**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

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

Текстовое поле

Лабораторная работа №1

**«Разработка веб-приложений»**

|  |  |
| --- | --- |
| Группа | 231-3211 |
| Студент | Чеботков Дмитрий Максимович |

Москва 2025 г.

Для запуска тестов необходимо выполнить следующие шаги:

1) Создайте виртуальное окружение:

python -m venv ve

2) Активируйте виртуальное окружение\*\*:

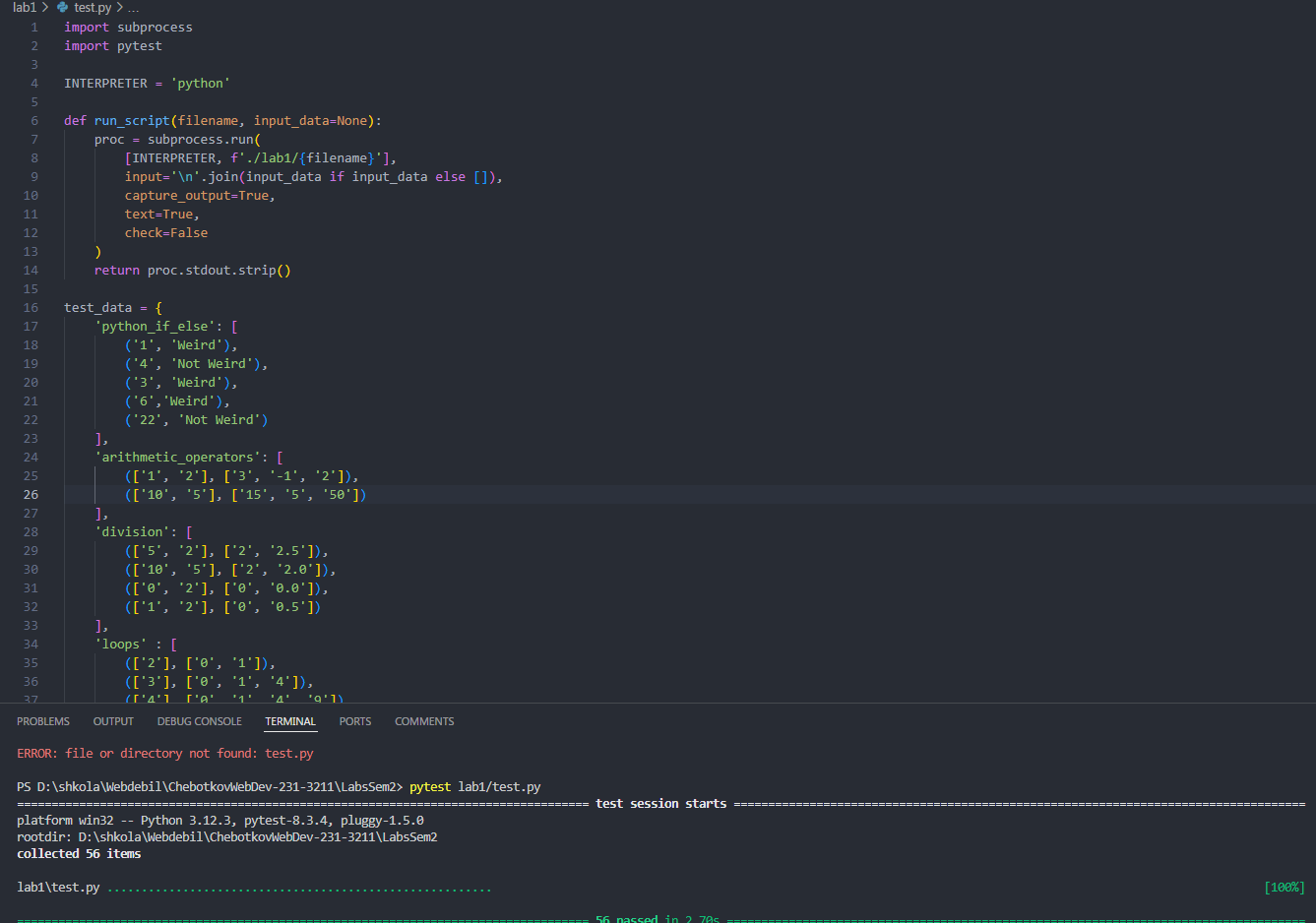
Команда для Windows: ve\Scripts\activate (в случае, если файлы хранятся в другой папке, необходимо прописать путь к этой папке. В данном случае «cd C:\WebDev»).

