

**ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ**

**ОТЧЕТ
по практическим работам
по дисциплине «Современные технологии разработки программного
обеспечения»**

Студент

Кривов Д.Е.

Воронеж

2025

Практическая работа №1. Введение в язык программирования Python

Цель работы: познакомиться со средой разработки Python. Изучить основные типы данных, команды ввода и вывода данных

Задание:

Напишите программу, которая запрашивала бы у пользователя: Имя, Фамилия, Возраст, Место жительства - фамилия, имя ("Ваши фамилия, имя?") - возраст ("Сколько Вам лет?") - место жительства ("Где вы живете?")

После этого выводила бы три строки: "Ваши фамилия, имя" "Ваш возраст" "Вы живете в"

Код программы:

```
a = input('Ваши фамилия, имя?')
```

```
b = input("Сколько Вам лет?")
```

```
c = input('Где вы живете?')
```

```
print('Ваше имя', a)
```

```
print('Ваш возраст', b)
```

```
print('Вы живёте', c)
```

Скриншот выполнения:

The screenshot shows the PyCharm IDE interface. On the left, the project structure is visible with a file named 'script.py'. In the center, the code editor contains the following Python script:

```
a = input('Ваши фамилия, имя?')
b = input("Сколько Вам лет?")
c = input('Где вы живете?')

print('Ваше имя', a)
print('Ваш возраст', b)
print('Вы живёте', c)
```

Below the code editor is a terminal window titled 'Run' with the tab 'script'. The terminal output shows the execution of the script:

```
C:\Users\user\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\user\PyCharmMiscProject\script.py
Ваши фамилия, имя? Кривов Дмитрий
Сколько Вам лет? 36
Где вы живете? в Воронеже
Ваше имя Кривов Дмитрий
Ваш возраст 36
Вы живёте в Воронеже

Process finished with exit code 0
```

Практическая работа №2. Математические операции в Python

Цель работы: познакомиться с основными математическими операциями в Python

Задание: реализовать математическую функцию в Python

$$4. s = |\cos x - \cos y|^{(1+2\sin^2 y)} \left(1 + z + \frac{z^2}{2} + \frac{z^3}{3} + \frac{z^4}{4} \right)$$

Вычислить значение s при заданных значениях x, y, z.

Код программы.

```
import math
```

```
x = float(input("Введите переменную x: "))
```

```
y = float(input("Введите переменную y: "))
```

```
z = float(input("Введите переменную z: "))
```

```
# Вычисление выражения s = |\cos x - \cos y|^{(1 + 2*\sin^2 y)} * (1 + z + z^2/2 + z^3/3 + z^4/4)
```

```
numerator = math.fabs(math.cos(x) - math.cos(y))
```

```
power = 1 + 2 * (math.sin(y) ** 2)
```

```
polynomial = 1 + z + (z**2)/2 + (z**3)/3 + (z**4)/4
```

```
s = math.pow(numerator, power) * polynomial
```

```
print("s ", s)
```

Скриншот выполнения:

Project ▾

- PyCharmMiscProject C:\Users\user
- .venv library root
 - prac1.py
 - prac2.py
- External Libraries
- Scratches and Consoles

Welcome to PyCharm prac1.py prac2.py ×

```
1 import math
2
3 x = float(input("Введите переменную x: "))
4 y = float(input("Введите переменную у: "))
5 z = float(input("Введите переменную z: "))
6
7 # Вычисление выражения s = |cos x - cos y|^(1 + 2*sin^2 y) * (1 + z + z
8 numerator = math.fabs(math.cos(x) - math.cos(y))
9 power = 1 + 2 * (math.sin(y) ** 2)
10 polynomial = 1 + z + (z**2)/2 + (z**3)/3 + (z**4)/4
11
12 s = math.pow(numerator, power) * polynomial
13
14 print("s ", s)
```

Run prac2 ×

C:\Users\user\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\user\PyCharmMiscProject\prac2.py

Ведите переменную x: 4000
Ведите переменную у: -0.875
Ведите переменную z: -0.000475
s 1.9872670808332158

Process finished with exit code 0

Практическая работа № 3. Структура ветвление в Python

Цель работы: познакомиться со структурой ветвление (if, if-else, if-elif-else).

Научиться работать с числами и строками используя данную структуру.

Задание:

4. Задание: Проверка числа на простоту.

Код программы:

```
num = int(input("Введите число: "))

if num <= 1:
    print(f"{num} не является простым числом")
else:
    is_prime = True

    for i in range(2, int(num ** 0.5) + 1):
        if num % i == 0:
            is_prime = False
            break

    if is_prime:
        print(f"{num} является простым числом")
    else:
        print(f"{num} не является простым числом")
```

Скриншот:

The screenshot shows the PyCharm IDE interface. On the left, the Project tool window displays a project named "PyCharmMiscProject" containing files ".venv library root", "prac1.py", "prac2.py", and "prac3.py". The "prac3.py" file is selected. On the right, the Editor tool window shows the following Python code:

```
1 num = int(input("Введите число: "))
2 if num <= 1:
3     print(f"{num} не является простым числом")
4 else:
5     is_prime = True
6
7 for i in range(2, int(num ** 0.5) + 1):
8     if num % i == 0:
9         is_prime = False
10        break
11
12 if is_prime:
13     print(f"{num} является простым числом")
14 else:
15     print(f"{num} не является простым числом")
```

Below the Editor is the Run tool window, which shows the current configuration is "prac3". The Run tab is active. The Run History section shows the command "C:\Users\user\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\user\PyCharmMiscProject\prac3.py" and the output:
Введите число: 13
13 является простым числом
Process finished with exit code 0

Практическая работа №4. Циклы

Задание:

4. Дано несколько чисел. Вычислите их сумму. Сначала введите количество чисел n , затем вводится ровно n целых чисел. Постройте решение так, чтобы использовалось минимальное количество переменных.

Код программы:

```
n = int(input())
```

```
s = 0
```

```
for _ in range(n):
```

```
    s += int(input())
```

```
print(s)
```

Скриншот выполнения:

The screenshot shows the PyCharm IDE interface. On the left, the Project tool window displays a project named 'PyCharmMiscProject' containing files 'prac1.py', 'prac2.py', 'prac3.py', and 'prac4.py'. The file 'prac4.py' is currently selected and shown in the main editor area. The code for 'prac4.py' is:

```
n = int(input())
s = 0

for _ in range(n):
    s += int(input())

print(s)
```

In the bottom right corner, the terminal window shows the execution of the program. It prompts for input, receives four integers (4, 1, 2, 3), adds them up, and prints the result (10). The process exits with code 0.

```
C:\Users\user\PyCharmMiscProject\.venv\Scripts\python.exe C:/Users/user/PyCharmMiscProject/prac4.py
4
1
2
3
4
10

Process finished with exit code 0
```

Практическая работа №5. Работа со строками в Python

Цель работы: познакомится с методами работы со строками.

