**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

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

Текстовое поле

Лабораторная работа №2

**«Разработка веб-приложений»**

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

Москва 2025 г.

Test.py

import subprocess

import pytest

INTERPRETER = 'python'

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

    proc = subprocess.run(

        [INTERPRETER, f'./lab2/{filename}'],

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

        capture\_output=True,

        text=True,

        check=False

    )

    return proc.stdout.strip()

test\_data = {

    'fact': [

        ('5', '120'),

        ('10', '3628800'),

        ('15', '1307674368000'),

        ('20', '2432902008176640000'),

        ('25', '15511210043330985984000000'),

        ('30', '265252859812191058636308480000000'),

        ('35', '10333147966386144929666651337523200000000'),

        ('40', '815915283247897734345611269596115894272000000000')

    ],

    'show\_employee': [

        (['Иванов Иван Иванович', '30000'], 'Иванов Иван Иванович: 30000 Р'),

        (['Петров Петр Петрович', '100000'], 'Петров Петр Петрович: 100000 Р'),

        (['Летов Егор', '20000'], 'Летов Егор: 20000 Р'),

        (['Кобейн Курт', '199100'], 'Кобейн Курт: 199100 Р'),

        (['Уильям Шульц', '3500000'], 'Уильям Шульц: 3500000 Р')

    ],

    'sum\_and\_sub': [

        (['5', '3'], '8 2'),

        (['-5', '3'], '-2 -8'),

        (['-5', '-3'], '-8 -2'),

        (['5', '-3'], '2 8'),

        (['102010201020', '101010101010'], '203020302030 1000100010')

    ],

    'process\_list': [

        ('1 2 3 4 5 6 7 8 9 10', '1 4 27 16 125 36 343 64 729 100'),

        ('-1 -2 -3 -4 -5 -6 -7 -8 -9 -10', '-1 4 -27 16 -125 36 -343 64 -729 100'),

        ('1 4 8 8 1 3 3 7 2 2 8', '1 16 64 64 1 27 27 343 4 4 64'),

        ('-1 -4 -8 -8 -3 -3 -7 -2 -2 -8', '-1 16 64 64 -27 -27 -343 4 4 64'),

        ('121 18312 9291 1230 1202', '1771561 335329344 802024029171 1512900 1444804'),

    ],

    'my\_sum': [

        ('1 2 3 4 5 6 7 8 9 10', '55'),

        ('-1 -2 -3 -4 -5 -6 -7 -8 -9 -10', '-55'),

        ('1 4 8 8 1 3 3 7 2 2 8', '47'),

        ('-1 -4 -8 -8 -3 -3 -7 -2 -2 -8', '-46'),

        ('121 18312 9291 1230 1202', '30156')

    ],

    'files\_sort': [

        ('./lab2/files', ['123.py', 'abc.txt', 'text.txt'])

    ],

    'file\_search': [

        ('happiness.py', ['Найден файл: D:\\shkola\\Webdebil\\ChebotkovWebDev-231-3211\\LabsSem2\\lab1\\happiness.py',

                        '',

                        'Первые 5 строк файла:',

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

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

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

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

    ],

    'email\_validation': [

        (['3', 'lara@mospolytech.ru', 'brian-23@mospolytech.ru', 'britts\_54@mospolytech.ru'],

            "['brian-23@mospolytech.ru', 'britts\_54@mospolytech.ru', 'lara@mospolytech.ru']"),

        (['2', 'igor2233balashov@gmail.com', 'tchebotkov.dima@yandex.ru'],

            "['igor2233balashov@gmail.com', 'tchebotkov.dima@yandex.ru']"),

        (['2', 'kakaxa1805@gmail.yandex', 'kakaxa1805@gmail.com'],

            "['kakaxa1805@gmail.com']"),

        (['1', 'john\_doe@rambler.goy'],

            "['john\_doe@rambler.goy']"),

        (['3', 'john\_doe@rambler.goyda', 'john\_doe@ram-bler.goy', 'john\_doe@rambler.goy'],