3) Установите pytest:

pip install pytest

4) Запустите тесты\*\*\*:

pytest test.py



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

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

Название файла: hello.py



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

n = int(input())

if n % 2 != 0:

    print("Weird")

else:

    if 2 <= n <= 5:

        print("Not Weird")

    elif 6 <= n <= 20:

        print("Weird")

    else:

        print ("Not Weird")

**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

a, b = int(input()), int(input())

print(a + b, a - b, a \* b, sep="\n")

**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

a, b = int(input()), int(input())

print(a // b, a / b, sep="\n")

**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

n = int(input())

for i in range(n):

    print(i\*\*2)

**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

n = int(input())

print(\*range(1, n + 1), sep="")

**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

n = int(input())

A = input().split()

if len(set(A)) > 1:

    print(sorted(set(A))[-2])

else:

    print("Нет второго элемента")

**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

def second\_lowest\_grade(records):

    grades = sorted(list(set([grade for \_, grade in records])))

    second\_lowest = grades[1]

    students\_with\_second\_lowest = [

        name for name, grade in records if grade == second\_lowest

    ]

    students\_with\_second\_lowest.sort()

    return students\_with\_second\_lowest

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

    n = int(input())

    records = []

    for \_ in range(n):

        name = input()

        grade = float(input())

        records.append([name, grade])

    result = second\_lowest\_grade(records)

    for student in result:

        print(student)

**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

n = int(input())

arr = []

for \_ in range(n):

    command = input().split()

    if command[0] == "insert":

        arr.insert(int(command[1]), int(command[2]))

    elif command[0] == "print":

        print(arr)

    elif command[0] == "remove":

        arr.remove(int(command[1]))

    elif command[0] == "append":

        arr.append(int(command[1]))

    elif command[0] == "sort":

        arr.sort()

    elif command[0] == "pop":

        arr.pop()

    elif command[0] == "reverse":

        arr.reverse()

**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

print(input().swapcase())

**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

n = input().split()

print('-'.join(n))

**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

with open("lab1/example.txt", 'r', encoding='utf-8') as file:

    text = file.read().lower()

    words = []

    current\_word = ""

    for char in text:

        if char.isalpha():

            current\_word += char

        elif current\_word:

            words.append(current\_word)

            current\_word = ""

    if current\_word:

        words.append(current\_word)

    max\_length = 0

    for word in words:

        max\_length = max(max\_length, len(word))

    longest\_words = [word for word in words if len(word) == max\_length]

    for word in longest\_words:

        print(word)

**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

with open("lab1/products.csv", 'r', encoding='utf-8') as file:

    next(file)

    total\_adult = 0.0

    total\_senior = 0.0

    total\_child = 0.0

    for line in file:

        parts = line.strip().split(',')

        if len(parts) >= 4:

            adult\_expense = float(parts[1].replace(',', ''))

            senior\_expense = float(parts[2].replace(',', ''))

            child\_expense = float(parts[3].replace(',', ''))

            total\_adult += adult\_expense

            total\_senior += senior\_expense

            total\_child += child\_expense

    print(f"{total\_adult:.2f}{"\n"}{total\_senior:.2f}{"\n"}{total\_child:.2f}")

**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

str1 = input()

str2 = input()

str1 = str1.replace(" ", "").lower()

str2 = str2.replace(" ", "").lower()

char\_count1 = {}

char\_count2 = {}

for char in str1:

    char\_count1[char] = char\_count1.get(char, 0) + 1

for char in str2:

    char\_count2[char] = char\_count2.get(char, 0) + 1

if char\_count1 == char\_count2:

    print("YES")

else:

    print("NO")

**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

n = int(input())

passenger\_times = []

for \_ in range(n):

    passenger\_times.append([int(i) for i in input().split()])

t = int(input())

i = 0

for time in passenger\_times:

    if time[0] <= t <= time[1]:

        i += 1

print(i)

**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

word = input("")

vowels = "AEIOUY"

consonant\_count = 0

vowel\_count = 0

for i in range(len(word)):

    if word[i] in vowels:

        vowel\_count += len(word) - i

    else:

        consonant\_count += (

            len(word) - i

        )

if consonant\_count > vowel\_count:

    print("Стюарт", consonant\_count)

elif vowel\_count > consonant\_count:

    print("Кевин", vowel\_count)

else:

    print("Ничья")

**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

def is\_leap(year):

    if year % 4 != 0:

        return False

    elif year % 100 == 0:

        return year % 400 == 0

    else:

        return True

year\_str = input()

year = int(year\_str)

result = is\_leap(year)

print(result)