Задание:

В строке заменить букву(а) буквой (о). Подсчитать количество замен. Подсчитать, сколько символов в строке.

Код программы:

```
s = input("Введите строку: ")
total_symbols = len(s)
count_a = s.count('a')
s_replaced = s.replace('a', 'o')

print("Строка после замены:", s_replaced)
print("Количество замен:", count_a)
print("Количество символов в строке:", total_symbols)
```

Скриншот:

The screenshot shows the PyCharm IDE interface. On the left, the Project tool window displays a project named 'PyCharmMiscProject' containing files: .venv, prac1.py, prac2.py, prac3.py, prac4.py, and prac5.py. The file 'prac5.py' is currently selected and shown in the main code editor area. The code in 'prac5.py' is identical to the one provided above. Below the code editor is the Run tool window, which shows the run configuration 'prac5'. The bottom part of the interface is the terminal window, which displays the execution of the script. The terminal output is as follows:

```
C:\Users\user\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\user\PyCharmMiscProject\prac5.py
Введите строку: ооооо123ккк
Строка после замены: оoooo123ккк
Количество замен: 3
Количество символов в строке: 11
Process finished with exit code 0
```

Практическая работа №6. Работа со списками. Операции над списками в Python

Цель работы: Изучение одномерных массивов в Python.

Задание:

Вариант 4

1. Дан массив целых чисел. Найти максимальный элемент массива и его порядковый номер.
2. Дан одномерный массив целого типа. Получить другой массив, состоящий только из нечетных чисел исходного массива или сообщить, что таких чисел нет. Полученный массив вывести в порядке убывания элементов.

Код программы:

```
arr = list(map(int, input("Введите элементы массива через пробел: ").split()))

if arr:
    max_element = arr[0]
    max_index = 0

    for i in range(1, len(arr)):
        if arr[i] > max_element:
            max_element = arr[i]
            max_index = i

    print(f"Максимальный элемент: {max_element}")
    print(f"Порядковый номер (с 1): {max_index + 1}")
    print(f"Индекс (с 0): {max_index}")

else:
    print("Массив пуст!")
```

Скриншот:

```
arr = list(map(int, input("Введите элементы массива через пробел: ").split()))
if arr:
    max_element = arr[0]
    max_index = 0

    for i in range(1, len(arr)):
        if arr[i] > max_element:
            max_element = arr[i]
            max_index = i

    print(f"Максимальный элемент: {max_element}")
    print(f"Порядковый номер (с 1): {max_index + 1}")
    print(f"Индекс (с 0): {max_index}")
else:
    print("Массив пуст!")
```

Run prac6

```
C:\Users\user\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\user\PyCharmMiscProject\prac6.py
Введите элементы массива через пробел: 2 3 1 2 4 6 3
Максимальный элемент: 6
Порядковый номер (с 1): 7
Индекс (с 0): 5
Process finished with exit code 0
```

Код программы 2:

```
arr = list(map(int, input("Введите элементы массива через пробел: ").split()))
```

```
odd_numbers = []
for num in arr:
    if num % 2 != 0: # Проверка на нечетность
        odd_numbers.append(num)

if odd_numbers:
    odd_numbers.sort(reverse=True)
    print("Массив нечетных чисел в порядке убывания:", odd_numbers)
else:
    print("Нечетных чисел в исходном массиве нет")
```

Скриншот:

Project

PyCharmMiscProject C:\Users\user\PyCharmMiscProject

.venv library root

prac1.py
prac2.py
prac3.py
prac4.py
prac5.py
prac6.py

External Libraries

Scratches and Consoles

Welcome to PyCharm

prac1.py prac2.py prac3.py prac4.py prac5.py prac6.py

```
1 arr = list(map(int, input("Введите элементы массива через пробел: ").split()))
2
3 odd_numbers = []
4 for num in arr:
5     if num % 2 != 0: # Проверка на нечетность
6         odd_numbers.append(num)
7
8 if odd_numbers:
9     odd_numbers.sort(reverse=True)
10    print("Массив нечетных чисел в порядке убывания:", odd_numbers)
11 else:
12    print("Нечетных чисел в исходном массиве нет")
13
```

Run

prac6

C:\Users\user\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\user\PyCharmMiscProject\prac6.py

Введите элементы массива через пробел: 3 2 4 7 5 2 3 4 5

Массив нечетных чисел в порядке убывания: [7, 5, 5, 3, 3]

Process finished with exit code 0

Практическая работа №7. Функции и процедуры в Python

Цель работы: изучение процедур и функций в Python

Задание:

Вариант 4.

1. Даны две дроби A/B и C/D (A, B, C, D — натуральные числа). Составить программу деления дроби на дробь. Ответ должен быть несократимой дробью. Использовать подпрограмму алгоритма Евклида для определения НОД.
2. Задана окружность $(x-a)^2 + (y-b)^2 = R^2$ и точки P(p1, p2), F(f1, f1), L(l1, l2). Выяснить и вывести на экран, сколько точек лежит внутри окружности. Проверку, лежит ли точка внутри окружности, оформить в виде процедуры.

Код программы 1:

```
def euclid_gcd(m, n):
```

```
    while n:
```

```
        m, n = n, m % n
```

```
    return m
```

```
A = int(input())
```

```
B = int(input())
```

```
C = int(input())
```

```
D = int(input())
```

```
num = A * D
```

```
den = B * C
```

```
gcd_value = euclid_gcd(num, den)
```

```
num_simplified = num // gcd_value
```

```
den_simplified = den // gcd_value
```

```
print(f'{num_simplified}/{den_simplified}')
```

Скриншот:

```
def euclid_gcd(m, n): 1 usage
    while n:
        m, n = n, m % n
    return m

A = int(input())
B = int(input())
C = int(input())
D = int(input())

num = A * D
den = B * C

gcd_value = euclid_gcd(num, den)

num_simplified = num // gcd_value
den_simplified = den // gcd_value

print(f"{num_simplified}/{den_simplified}")
```

Run prac71

```
C:\Users\user\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\user\PyCharmMiscProject\prac71.py
1
3
2
5
5/6
Process finished with exit code 0
```

Код программы 2:

```
def check_point_in_circle(px, py, a, b, r):
    return (px - a)**2 + (py - b)**2 < r**2
```

```
a = float(input("Введите a: "))
b = float(input("Введите b: "))
r = float(input("Введите R: "))
```

```
p1 = float(input("Введите p1: "))
p2 = float(input("Введите p2: "))
f1 = float(input("Введите f1: "))
f2 = float(input("Введите f2: "))
l1 = float(input("Введите l1: "))
l2 = float(input("Введите l2: "))
```

```
points = [(p1, p2, 'P'), (f1, f2, 'F'), (l1, l2, 'L')]
inside_count = 0
```

for x, y, name in points:

```
if check_point_in_circle(x, y, a, b, r):
```

```
    print(f"Точка {name} лежит внутри окружности.")
```

```
    inside_count += 1
```

else:

```
    print(f"Точка {name} не лежит внутри окружности.")
```

```
print(f"Количество точек внутри окружности: {inside_count}")
```