            "['john\_doe@rambler.goy']"),

    ],

    'fibonacci': [

        ('1', '0'),

        ('2', '0 1'),

        ('3', '0 1 1'),

        ('4', '0 1 1 8'),

        ('5', '0 1 1 8 27'),

        ('6', '0 1 1 8 27 125'),

        ('7', '0 1 1 8 27 125 512'),

        ('8', '0 1 1 8 27 125 512 2197'),

        ('9', '0 1 1 8 27 125 512 2197 9261'),

        ('10', '0 1 1 8 27 125 512 2197 9261 39304'),

        ('11', '0 1 1 8 27 125 512 2197 9261 39304 166375'),

        ('12', '0 1 1 8 27 125 512 2197 9261 39304 166375 704969'),

        ('13', '0 1 1 8 27 125 512 2197 9261 39304 166375 704969 2985984'),

        ('14', '0 1 1 8 27 125 512 2197 9261 39304 166375 704969 2985984 12649337'),

        ('15', '0 1 1 8 27 125 512 2197 9261 39304 166375 704969 2985984 12649337 53582633')

    ],

    'average\_scores': [

            (['5 3', '89 90 78 93 80', '90 91 85 88 86', '91 92 83 89 90.5'], ['90.0', '91.0', '82.0', '90.0', '85.5'])

        ],

        'plane\_angle': [

        (['0 0 0', '1 0 0', '1 1 0', '1 1 1'], '90.00'),

        (['-1 0 0', '1 0 0', '-1 1 0', '1 1 1'], '48.19'),

        (['4 5 6', '1 2 3', '1 4 8', '2 2 8'], '89.18'),

        (['0 -1 0', '-1 0 0', '0 0 -1', '0 0 0'], '125.26'),

        (['0.5 21 3', '-1 6 8', '-123 3 2', '10 123 -9'], '169.83')

    ],

    'phone\_number': [

        (['3', '07895462130', '89875641230', '9195969878'],

            ['+7 (789) 546-21-30', '+7 (919) 596-98-78', '+7 (987) 564-12-30']),

        (['4', '07895362230', '89815631250', '9195369832', '89324532312'],

            ['+7 (789) 536-22-30', '+7 (919) 536-98-32', '+7 (932) 453-23-12', '+7 (981) 563-12-50'])

    ],

    'people\_sort': [

        (['3', 'Mike Thomson 20 M', 'Robert Bustle 32 M', 'Andria Bustle 30 F'],

            ['Mr. Mike Thomson', 'Ms. Andria Bustle', 'Mr. Robert Bustle'])

    ],

    'complex\_numbers': [

        (['2 1', '5 6'], ['7.00+7.00i', '-3.00-5.00i', '4.00+17.00i', '0.26-0.11i', '2.24+0.00i', '7.81+0.00i'])

    ]

}

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

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

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

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

def test\_show\_employee(input\_data, expected):

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

@pytest.mark.parametrize('input\_data, expected', test\_data['sum\_and\_sub'])

def test\_sum\_and\_sub(input\_data, expected):

    assert run\_script('sum\_and\_sub.py', input\_data) == expected

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

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

    assert run\_script('process\_list.py', [input\_data]) == expected

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

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

    assert run\_script('my\_sum.py', [input\_data]) == expected

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

def test\_files\_sort(input\_data, expected):

    assert run\_script('files\_sort.py', [input\_data]).split('\n') == expected

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

def test\_file\_search(input\_data, expected):

    assert run\_script('file\_search.py', [input\_data]).split('\n') == expected

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

def test\_email\_validation(input\_data, expected):

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

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

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

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

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

def test\_average\_scores(input\_data, expected):

    assert run\_script('average\_scores.py', input\_data).split('\n') == expected

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

def test\_plane\_angle(input\_data, expected):

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

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

def test\_phone\_number(input\_data, expected):

    assert run\_script('phone\_number.py', input\_data).split('\n') == expected

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

def test\_people\_sort(input\_data, expected):

    assert run\_script('people\_sort.py', input\_data).split('\n') == expected

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

def test\_complex\_numbers(input\_data, expected):

    assert run\_script('complex\_numbers.py', input\_data).split('\n') == expected

**1) fact.py**

import timeit

def fact\_rec(n):

    if n == 1:

        return 1

    return n \* fact\_rec(n - 1)

def fact\_it(n):

    result = 1

    for i in range(1, n + 1):

        result \*= i

    return result

if \_\_name\_\_ == '\_\_main\_\_':

    n = int(input())

    print(fact\_rec(n))

'''Рекурсивная версия: 0.00000048 сек за вызов

Итеративная версия: 0.00000031 сек за вызов'''

**2)** **show\_employee.py**

def show\_employee(name, salary=100000):

    return f"{name}: {salary} Р"

if \_\_name\_\_ == '\_\_main\_\_':

    a, b = input(), int(input())

    print(show\_employee(a, b))

