые четыре года в календарь добавляется дополнительный день — 29 февраля, и этот день называется високосным. Он вносит в календарь поправку на тот факт, что наша планета совершает оборот вокруг Солнца примерно за 365,25 дней.

**Задача**

Напишите функцию, позволяющую определить, является ли год високосным. Если это високосный год, верните логическое значение True, в противном случае верните False.

Ваша программа должна считывать год для проверки из стандартного потока ввода.

**Формат ввода:**

Строка, содержащая год для проверки (year).

**Ограничения:**

1900⩽year⩽1051900⩽year⩽105

**Выходной формат:**

Результат вызова функции (True/False).

**Название файла:** is\_leap.py

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

**18) Предпочтения**

Вам дан массив из n целых чисел. Существуют также 2 непересекающиеся множества A и B, каждое из которых содержит m целых чисел. Вам нравятся все числа в наборе A и не нравятся все числа в наборе B. Ваше первоначальное настроение равно 0. Для каждого целого числа i в массиве, если i∈Ai∈A, ваше настроение повышается на 1. Если i∈Bi∈B, ваше настроение уменьшается на 1. В противном случае ваше настроение не изменится. В конце выведите своё окончательное настроение.

Примечание. Поскольку A и B являются множествами, они не содержат повторяющихся элементов. Однако массив может содержать повторяющиеся элементы.

**Ограничения:**

1⩽n⩽1051⩽n⩽105

1⩽m⩽1051⩽m⩽105

1⩽i⩽1091⩽i⩽109

**Формат ввода:**

Первая строка содержит целые числа n и m, разделённые пробелом.

Вторая строка содержит n целых чисел — элементы массива.

Третья и четвёртая строки содержат по m целых чисел, входящих в множества A и B соответственно.

**Формат вывода:**

Выведите одно целое число — ваше общее настроение.

**Название файла:** happiness.py

def calculate\_mood(array, set\_a, set\_b):

    mood = 0

    # Пройти по каждому числу в массиве

    for number in array:

        if number in set\_a:

            mood += 1  # Увеличиваем настроение

        elif number in set\_b:

            mood -= 1  # Уменьшаем настроение

    return mood

# Чтение входных данных

n, m = map(int, input().split())

# Проверка ограничения на n и m

if not (1 <= n <= 10\*\*5) or not (1 <= m <= 10\*\*5):

    print("n и m должны быть в диапазоне от 1 до 10^5.")

# Считываем основной массив

array = list(map(int, input().split()))

# Проверка на размер массива

if len(array) != n:

    print("Размер основного массива не соответствует n.")

# Считываем множества A и B

set\_a = set(map(int, input().split()))

set\_b = set(map(int, input().split()))

# Проверка размерности множеств A и B

if len(set\_a) != m or len(set\_b) != m:

    print("Размеры множеств A и B должны быть равны m.")

# Проверка ограничений на значения элементов

for num in array + list(set\_a) + list(set\_b):

    if not (1 <= num <= 10\*\*9):

        print("Элементы должны быть в диапазоне от 1 до 10^9.")

# Вычисление и вывод конечного настроения

final\_mood = calculate\_mood(array, set\_a, set\_b)

print(final\_mood)

**19) Пиратский корабль**

Капитан пиратского корабля награбил больше, чем рассчитывал, а грузоподъёмность его судна ограничена. Помогите ему найти комбинацию наиболее ценных предметов с учётом того, что на корабль можно загружать не только целые предметы, но и их части.

**Формат ввода:**

Первая строка содержит целые числа n и m, разделённые пробелом (n -- грузоподъёмность судна, m -- количество наименований грузов). Далее идёт m срок, в каждой из которых содержится по три значения: название груза, вес (целое число), стоимость (целое число).

**Формат вывода:**

Выведите в порядке убывания стоимости список грузов, которые нужно погрузить на корабль. Каждый груз выведите в отдельной строке в формате "название вес стоимость" (вес и стоимость должны быть указаны с учётом ограничения грузоподъёмности судна, в случае дробных значений округлите до второго знака после десятичного разделителя).