Скриншот:

The screenshot shows the PyCharm IDE interface. On the left, the Project tool window displays a file structure for a project named 'PyCharmMiscProject' located at 'C:\Users\user'. The 'prac72.py' file is selected and shown in the code editor. The code implements a function to check if points lie within a circle and prints the count of such points. In the bottom right, the Run tool window shows the terminal output of running 'prac72.py'. The user inputs coordinates and labels for three points: P(1, 2), F(0.4, 1.5), and L(3, 0.1). Points P and F are inside the circle with center (0, 0) and radius 0.5, while point L is outside.

```
def check_point_in_circle(px, py, a, b, r): 1 usage
    return (px - a) ** 2 + (py - b) ** 2 < r ** 2

a = float(input("Введите a: "))
b = float(input("Введите b: "))
r = float(input("Введите R: "))

p1 = float(input("Введите p1: "))
p2 = float(input("Введите p2: "))
f1 = float(input("Введите f1: "))
f2 = float(input("Введите f2: "))
l1 = float(input("Введите l1: "))
l2 = float(input("Введите l2: "))

points = [(p1, p2, 'P'), (f1, f2, 'F'), (l1, l2, 'L')]
inside_count = 0

for x, y, name in points:
    if check_point_in_circle(x, y, a, b, r):
        print(f"Точка {name} лежит внутри окружности.")
    inside_count += 1

Количество точек внутри окружности: 2
```

```
C:\Users\user\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\user\PyCharmMiscProject\prac72.py
Введите a: 1
Введите b: 2
Введите R: 2
Введите p1: 0.5
Введите p2: 0.4
Введите f1: 2
Введите f2: 1.5
Введите l1: 3
Введите l2: 0.1
Точка P лежит внутри окружности.
Точка F лежит внутри окружности.
Точка L не лежит внутри окружности.
Количество точек внутри окружности: 2
```

Лабораторная работа №8. Работа с двумерными массивами

Цель работы: изучение двумерных массивов в Python.

Задание:

Вариант 4.

1. Данна прямоугольная матрица. Найти строку с наибольшей и строку с наименьшей суммой элементов. Вывести на печать найденные строки и суммы их элементов.
2. Данна квадратная матрица $A[N, N]$, Записать на место отрицательных элементов матрицы нули, а на место положительных — единицы. Вывести на печать нижнюю треугольную матрицу в общепринятом виде.

Код программы 1:

```
m = int(input("Введите количество строк: "))

n = int(input("Введите количество столбцов: "))

matrix = []
for i in range(m):
    row = list(map(int, input(f"Введите {n} элементов {i+1}-й строки через пробел: ").split()))
    matrix.append(row)

max_sum = float('-inf')
min_sum = float('inf')
max_row_index = -1
min_row_index = -1

for i in range(m):
    row_sum = sum(matrix[i])
    if row_sum > max_sum:
        max_sum = row_sum
        max_row_index = i
    if row_sum < min_sum:
        min_sum = row_sum
        min_row_index = i
```

```

if row_sum < min_sum:
    min_sum = row_sum
    min_row_index = i

print("\nСтрока с наибольшей суммой элементов:")
print(f"Строка {max_row_index + 1}: {matrix[max_row_index]}, Сумма = {max_sum}")

print("\nСтрока с наименьшей суммой элементов:")
print(f"Строка {min_row_index + 1}: {matrix[min_row_index]}, Сумма = {min_sum}")

```

Скриншот:

The screenshot shows the PyCharm IDE interface. On the left, the Project tool window displays a file structure for 'PyCharmMiscProject' containing several Python files: c1.py, prac2.py, prac3.py, prac4.py, prac5.py, prac6.py, prac71.py, prac72.py, and prac81.py. The file 'prac81.py' is currently selected and shown in the main editor area. The code in 'prac81.py' is identical to the one provided above. Below the editor is the 'Run' tool window, which shows the execution of 'prac81'. The terminal output shows the user inputting matrix dimensions (3 rows, 4 columns) and then entering three rows of 4-element lists. The program then prints the row with the maximum sum (row 3, sum 21) and the row with the minimum sum (row 2, sum 17).

```

1 m = int(input("Введите количество строк: "))
2 n = int(input("Введите количество столбцов: "))
3
4 matrix = []
5 for i in range(m):
6     row = list(map(int, input(f"Введите {n} элементов {i+1}-й строки через пробел: ").split()))
7     matrix.append(row)
8
9 max_sum = float('-inf')
10 min_sum = float('inf')
11 max_row_index = -1
12 min_row_index = -1
13
14 for i in range(m):
15     row_sum = sum(matrix[i])
16     if row_sum > max_sum:
17         max_sum = row_sum
18         max_row_index = i
19     if row_sum < min_sum:

```

```

Run prac81
C:\Users\user\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\user\PyCharmMiscProject\prac81.py
Введите количество строк: 3
Введите количество столбцов: 4
Введите 4 элементов 1-й строки через пробел: 3 4 5 6
Введите 4 элементов 2-й строки через пробел: 7 4 5 1
Введите 4 элементов 3-й строки через пробел: 7 3 9 2
Строка с наибольшей суммой элементов:
Строка 3: [7, 3, 9, 2], Сумма = 21

Строка с наименьшей суммой элементов:
Строка 2: [7, 4, 5, 1], Сумма = 17

Process finished with exit code 0

```

Код программы 2:

```

N = int(input("\nВведите размер квадратной матрицы N: "))

A = []

for i in range(N):

```

```
row = list(map(int, input(f"Введите {N} элементов {i+1}-й строки через  
пробел: ").split()))  
A.append(row)
```

```
for i in range(N):  
    for j in range(N):  
        if A[i][j] < 0:  
            A[i][j] = 0  
        elif A[i][j] > 0:  
            A[i][j] = 1
```

```
print("\nНижняя треугольная матрица:")  
for i in range(N):  
    for j in range(i + 1):  
        print(A[i][j], end=' ')  
    print()
```

Скриншот

Project

PyCharmMiscProject C:\Users\user

.py prac3.py prac4.py prac5.py prac6.py prac71.py prac72.py prac81.py экфс82.py

PyCharmMiscProject C:\Users\user

.venv library root

prac1.py
prac2.py
prac3.py
prac4.py
prac5.py
prac6.py
prac71.py
prac72.py
prac81.py
экфс82.py

External Libraries

Scratches and Consoles

Run

экфс82

```
N = int(input("\nВведите размер квадратной матрицы N: "))
A = []
for i in range(N):
    row = list(map(int, input(f"Введите {N} элементов {i+1}-й строки через пробел: ").split()))
    A.append(row)

for i in range(N):
    for j in range(N):
        if A[i][j] < 0:
            A[i][j] = 0
        elif A[i][j] > 0:
            A[i][j] = 1

print("\nНижняя треугольная матрица:")
for i in range(N):
    for j in range(i + 1):
        print(A[i][j], end=' ')
    print()
```

C:\Users\user\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\user\PyCharmMiscProject\экфс82.py

Введите размер квадратной матрицы N: 3
Введите 3 элементов 1-й строки через пробел: 2 5 3
Введите 3 элементов 2-й строки через пробел: 7 3 5
Введите 3 элементов 3-й строки через пробел: 9 1 2

Нижняя треугольная матрица:

1
1 1
1 1 1

Process finished with exit code 0

Практическая работа №9. Файлы python

Задание:

Для заданий из практической работы №8 для своего варианта.