**3)** **sum\_and\_sub.py**

def sum\_and\_sub(a, b):

    print(a + b, a - b)

if \_\_name\_\_ == '\_\_main\_\_':

    a,b = int(input()), int(input())

    sum\_and\_sub(a, b)

**4)** **process\_list.py**

import timeit

*#четные в квадрат, нечетные в куб*

*# Версия с List Comprehension: 0.00000054 сек за вызов*

def process\_list(arr):

    result = [i \*\* 2 if i % 2 == 0 else i \*\* 3 for i in arr]

    return result

*# Версия-генератор: 0.00000015 сек за вызов*

def process\_list\_gen(arr):

    return (i \*\* 2 if i % 2 == 0 else i \*\* 3 for i in arr)

if \_\_name\_\_ == '\_\_main\_\_':

*#*

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

    print(\*list(process\_list\_gen(nums)))

'''Версия с List Comprehension: 0.00000068 сек за вызов

Версия-генератор: 0.00000016 сек за вызов'''

**5)** **my\_sum.py**

def my\_sum(\*args):

    total = 0

    for number in args:

        total += number

    return total

if \_\_name\_\_ == '\_\_main\_\_':

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

    result = my\_sum(\*nums)

    print(result)

**6)** **my\_sum\_argv.py**

import sys

def my\_sum(\*args):

    return sum(args)

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

    if len(sys.argv) > 1:

        numbers = map(int, sys.argv[1:])

        result = my\_sum(\*numbers)

        print(result)

    else:

        number = [int(num) for num in input("Введите числа через пробел: ").split()]

        result = my\_sum(\*number)

        print(result)

*#python lab2/my\_sum\_argv.py 1 2 3 4 5*

**7) files\_sort.py**

import os

def list\_files\_by\_extension(directory):

    files = [f for f in os.listdir(directory) if os.path.isfile(os.path.join(directory, f))]

    files.sort()

    extensions = sorted(list(set([f.split('.')[-1] for f in files])))

    for ext in extensions:

        for file in files:

            if file.endswith(f'.{ext}'):

                print(file)

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

    directory\_path = input("")

    if not os.path.isdir(directory\_path):

        print("Ошибка: Указанный путь не является директорией.")

        exit(1)

    list\_files\_by\_extension(directory\_path)

**8) file\_search.py**

import os

def find\_file(filename, directory):

    for root, \_, files in os.walk(directory):

        if filename in files:

            return os.path.join(root, filename)

    return None

def print\_first\_lines(filepath, num\_lines=5):

    try:

        with open(filepath, 'r', encoding='utf-8') as f:

            for i, line in enumerate(f):

                if i >= num\_lines:

                    break

                print(line, end='')

    except FileNotFoundError:

        print(f"Файл '{filepath}' не найден.")

    except Exception as e:

        print(f"Произошла ошибка при чтении файла: {e}")

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

    filename = input()

*# Получаем текущую директорию*

    current\_directory = os.getcwd()

*# Ищем файл*

    filepath = find\_file(filename, current\_directory)

    if filepath:

        print(f"\nНайден файл: {filepath}")

        print("\nПервые 5 строк файла:")

        print\_first\_lines(filepath)

    else:

        print(f"\nФайл '{filename}' не найден.")

**9)** **email\_validation.py**

import re

def fun(s):

    pattern = r'^[a-zA-Z0-9.\_-]+@[a-zA-Z0-9]+\.[a-zA-Z]{1,3}$'

    return re.match(pattern, s) is not None

def filter\_mail(emails):

    return list(filter(fun, emails))

if \_\_name\_\_ == '\_\_main\_\_':

    n = int(input())

    emails = []

    for \_ in range(n):

        emails.append(input())

    filtered\_emails = filter\_mail(emails)

    filtered\_emails.sort()

    print(filtered\_emails)

**10)** **fibonacci.py**

cube = lambda x: x\*\*3

def fibonacci(n):

    fib\_list = [0, 1]

    for i in range(2, n):

        fib\_list.append(fib\_list[i-1] + fib\_list[i-2])

    return fib\_list[:n]

if \_\_name\_\_ == '\_\_main\_\_':

    n = int(input())

    print(\*list(map(cube, fibonacci(n))))