**Название файла:** pirate\_ship.py

class Cargo:

    def \_\_init\_\_(self, name, weight, value):

        self.name = name

        self.weight = weight

        self.value = value

        self.value\_per\_weight = (

            value / weight

        )  # Считаем ценность на единицу веса для сортировки

def main():

    # Считывание ввода

    n, m = map(int, input().split())

    cargos = []

    for \_ in range(m):

        data = input().split()

        name = data[0]

        weight = int(data[1])

        value = int(data[2])

        cargos.append(Cargo(name, weight, value))

    # Сортируем грузы по ценности на единицу веса в порядке убывания

    cargos.sort(key=lambda x: x.value\_per\_weight, reverse=True)

    total\_weight = 0

    result = []

    for cargo in cargos:

        if total\_weight < n:  # Пока есть место на судне

            if total\_weight + cargo.weight <= n:

                # Если помещается целиком

                total\_weight += cargo.weight

                result.append((cargo.name, cargo.weight, cargo.value))

            else:

                # Если помещается частично

                remaining\_weight = n - total\_weight

                partial\_value = (remaining\_weight / cargo.weight) \* cargo.value

                total\_weight += remaining\_weight

                result.append((cargo.name, remaining\_weight, partial\_value))

    # Вывод результата

    for name, weight, value in result:

        print(f"{name} {weight:.2f} {value:.2f}")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

**20) Произведение матриц**

Напишите функцию, которая вычисляет произведение двух квадратных матриц (A и B). При реализации нельзя использовать сторонние библиотеки (NumPy и т. д.).

**Формат ввода:**

Первая строка содержит целое число n -- размерность матрицы.

Далее идут 2n строк, содержащие по n целых чисел, разделённых пробелами, -- содержимое матриц (первые n строк относятся к матрице A, остальные -- к матрице B).

**Формат вывода:**

Произведение матриц A и B (n строк по n целых чисел, разделённых пробелами).

**Ограничения:**

2⩽n⩽102⩽n⩽10

**Название файла:** matrix\_mult.py

def matrix\_multiply(A, B, n):

    # Инициализируем результирующую матрицу C нулями

    C = [[0] \* n for \_ in range(n)]

    # Выполняем умножение матриц

    for i in range(n):

        for j in range(n):

            for k in range(n):

                C[i][j] += A[i][k] \* B[k][j]

    return C

def main():

    # Считываем размерность матриц

    n = int(input())

    if 2 <= n <= 10:

        # Считываем первую матрицу A и заполняем строки

        A = []

        for \_ in range(n):

            row = list(map(int, input().strip().split()))

            A.append(row)

        # Считываем вторую матрицу B и заполняем строки

        B = []

        for \_ in range(n):

            row = list(map(int, input().strip().split()))

            B.append(row)

        # Получаем результат умножения матриц

        C = matrix\_multiply(A, B, n)

        # Выводим результирующую матрицу

        for row in C:

            print(" ".join(map(str, row)))

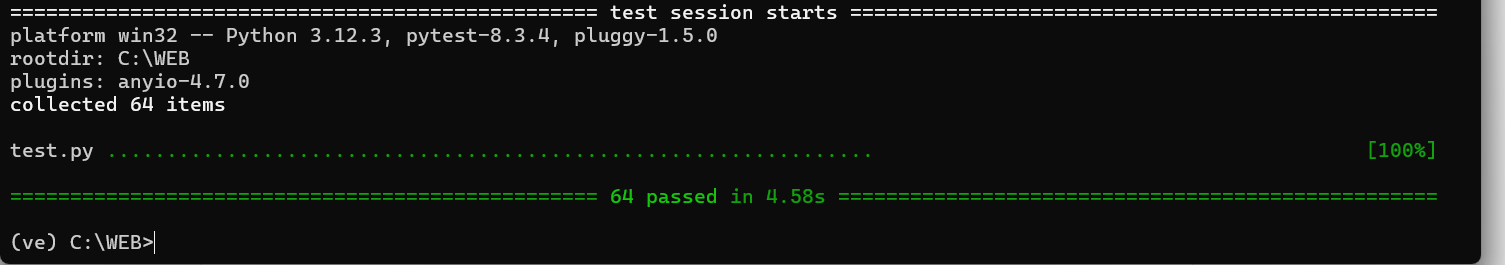
    else:

        print("Неверный размер матрицы (должен быть от 2 до 10).")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

Всего было проведено 64 тестов с помощью файла test.py:



Код тестов представлен ниже:

import subprocess

import pytest

INTERPRETER = "python"

def run\_script(filename, input\_data=None):

    proc = subprocess.run(

        [INTERPRETER, filename],

        input="\n".join(input\_data if input\_data else []),

        capture\_output=True,

        text=True,

        check=False,

    )

    return proc.stdout.strip()