Организовать ввод данных (матриц) из файла (имя: ФИО_группа_vvod.txt)

И вывод результатов в файл (имя: ФИО_группа_vivod.txt).

Код программы:

```
with open("DMO_группа_vvod.txt", "r", encoding="utf-8") as file:
```

```
    data = file.readlines()
```

```
# Задание 1: Прямоугольная матрица
```

```
lines = [line.strip() for line in data if line.strip()]
```

```
current_line = 0
```

```
m, n = map(int, lines[current_line].split())
```

```
current_line += 1
```

```
matrix = []
```

```
for _ in range(m):
```

```
    matrix.append(list(map(int, lines[current_line].split())))
```

```
    current_line += 1
```

```
max_sum = float('-inf')
```

```
min_sum = float('inf')
```

```
max_row_index = -1
```

```
min_row_index = -1
```

```
for i in range(m):
```

```
    row_sum = sum(matrix[i])
```

```
    if row_sum > max_sum:
```

```
        max_sum = row_sum
```

```
        max_row_index = i
```

```
    if row_sum < min_sum:
```

```
min_sum = row_sum
```

```
min_row_index = i
```

```
# Задание 2: Квадратная матрица
```

```
N = int(lines[current_line].strip())
```

```
current_line += 1
```

```
A = []
```

```
for _ in range(N):
```

```
    A.append(list(map(int, lines[current_line].split())))
```

```
    current_line += 1
```

```
for i in range(N):
```

```
    for j in range(N):
```

```
        if A[i][j] < 0:
```

```
            A[i][j] = 0
```

```
        elif A[i][j] > 0:
```

```
            A[i][j] = 1
```

```
with open("DMO_группа_vivod.txt", "w", encoding="utf-8") as file:
```

```
    file.write("Результаты:\n\n")
```

```
    file.write("1. Прямоугольная матрица:\n")
```

```
    for row in matrix:
```

```
        file.write(" ".join(map(str, row)) + "\n")
```

```
    file.write(f"\nСтрока с наибольшей суммой (индекс {max_row_index}):  
{matrix[max_row_index]}\n")
```

```
    file.write(f"Сумма элементов: {max_sum}\n")
```

```
    file.write(f"\nСтрока с наименьшей суммой (индекс {min_row_index}):  
{matrix[min_row_index]}\n")
```

```

file.write(f'Сумма элементов: {min_sum}\n')

file.write("\n\n2. Квадратная матрица после преобразования:\n")
for row in A:
    file.write(" ".join(map(str, row)) + "\n")

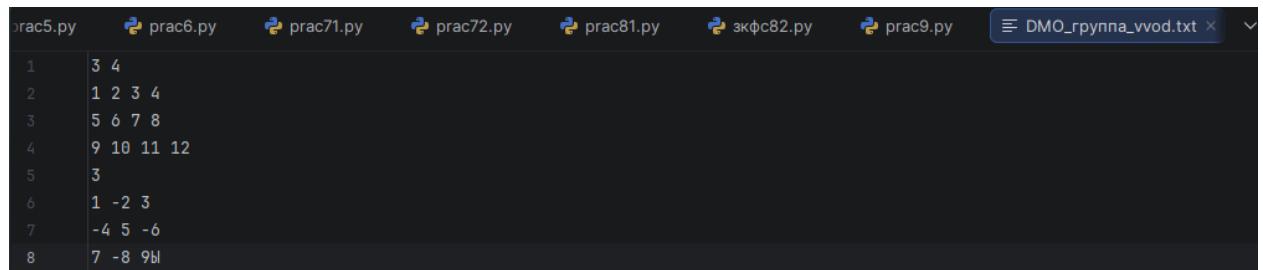
file.write("\nНижняя треугольная матрица:\n")
for i in range(N):
    for j in range(i + 1):
        file.write(f"{A[i][j]} ")
    file.write("\n")

print("Результаты записаны в файл DMO_группа_vivod.txt")

```

Скриншоты

Ввод данных



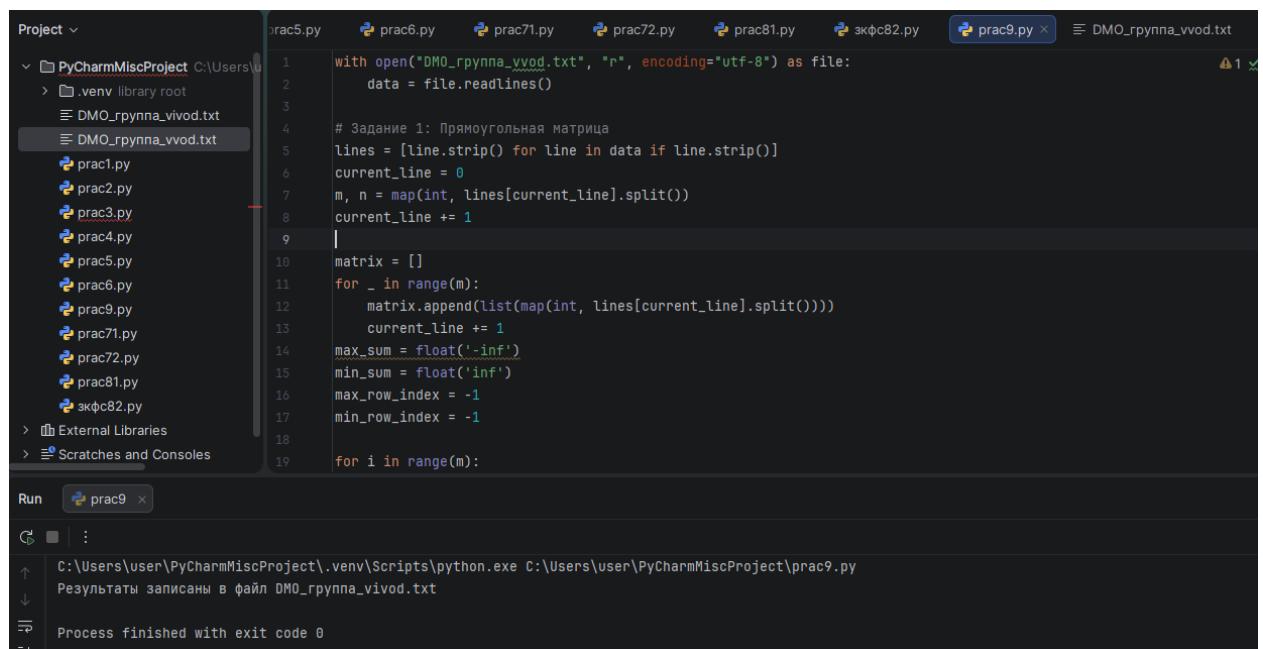
The screenshot shows a PyCharm interface with multiple files listed in the top bar: prac5.py, prac6.py, prac71.py, prac72.py, prac81.py, зкфс82.py, prac9.py, and DMO_группа_vvod.txt. The DMO_группа_vvod.txt file is selected. The code editor displays the following matrix data:

```

1 3 4
2 1 2 3 4
3 5 6 7 8
4 9 10 11 12
5 3
6 1 -2 3
7 -4 5 -6
8 7 -8 9