**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

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

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

a = set(map(int, input().split()))

b = set(map(int, input().split()))

mood = 0

for num in arr:

    if num in a:

        mood += 1

    elif num in b:

        mood -= 1

print(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

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

items = []

for \_ in range(m):

    name, weight, value = input().split()

    weight = int(weight)

    value = int(value)

    items.append((name, weight, value))

items.sort(key=lambda x: x[2] / x[1], reverse=True)

total\_weight = 0

result = []

for name, weight, value in items:

    if total\_weight + weight <= n:

        total\_weight += weight

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

    else:

        remaining\_weight = n - total\_weight

        fraction\_value = (value / weight) \* remaining\_weight

        total\_weight += remaining\_weight

        result.append((name, remaining\_weight, round(fraction\_value)))

        break

for name, weight, value in result:

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

**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

n = int(input())

a = []

b = []

for \_ in range(n):

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

    a.append(row)

for \_ in range(n):

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

    b.append(row)

c = [[0 for \_ in range(n)] for \_ in range(n)]

for i in range(n):

    for j in range(n):

        for k in range(n):

            c[i][j] += a[i][k] \* b[k][j]

for row in c:

    print(\*row)

test.py:

import subprocess

import pytest

INTERPRETER = 'python'

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

    proc = subprocess.run(

        [INTERPRETER, f'./lab1/{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']),

        (['9', '1'], ['10', '8', '9']),

        (['6', '-3'], ['3', '9', '-18']),

    ],

    'division': [

        (['5', '2'], ['2', '2.5']),

        (['10', '5'], ['2', '2.0']),

        (['0', '2'], ['0', '0.0']),

        (['1', '2'], ['0', '0.5'])

    ],

    'loops' : [

        (['2'], ['0', '1']),

        (['3'], ['0', '1', '4']),

        (['4'], ['0', '1', '4', '9']),

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

    ],

    "print\_function": [

        (["5"], "12345"),

        (["10"], "12345678910"),

        (["3"], "123"),

        (["7"], "1234567")

    ],

    "second\_score": [

        (["5", "2 3 6 6 5"], "5"),

        (["7", "15 13 11 10 15 13 11"], "13"),

        (["3", "10 11 12"], "11")

    ],

    "nested\_list": [

        (["3", "Lena", "3", "Leha", "5", "Vova", "4"], "Vova"),

        (["4", "Lena", "3", "Leha", "5", "Vova", "4", "Egor", "4"], "Egor\nVova")

    ],

    "lists": [

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

        (["3", "append 1", "append 2", "print"], "[1, 2]"),

        (["4", "append 10", "append 15", "remove 10", "print"], "[15]"),

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

    ],

    "swap\_case": [

        ("HELLO", "hello"),

        ("abcDeF123", "ABCdEf123"),

        ("", ""),

        ("!@#", "!@#"),

        ("asdasasd", "ASDASASD")

    ],

    "split\_and\_join": [

        ("this is a string", "this-is-a-string"),

        ("1 # @ #", "1-#-@-#"),

        ("12 2 3 1", "12-2-3-1")

    ],

    "max\_word": [

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

    ],

    "price\_sum": [

        (["products.csv"], "6842.84\n5891.06\n6810.90")

    ],

    "anagram": [

        (["lalka", "kaall"], "YES"),

        (["puppy", "taras"], "NO"),

        (["oves", "esvo"], "YES"),

        (["a", "p"], "NO"),

        (["car", "ra"], "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 21"),

        (["Minion"], "Стюарт 21"),

        (["Gilbert"], "Стюарт 28")

    ],

    "is\_leap": [

        ("1951", "False"),

        ("2024", "True"),

        ("2004", "True"),

        ("1900", "False"),

        ("2100", "False")

    ],

    "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", "золото 100 1500", "серебро 70 20000", "бананы 400 15000", "металл 150 30000",],"серебро 70.00 20000.00\nметалл 150.00 30000.00\nбананы 280.00 10500.00"),

    ],

    "matrix\_mult": [

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

        (["3", "1 2 3", "4 5 5", "3 2 1", "1 2 2", "3 3 3", "5 4 6"], "22 20 26\n44 43 53\n14 16 18"),

    ],

}

def test\_hello\_world():

    assert run\_script('hello.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) == 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) == 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) == expected

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

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

    assert run\_script('lists.py', input\_data) == 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]) == 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) == 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) == expected

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

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

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

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

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

    assert run\_script('metro.py', input\_data) == 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]) == expected

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

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

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