test\_data = {

    "python\_if\_else": [

        ("1", "Weird"),

        ("4", "Not Weird"),

        ("3", "Weird"),

        ("6", "Weird"),

        ("22", "Not Weird"),

    ],

    "arithmetic\_operators": [

        (["1", "2"], ["3", "-1", "2"]),

        (["10", "5"], ["15", "5", "50"]),

        (["0", "5"], ["Числа должны быть в диапазоне от 1 до 10^10"]),

    ],

    "division": [

        (["4", "2"], ["2", "2.0"]),

        (["10", "6"], ["1", "1.6666666666666667"]),

        (["10", "0"], ["На ноль делить нельзя"]),

    ],

    "loops": [

        (["3"], ["0", "1", "4"]),

        (["5"], ["0", "1", "4", "9", "16"]),

        (["22"], ["Число должно быть в диапазоне от 1 до 20 включительно."]),

        (["-5"], ["Число должно быть в диапазоне от 1 до 20 включительно."]),

    ],

    "print\_function": [

        (["5"], "12345"),

        (["10"], "12345678910"),

        (["0"], "Число должно быть в диапазоне от 1 до 20 включительно."),

    ],

    "second\_score": [

        (["5", "10 10 10 10 10"], "Недостаточно данных"),

        (["7", "100 99 100 98 99 98 97"], "99"),

        (["3", "5 5 5"], "Недостаточно данных"),

    ],

    "nested\_list": [

        (

            ["4", "Anna", "4", "Borya", "4", "Sergey", "4", "Ivan", "5"],

            "Anna\nBorya\nSergey",

        ),

        (

            ["7", "Anna", "6", "Borya", "3", "Sergey", "1", "Egor", "2"],

            "Число должно быть в диапазоне от 2 до 5 включительно.",

        ),

        (["2", "Ivan", "5", "Alexey", "4"], "Alexey"),

    ],

    "lists": [

        (["7", "insert 0 5", "insert 1 10", "print"], "[5, 10]"),

        (["4", "append 6", "pop", "print"], "[]"),

        (["6", "append 10", "append 20", "remove 10", "print"], "[20]"),

    ],

    "swap\_case": [

        ("Hello World", "hELLO wORLD"),

        ("123abcDEF", "123ABCdef"),

        ("", ""),

        ("!@#$%^&\*", "!@#$%^&\*"),

    ],

    "split\_and\_join": [

        ("Hello World", "Hello-World"),

        (

            "   Hello   How   are you   ",

            "Hello-How-are-you",

        ),

        (

            "Hiii       What's a lovely day!   ",

            "Hiii-What's-a-lovely-day!",

        ),

        ("", ""),

        ("SingleWord", "SingleWord"),

    ],

    "max\_word": [

        (["example.txt"], "сосредоточенности"),

        (["example\_2.txt"], "распростился"),

        (["example\_3.txt"], "Файл пуст."),

    ],

    "price\_sum": [

        (["products.csv"], "6842.84 5891.06 0.00"),

    ],

    "anagram": [

        (["listen", "silent"], "YES"),

        (["apple", "papet"], "NO"),

        (["a", "a"], "YES"),

        (["a", "b"], "NO"),

        (["abcd", "ab"], "NO"),

    ],

    "metro": [

        (["5", "30 35", "15 26", "25 75", "30 59", "12 61", "35"], "4"),

        (["4", "15 20", "20 35", "41 56", "10 60", "21"], "2"),

    ],

    "minion\_game": [

        (["Banana"], "Стюарт 12"),

        ([" "], "Строка должна содержать от 1 до 10^6 символов."),

        (["A"], "Кевин 1"),

    ],

    "is\_leap": [

        ("1901", "False"),

        ("202", "Год должен быть в диапазоне от 1900 до 10^5."),

        ("2024", "True"),

        ("100000000000000000", "Год должен быть в диапазоне от 1900 до 10^5."),

    ],

    "is\_leap": [

        (["1901"], "False"),

        (["202"], "Год должен быть в диапазоне от 1900 до 10^5."),

        (["2024"], "True"),

    ],

    "happiness": [

        (

            [

                "12 5",

                "16 41 52 37 89 41 25 67 12 23 45 65",

                "16 20 52 23 40",

                "17 41 65 47 12",

            ],

            "-1",

        ),

        (["4 2", "10 20 30 40", "10 30", "20 40"], "0"),

        (["3 3", "100 200 300", "100 200 300", "400 500 600"], "3"),

        (["3 3", "10 20 30", "40 50 60", "10 20 30"], "-3"),

        (["1 1", "100", "200", "300"], "0"),

    ],

    "pirate\_ship": [

        (

            [

                "500 4",

                "giri 100 1500",

                "TV 70 20000",

                "bed 400 15000",

                "sofa 150 30000",

            ],

            "TV 70.00 20000.00\nsofa 150.00 30000.00\nbed 280.00 10500.00",

        ),

        (

            [

                "200 6",

                "стул 56 250",

                "стол 120 560",

                "принтер 30 1500",

                "ручка 1 5",

                "ноутбук 25 20000",

                "колонки 50 5000",

            ],

            "ноутбук 25.00 20000.00\nколонки 50.00 5000.00\nпринтер 30.00 1500.00\nручка 1.00 5.00\nстол 94.00 438.67",