```

Выполнение программы



The screenshot shows the PyCharm interface during program execution. The left sidebar shows the project structure with files like prac5.py, prac6.py, etc., and the DMO_группа_vvod.txt file is highlighted. The right pane shows the code of prac9.py:

```

with open("DMO_группа_vvod.txt", "r", encoding="utf-8") as file:
    data = file.readlines()

# Задание 1: Прямоугольная матрица
lines = [line.strip() for line in data if line.strip()]
current_line = 0
m, n = map(int, lines[current_line].split())
current_line += 1

matrix = []
for _ in range(m):
    matrix.append(list(map(int, lines[current_line].split())))
    current_line += 1
max_sum = float('-inf')
min_sum = float('inf')
max_row_index = -1
min_row_index = -1
for i in range(m):

```

The bottom status bar indicates the command run: C:\Users\user\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\user\PyCharmMiscProject\prac9.py. The output window shows the results:

```

Результаты записаны в файл DMO_группа_vivod.txt
Process finished with exit code 0

```

Вывод данных

```
1 Результаты:
2
3 1. Прямоугольная матрица:
4 1 2 3 4
5 5 6 7 8
6 9 10 11 12
7
8 Страна с наибольшей суммой (индекс 2): [9, 10, 11, 12]
9 Сумма элементов: 42
10
11 Страна с наименьшей суммой (индекс 0): [1, 2, 3, 4]
12 Сумма элементов: 10
13
14 2. Квадратная матрица после преобразования:
15 1 0 1
16 0 1 0
17 1 0 1
18
19
```

Run `prac9`

External Libraries

Scratches and Consoles

```
C:\Users\user\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\user\PyCharmMiscProject\prac9.py
Результаты записаны в файл DMO_группа_vivod.txt
Process finished with exit code 0
```

Практическая работа №10. Создание GUI

Задание:

Создать приложение с графическим интерфейсом со следующими параметрами:

1. Название приложения: ФИО автора.
2. Создать в приложении 3 вкладки с отступом друг от друга (равномерно распределить по окну)
 - Первая вкладка: простой калькулятор для двух чисел (между ними выпадающий список + - * /).
 - Вторая вкладка: сделать три чекбокса
 Первый
 Второй
 Третий
 И кнопку, при нажатии на нее в зависимости от выбора, выводить всплывающее окно с надписью (например: *вы выбрали первый вариант*).
 - Третья вкладка: работа с текстом, можно текст загрузить файла по кнопке из созданного вами меню.

Код программы:

```
import tkinter as tk
from tkinter import ttk, messagebox, filedialog

class Application(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title("Кривов Дмитрий Евгеньевич")
        self.geometry("500x400")

        self.create_menu()

        self.notebook = ttk.Notebook(self)
        self.notebook.pack(fill='both', expand=True, padx=10, pady=10)

        # Вкладка 1: Калькулятор
        self.create_calculator_tab()

        # Вкладка 2: Чекбоксы
        self.create_checkboxes_tab()

        # Вкладка 3: Работа с текстом
        self.create_text_tab()

    def create_menu(self):
```

```

"""Создание меню для загрузки файла"""
menubar = tk.Menu(self)
self.config(menu=menubar)

file_menu = tk.Menu(menubar, tearoff=0)
menubar.add_cascade(label="Файл", menu=file_menu)
file_menu.add_command(label="Загрузить текст",
command=self.load_file)

def create_calculator_tab(self):
    """Первая вкладка - калькулятор"""
    tab1 = ttk.Frame(self.notebook)
    self.notebook.add(tab1, text="Калькулятор")

    # Поле ввода первого числа
    tk.Label(tab1, text="Первое число:").grid(row=0, column=0, padx=10,
pady=10)
    self.num1_entry = tk.Entry(tab1)
    self.num1_entry.grid(row=0, column=1, padx=10, pady=10)

    # Выпадающий список операций
    self.operation_var = tk.StringVar(value="+")
    operations = ["+", "-", "*", "/"]
    tk.Label(tab1, text="Операция:").grid(row=1, column=0, padx=10,
pady=10)
    operation_menu = ttk.Combobox(tab1, textvariable=self.operation_var,
values=operations, state="readonly")
    operation_menu.grid(row=1, column=1, padx=10, pady=10)

    # Поле ввода второго числа
    tk.Label(tab1, text="Второе число:").grid(row=2, column=0, padx=10,
pady=10)
    self.num2_entry = tk.Entry(tab1)
    self.num2_entry.grid(row=2, column=1, padx=10, pady=10)

    # Кнопка вычисления
    calc_button = tk.Button(tab1, text="Вычислить",
command=self.calculate)
    calc_button.grid(row=3, column=0, columnspan=2, pady=20)

    # Поле для результата
    self.result_label = tk.Label(tab1, text="Результат: ")
    self.result_label.grid(row=4, column=0, columnspan=2, pady=10)

def calculate(self):
    """Выполнение операции калькулятора"""
    try:
        num1 = float(self.num1_entry.get())
        num2 = float(self.num2_entry.get())
        operation = self.operation_var.get()

        if operation == "+":
            result = num1 + num2
        elif operation == "-":
            result = num1 - num2
        elif operation == "*":
            result = num1 * num2
        elif operation == "/":
            if num2 == 0:
                messagebox.showerror("Ошибка", "Деление на ноль!")
                return
            result = num1 / num2

        self.result_label.config(text=f"Результат: {result}")

    except ValueError:
        messagebox.showerror("Ошибка", "Неверный формат числа")