**11) average\_scores.py**

def compute\_average\_scores(scores):

    num\_students = len(scores[0])

    average\_scores = []

    for i in range(num\_students):

        student\_scores = [scores[j][i] for j in range(len(scores))] *#все оценки конкретного студента по всем предметам*

        average\_score = sum(student\_scores) / len(student\_scores)

        average\_scores.append(average\_score)

    return average\_scores

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

    n, x = map(int, input().split())

    scores = []

    for \_ in range(x):

        subject\_scores = list(map(float, input().split()))

        scores.append(subject\_scores)

    average\_scores = compute\_average\_scores(scores)

    for score in average\_scores:

        print(f"{score:.1f}")

**12) plane\_angle.py**

import math

class Point:

    def \_\_init\_\_(self, x, y, z):

        self.x = x

        self.y = y

        self.z = z

    def \_\_sub\_\_(self, other):

*# Вычитание векторов*

        return Point(self.x - other.x, self.y - other.y, self.z - other.z)

    def dot(self, other):

*# Скалярное произведение*

        return self.x \* other.x + self.y \* other.y + self.z \* other.z

    def cross(self, other):

*# Векторное произведение*

        return Point(self.y \* other.z - self.z \* other.y,

                     self.z \* other.x - self.x \* other.z,

                     self.x \* other.y - self.y \* other.x)

    def absolute(self):

*# Модуль вектора*

        return math.sqrt(self.x \*\* 2 + self.y \*\* 2 + self.z \*\* 2)

def plane\_angle(a, b, c, d):

*# Векторы AB, BC, CD*

    ab = b - a

    bc = c - b

    cd = d - c

*# Векторные произведения AB × BC и BC × CD*

    x = ab.cross(bc)

    y = bc.cross(cd)

*# Скалярное произведение X · Y*

    dot\_product = x.dot(y)

*# Модули векторов X и Y*

    x\_magnitude = x.absolute()

    y\_magnitude = y.absolute()

*# Косинус угла между X и Y*

    cos\_phi = dot\_product / (x\_magnitude \* y\_magnitude)

*# Угол в радианах -> градусы*

    angle = 180 - math.degrees(math.acos(cos\_phi))

    return angle

if \_\_name\_\_ == '\_\_main\_\_':

    a = Point(\*map(float, input().split()))

    b = Point(\*map(float, input().split()))

    c = Point(\*map(float, input().split()))

    d = Point(\*map(float, input().split()))

    result = plane\_angle(a, b, c, d)

    print(f"{result:.2f}")

**13)** **phone\_number.py**

def wrapper(f):

    def fun(l):

*# Преобразуем каждый номер телефона в стандартный формат*

        formatted\_numbers = []

        for number in l:

            clean\_number = number[-10:]  *# Извлекаем последние 10 цифр*

            formatted\_number = f"+7 ({clean\_number[:3]}) {clean\_number[3:6]}-{clean\_number[6:8]}-{clean\_number[8:]}"

            formatted\_numbers.append(formatted\_number)

        return f(formatted\_numbers)

    return fun

@wrapper

def sort\_phone(l):

    return sorted(l)

if \_\_name\_\_ == '\_\_main\_\_':

    l = [input() for \_ in range(int(input()))]

    print(\*sort\_phone(l), sep='\n')

'''3

07895462130

89875641230

9195969878'''

**14) people\_sort.py**

import operator

def person\_lister(f):

    def inner(people):

*# Сортируем список по возрасту (поле person[2], преобразованное в int)*

        return map(f, sorted(people, key=lambda x: int(x[2])))

    return inner

@person\_lister

def name\_format(person):

*# Форматируем вывод в зависимости от пола*

    return ("Mr. " if person[3] == "M" else "Ms. ") + person[0] + " " + person[1]

if \_\_name\_\_ == '\_\_main\_\_':

*# Ввод данных: количество людей и их параметры*

    people = [input().split() for i in range(int(input()))]

    print(\*name\_format(people), sep='\n')

"""3

Mike Thomson 20 M

Robert Bustle 32 M

Andria Bustle 30 F"""

**15) complex\_numbers.py**

import math

class Complex(object):

    def \_\_init\_\_(self, real, imaginary):

        self.real = real

        self.imaginary = imaginary

*#сложение комплексных чисел*

    def \_\_add\_\_(self, no):

        return Complex(self.real + no.real, self.imaginary + no.imaginary)

    def \_\_sub\_\_(self, no):

        return Complex(self.real - no.real, self.imaginary - no.imaginary)

    def \_\_mul\_\_(self, no):

        real = self.real \* no.real - self.imaginary \* no.imaginary

        imaginary = self.real \* no.imaginary + self.imaginary \* no.real

        return Complex(real, imaginary)

    def \_\_truediv\_\_(self, no):