        ),

    ],

    "matrix\_mult": [

        (["2", "1 2", "3 4", "5 6", "7 8"], "19 22\n43 50"),

        (["1"], "Неверный размер матрицы (должен быть от 2 до 10)."),

        (["11"], "Неверный размер матрицы (должен быть от 2 до 10)."),

    ],

}

def test\_hello\_world():

    assert run\_script("hello\_world.py") == "Hello, world!"

@pytest.mark.parametrize("input\_data, expected", test\_data["python\_if\_else"])

def test\_python\_if\_else(input\_data, expected):

    assert run\_script("python\_if\_else.py", [input\_data]) == expected

@pytest.mark.parametrize("input\_data, expected", test\_data["arithmetic\_operators"])

def test\_arithmetic\_operators(input\_data, expected):

    assert run\_script("arithmetic\_operators.py", input\_data).split("\n") == expected

@pytest.mark.parametrize("input\_data, expected", test\_data["division"])

def test\_division(input\_data, expected):

    assert run\_script("division.py", input\_data).split("\n") == expected

@pytest.mark.parametrize("input\_data, expected", test\_data["loops"])

def test\_loops(input\_data, expected):

    assert run\_script("loops.py", input\_data).split("\n") == expected

@pytest.mark.parametrize("input\_data, expected", test\_data["print\_function"])

def test\_print\_function(input\_data, expected):

    assert run\_script("print\_function.py", input\_data).strip() == expected

@pytest.mark.parametrize("input\_data, expected", test\_data["second\_score"])

def test\_second\_score(input\_data, expected):

    assert run\_script("second\_score.py", input\_data).strip() == expected

@pytest.mark.parametrize("input\_data, expected", test\_data["nested\_list"])

def test\_nested\_list(input\_data, expected):

    assert run\_script("nested\_list.py", input\_data).strip() == expected

@pytest.mark.parametrize("input\_data, expected", test\_data["lists"])

def test\_lists(input\_data, expected):

    assert run\_script("lists.py", input\_data).strip() == expected

@pytest.mark.parametrize("input\_data, expected", test\_data["swap\_case"])

def test\_swap\_case(input\_data, expected):

    assert run\_script("swap\_case.py", [input\_data]).strip() == expected

@pytest.mark.parametrize("input\_data, expected", test\_data["split\_and\_join"])

def test\_split\_and\_join(input\_data, expected):

    assert run\_script("split\_and\_join.py", [input\_data]).strip() == expected

@pytest.mark.parametrize("input\_data, expected", test\_data["max\_word"])

def test\_max\_word(input\_data, expected):

    assert run\_script("max\_word.py", input\_data).strip() == expected

@pytest.mark.parametrize("input\_data, expected", test\_data["price\_sum"])

def test\_price\_sum(input\_data, expected):

    assert run\_script("price\_sum.py", input\_data).strip() == expected

@pytest.mark.parametrize("input\_data, expected", test\_data["anagram"])

def test\_anagram(input\_data, expected):

    assert run\_script("anagram.py", input\_data).strip() == expected

@pytest.mark.parametrize("input\_data, expected", test\_data["metro"])

def test\_metro(input\_data, expected):

    assert run\_script("metro.py", input\_data).strip() == expected

@pytest.mark.parametrize("input\_data, expected", test\_data["minion\_game"])

def test\_minion\_game(input\_data, expected):

    assert run\_script("minion\_game.py", input\_data) == expected

@pytest.mark.parametrize("input\_data, expected", test\_data["is\_leap"])

def test\_is\_leap(input\_data, expected):

    assert run\_script("is\_leap.py", input\_data).strip() == expected

@pytest.mark.parametrize("input\_data, expected", test\_data["happiness"])

def test\_happiness(input\_data, expected):

    assert run\_script("happiness.py", input\_data).strip() == expected

@pytest.mark.parametrize("input\_data, expected", test\_data["pirate\_ship"])

def test\_pirate\_ship(input\_data, expected):

    assert run\_script("pirate\_ship.py", input\_data).strip() == expected

@pytest.mark.parametrize("input\_data, expected", test\_data["matrix\_mult"])

def test\_matrix\_mult(input\_data, expected):

    assert run\_script("matrix\_mult.py", input\_data).strip() == expected