```

```

except ValueError:
    messagebox.showerror("Ошибка", "Введите корректные числа!")

def create_checkboxes_tab(self):
    """Вторая вкладка - чекбоксы"""
    tab2 = ttk.Frame(self.notebook)
    self.notebook.add(tab2, text="Выбор варианта")

    # Переменные для чекбоксов
    self.var1 = tk.BooleanVar()
    self.var2 = tk.BooleanVar()
    self.var3 = tk.BooleanVar()

    # Создаем чекбоксы
    cb1 = tk.Checkbutton(tab2, text="Первый вариант", variable=self.var1)
    cb1.pack(pady=10)

    cb2 = tk.Checkbutton(tab2, text="Второй вариант", variable=self.var2)
    cb2.pack(pady=10)

    cb3 = tk.Checkbutton(tab2, text="Третий вариант", variable=self.var3)
    cb3.pack(pady=10)

    # Кнопка показа выбора
    show_button = tk.Button(tab2, text="Показать выбор",
                           command=self.show_selection)
    show_button.pack(pady=20)

def show_selection(self):
    """Показ выбранных вариантов"""
    selected = []
    if self.var1.get():
        selected.append("Первый вариант")
    if self.var2.get():
        selected.append("Второй вариант")
    if self.var3.get():
        selected.append("Третий вариант")

    if selected:
        message = "Вы выбрали: " + ", ".join(selected)
    else:
        message = "Вы ничего не выбрали"

    messagebox.showinfo("Ваш выбор", message)

def create_text_tab(self):
    """Третья вкладка - работа с текстом"""
    tab3 = ttk.Frame(self.notebook)
    self.notebook.add(tab3, text="Текст")

    # Текстовое поле с прокруткой
    text_frame = tk.Frame(tab3)
    text_frame.pack(fill='both', expand=True, padx=10, pady=10)

    scrollbar = tk.Scrollbar(text_frame)
    scrollbar.pack(side='right', fill='y')

    self.text_widget = tk.Text(text_frame, height=15,
                               yscrollcommand=scrollbar.set)
    self.text_widget.pack(side='left', fill='both', expand=True)

    scrollbar.config(command=self.text_widget.yview)

    # Информационная метка

```

```
    info_label = tk.Label(tab3, text="Используйте меню 'Файл → Загрузить текст' для загрузки файла")
    info_label.pack(pady=10)

def load_file(self):
    """Загрузка текста из файла"""
    file_path = filedialog.askopenfilename(
        title="Выберите текстовый файл",
        filetypes=[("Текстовые файлы", "*.txt"), ("Все файлы", "*.*")]
    )

    if file_path:
        try:
            with open(file_path, 'r', encoding='utf-8') as file:
                content = file.read()

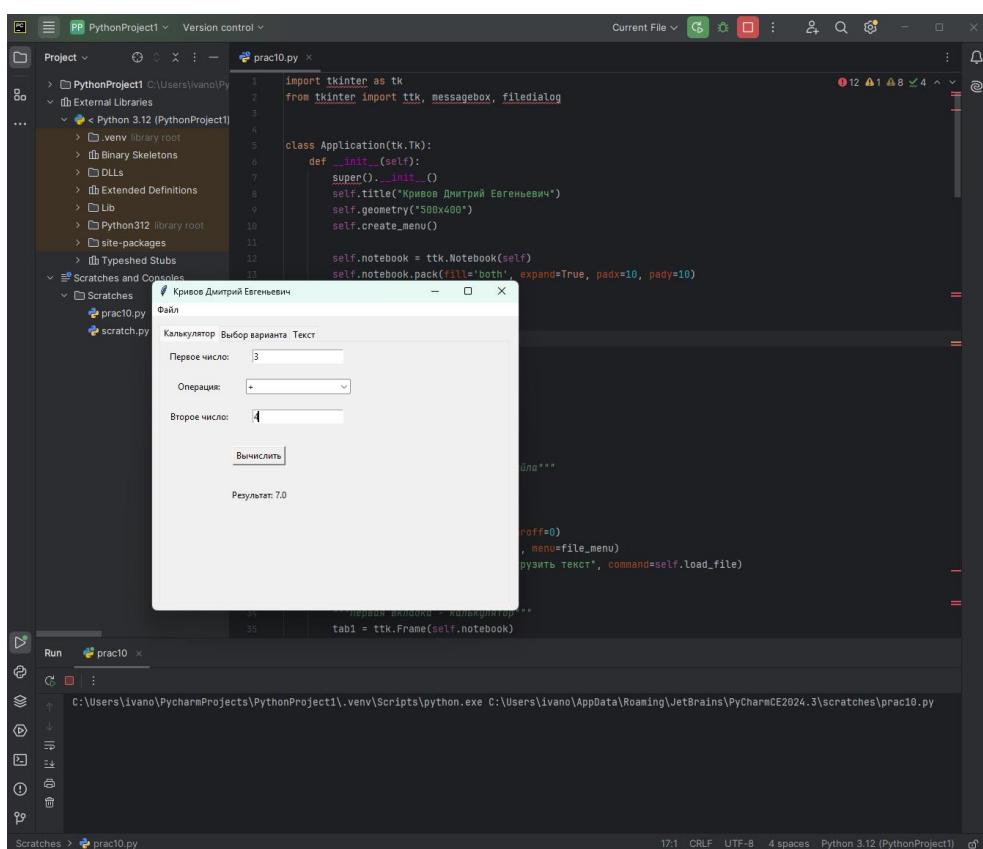
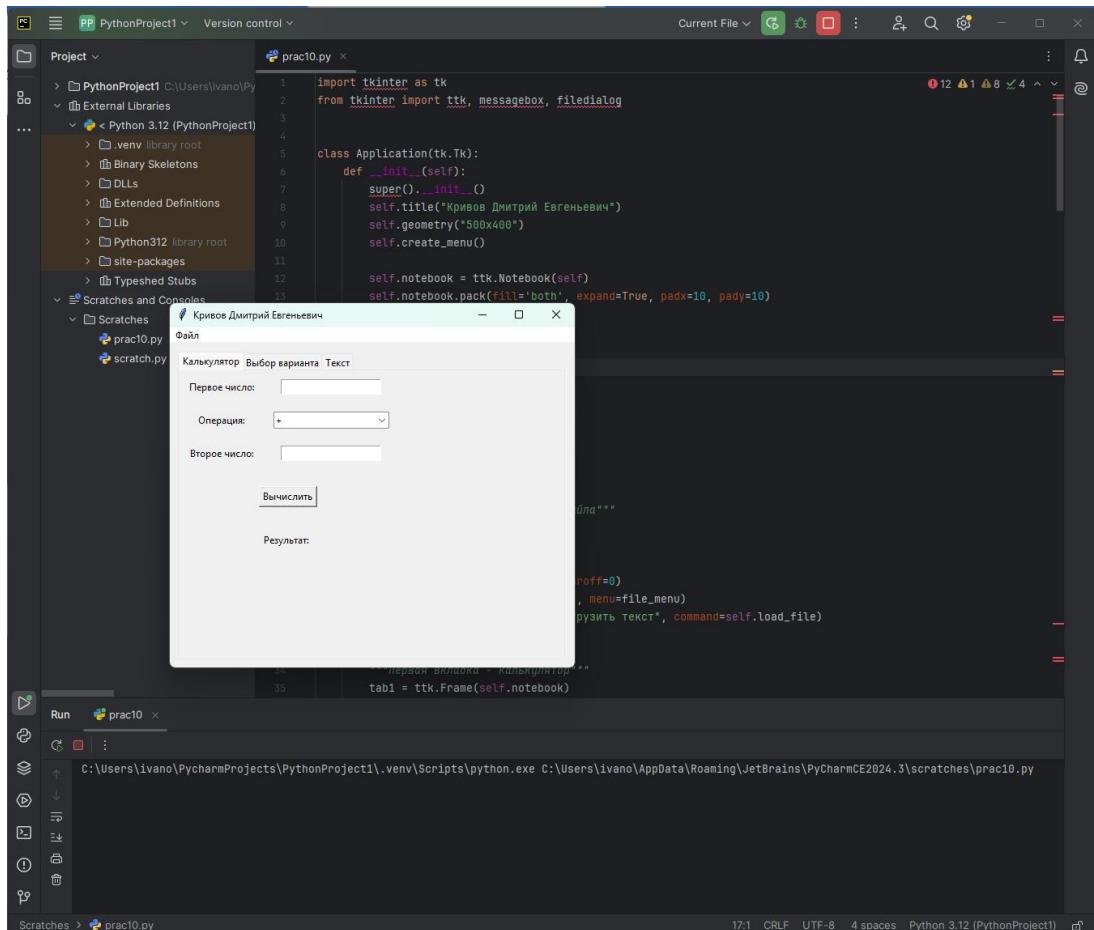
            # Очищаем текстовое поле и вставляем новый текст
            self.text_widget.delete(1.0, tk.END)
            self.text_widget.insert(1.0, content)

            # Переключаемся на третью вкладку
            self.notebook.select(2)

        except Exception as e:
            messagebox.showerror("Ошибка", f"Не удалось загрузить файл:\n{str(e)}")

if __name__ == "__main__":
    app = Application()
    app.mainloop()
```