        denom = no.real \*\* 2 + no.imaginary \*\* 2 *#знаменатель деления*

        real = (self.real \* no.real + self.imaginary \* no.imaginary) / denom

        imaginary = (self.imaginary \* no.real - self.real \* no.imaginary) / denom

        return Complex(real, imaginary)

    def mod(self): *#модуль комплексного числа*

        return Complex(math.sqrt(self.real \*\* 2 + self.imaginary \*\* 2), 0)

    def \_\_str\_\_(self):

        if self.imaginary == 0:

            result = "%.2f+0.00i" % (self.real)

        elif self.real == 0:

            if self.imaginary >= 0:

                result = "0.00+%.2fi" % (self.imaginary)

            else:

                result = "0.00-%.2fi" % (abs(self.imaginary))

        elif self.imaginary > 0:

            result = "%.2f+%.2fi" % (self.real, self.imaginary)

        else:

            result = "%.2f-%.2fi" % (self.real, abs(self.imaginary))

        return result

if \_\_name\_\_ == '\_\_main\_\_':

*# Считываем входные данные*

    c = map(float, input().split())

    d = map(float, input().split())

    x = Complex(\*c)

    y = Complex(\*d)

*# Выводим результат операций*

    print(\*map(str, [x + y, x - y, x \* y, x / y, x.mod(), y.mod()]), sep='\n')

**16) circle\_square\_mk.py**

import random

import math

def circle\_square\_mk(r, n):

    count\_inside\_circle = 0

*# Генерация n случайных точек в квадрате [-r, r] x [-r, r]*

    for \_ in range(n):

        x = random.uniform(-r, r)

        y = random.uniform(-r, r)

*# Проверяем, лежит ли точка внутри окружности*

        if x\*\*2 + y\*\*2 <= r\*\*2:

            count\_inside\_circle += 1

*# Площадь квадрата со стороной 2r*

    square\_area = (2 \* r) \*\* 2

*# Доля точек, попавших внутрь окружности*

    fraction\_inside\_circle = count\_inside\_circle / n

*# Приближённая площадь окружности*

    circle\_area = fraction\_inside\_circle \* square\_area

    return circle\_area

*# Тестирование функции*

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

    radius = 5  *# Радиус окружности*

    experiments = 100000  *# Количество экспериментов*

    monte\_carlo\_area = circle\_square\_mk(radius, experiments)

*# Точная площадь окружности*

    exact\_area = math.pi \* radius \*\* 2

*# Оценка погрешности*

    error = abs(monte\_carlo\_area - exact\_area) / exact\_area \* 100

*# Результаты*

    print(f"Радиус окружности: {radius}")

    print(f"Количество экспериментов: {experiments}")

    print(f"Вычисленная площадь (Монте-Карло): {monte\_carlo\_area}")

    print(f"Точная площадь: {exact\_area}")

    print(f"Погрешность: {error:.5f}%")

**17) log\_decorator.py**

import os

import time

from datetime import datetime

def function\_logger(log\_file):

    def decorator(func):

        def wrapper(\*args, \*\*kwargs):

*# Получение текущего времени в указанном формате*

            start\_time = datetime.now()

            start\_time\_str = start\_time.strftime('%Y-%m-%d %H:%M:%S.%f')

*# Логирование названия функции*

            log\_data = [func.\_\_name\_\_]

*# Логирование времени начала выполнения*

            log\_data.append(start\_time\_str)

*# Логирование входных аргументов*

            log\_data.append(repr(args))

            log\_data.append(repr(kwargs))

*# Выполнение функции*

            try:

                result = func(\*args, \*\*kwargs)

            except Exception as e:

                result = f"Exception: {e}"

            finally:

                end\_time = datetime.now()

                duration = end\_time - start\_time

                end\_time\_str = end\_time.strftime('%Y-%m-%d %H:%M:%S.%f')

*# Добавление результата в логи*

                log\_data.append(result if result is not None else "-")

*# Логирование времени завершения*

                log\_data.append(end\_time\_str)

*# Логирование продолжительности выполнения*

                log\_data.append(str(duration))

*# Запись в файл*

                with open(log\_file, 'a') as f:

                    for entry in log\_data:

                        f.write(str(entry) + '\n')

                    f.write('\n')  *# Разделитель для записи*

            return result

        return wrapper

    return decorator

*# Пример использования*

@function\_logger('test.log')

def greeting\_format(name):

    return f"Hello, {name}!"

greeting\_format('John')