Скриншоты:



The screenshot shows the PyCharm IDE interface with a project named "PythonProject1". The code editor displays "prac10.py" with the following content:

```
1 import tkinter as tk
2 from tkinter import ttk, messagebox, filedialog
3
4 class Application(tk.Tk):
5     def __init__(self):
6         super().__init__()
7         self.title("Кривов Дмитрий Евгеньевич")
8         self.geometry("500x400")
9         self.create_menu()
10
11     self.notebook = ttk.Notebook(self)
12     self.notebook.pack(fill='both', expand=True, padx=10, pady=10)
```

A secondary window titled "Калькулятор Выбор варианта Текст" is open, showing three radio buttons:

- Первый вариант
- Второй вариант
- Третий вариант

A modal dialog titled "Ваш выбор" displays the message: "Вы выбрали: Первый вариант" with an "OK" button.

The status bar at the bottom right shows: 17:1 CRLF UTF-8 4 spaces Python 3.12 (PythonProject1).

The screenshot shows the PyCharm IDE interface with a Python project named "PythonProject1". The project structure on the left includes "External Libraries" and "Scratches and Consoles". In the "Scratches" section, there are files "prac10.py" and "scratch.py". The main editor window displays the code for "prac10.py". A small application window titled "Калькулятор Выбор варианта Текст" is overlaid on the PyCharm interface. This window contains text about network setup and file transfer instructions. The status bar at the bottom shows the path "C:\Users\ivano\PycharmProjects\PythonProject1\.venv\Scripts\python.exe C:\Users\ivano\AppData\Roaming\JetBrains\PyCharmCE2024.3\scratches\prac10.py".

```
import tkinter as tk
from tkinter import ttk, messagebox, filedialog

class Application(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title("Криков Дмитрий Евгеньевич")
        self.geometry("500x400")
        self.create_menu()
        self.notebook = ttk.Notebook(self)
        self.notebook.pack(fill='both', expand=True, padx=10, pady=10)

    def create_menu(self):
        menu_bar = tk.Menu(self)
        file_menu = tk.Menu(menu_bar, tearoff=0)
        file_menu.add_command(label="Выход", command=self.quit)
        menu_bar.add_cascade(label="Файл", menu=file_menu)
        self.config(menu=menu_bar)
```

Практическая работа № 11. Получение данных пользователя

Задание:

Задание: даны самые популярные репозитории на github

<https://habr.com/ru/post/453444/>, по последней цифре зачетки получить JSON для вашего варианта .

Программа с графическим интерфейсом вводим в поле имя репозитория и по нажатию кнопки получаем результат.

Необходимо получить в новый файл следующую информацию:

```
'company': None,  
'created_at': '2015-08-03T17:55:43Z',  
'email': None,  
'id': 13629408,  
'name': 'Kubernetes',  
'url': 'https://api.github.com/users/kubernetes'}
```

Все прикрепить одним архивом.

Код программы:

```
import tkinter as tk  
from tkinter import ttk, messagebox  
import requests  
import json  
import os  
  
class GitHubRepoApp:  
    def __init__(self, root):  
        self.root = root  
        self.root.title("GitHub Repository Info")  
        self.root.geometry("500x400")  
  
        # Создаем основной интерфейс  
        self.create_widgets()  
  
    def create_widgets(self):  
        # Заголовок  
        title_label = tk.Label(  
            self.root,  
            text="GitHub Repository Information",  
            font=("Arial", 16, "bold"))  
        title_label.pack(pady=20)  
  
        # Метка и поле ввода для имени репозитория  
        input_frame = tk.Frame(self.root)  
        input_frame.pack(pady=20)  
  
        repo_label = tk.Label(  
            input_frame,  
            text="Repository Name (owner/repo):",  
            font=("Arial", 10))  
        repo_label.pack(side=tk.LEFT, padx=(0, 10))  
  
        self.repo_entry = tk.Entry(input_frame, width=30, font=("Arial", 10))  
        self.repo_entry.pack(side=tk.LEFT)  
        self.repo_entry.insert(0, "kubernetes/kubernetes")  
  
        # Кнопка для получения информации
```

```

        self.get_info_btn = tk.Button(
            self.root,
            text="Get Repository Info",
            command=self.get_repo_info,
            bg="#2ea44f",
            fg="white",
            font=("Arial", 10, "bold"),
            padx=20,
            pady=10
        )
        self.get_info_btn.pack(pady=20)

        # Область для отображения результата
        result_frame = tk.LabelFrame(self.root, text="Result", padx=10,
                                     pady=10)
        result_frame.pack(padx=20, pady=10, fill=tk.BOTH, expand=True)

        self.result_text = tk.Text(result_frame, height=10, font=("Courier",
         9))
        self.result_text.pack(fill=tk.BOTH, expand=True)

        # Добавляем скроллбар
        scrollbar = tk.Scrollbar(self.result_text)
        scrollbar.pack(side=tk.RIGHT, fill=tk.Y)
        self.result_text.config(yscrollcommand=scrollbar.set)
        scrollbar.config(command=self.result_text.yview)

        # Статус бар
        self.status_bar = tk.Label(
            self.root,
            text="Ready",
            bd=1,
            relief=tk.SUNKEN,
            anchor=tk.W
        )
        self.status_bar.pack(side=tk.BOTTOM, fill=tk.X)

    def get_repo_info(self):
        # Получаем имя репозитория из поля ввода
        repo_name = self.repo_entry.get().strip()

        if not repo_name:
            messagebox.showerror("Error", "Please enter a repository name!")
            return

        self.status_bar.config(text="Fetching data from GitHub...")
        self.root.update()

        try:
            # Формируем URL для API GitHub
            url = f"https://api.github.com/repos/{repo_name}"

            # Делаем запрос к GitHub API
            response = requests.get(url, timeout=10)

            if response.status_code == 200:
                data = response.json()

                # Извлекаем информацию о владельце
                owner_info = data.get('owner', {})

                # Формируем результат в нужном формате
                result_data = {
                    'company': owner_info.get('company'),

```

```

        'created_at': owner_info.get('created_at'),
        'email': owner_info.get('email'),
        'id': owner_info.get('id'),
        'name': owner_info.get('login'),
        'url': owner_info.get('url')
    }

    # Отображаем результат в текстовом поле
    self.result_text.delete(1.0, tk.END)
    formatted_json = json.dumps(result_data, indent=2,
ensure_ascii=False)
    self.result_text.insert(1.0, formatted_json)

    # Сохраняем в файл
    self.save_to_file(result_data, repo_name)

    self.status_bar.config(text="Data fetched and saved
successfully!")

    elif response.status_code == 404:
        messagebox.showerror("Error", "Repository not found!")
        self.status_bar.config(text="Error: Repository not found")
    else:
        messagebox.showerror("Error", f"GitHub API error:
{response.status_code}")
        self.status_bar.config(text=f"Error: GitHub API returned
{response.status_code}")

    except requests.exceptions.RequestException as e:
        messagebox.showerror("Error", f"Network error: {str(e)}")
        self.status_bar.config(text="Error: Network connection failed")
    except Exception as e:
        messagebox.showerror("Error", f"Unexpected error: {str(e)}")
        self.status_bar.config(text="Error: Unexpected error occurred")

def save_to_file(self, data, repo_name):
    # Создаем имя файла на основе имени репозитория
    safe_repo_name = repo_name.replace('/', '_')
    filename = f"github_info_{safe_repo_name}.json"

    # Сохраняем данные в JSON файл
    with open(filename, 'w', encoding='utf-8') as f:
        json.dump(data, f, indent=2, ensure_ascii=False)

    # Показываем сообщение об успешном сохранении
    messagebox.showinfo(
        "Success",
        f"Data has been saved to:\n{os.path.abspath(filename)}"
    )

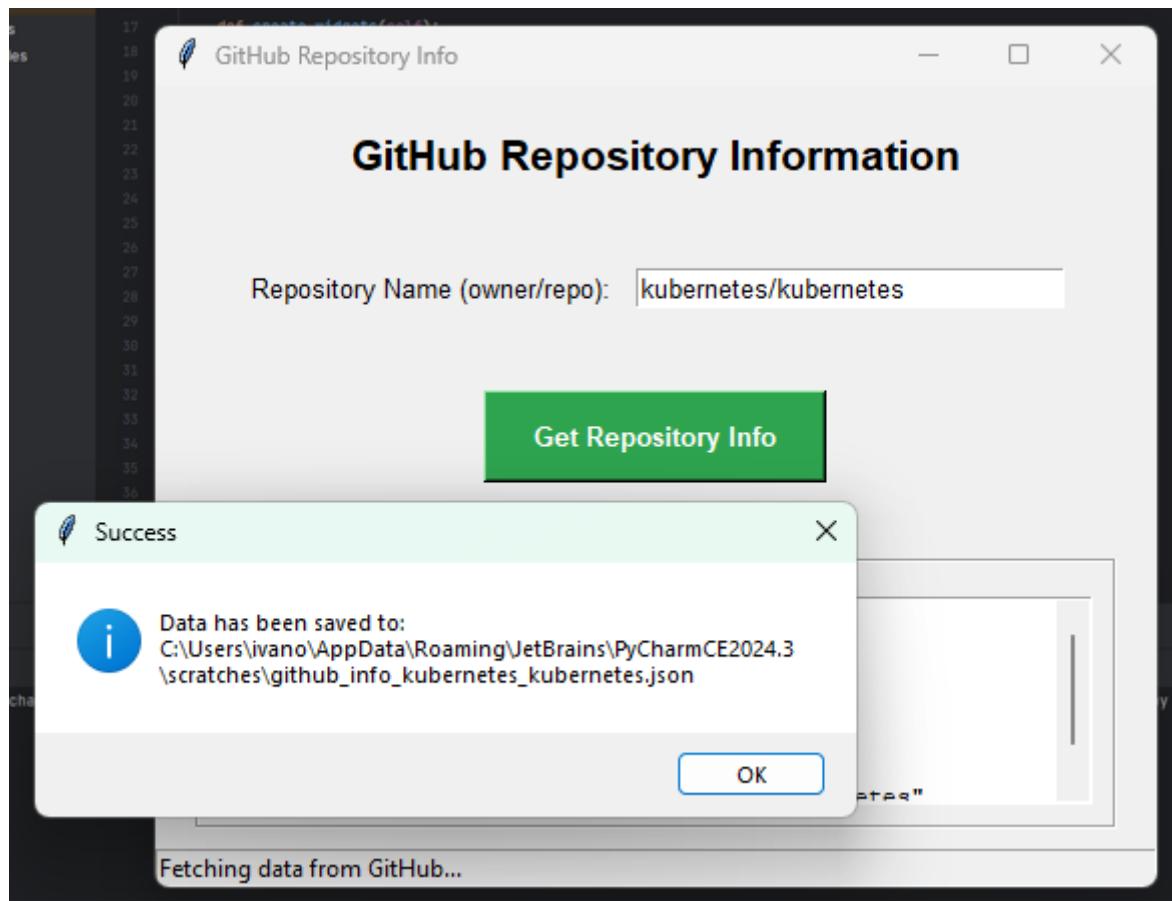
def main():
    root = tk.Tk()
    app = GitHubRepoApp(root)
    root.mainloop()

if __name__ == "__main__":
    main()

```

The screenshot shows the PyCharm IDE interface with the following details:

- Project View:** Shows the project structure under "PythonProject1". The "External Libraries" section is expanded, showing "Python 3.12 (PythonProject1)" which includes "venv library root", "Binary Skeletons", "DLLs", "Extended Definitions", "Lib", "Python312 library root", "site-packages", and "Typeshed Stubs".
- Code Editor:** The file "prac11.py" is open, displaying Python code for a GUI application. The code defines a class `GitHubRepoApp` with methods `__init__` and `create_widgets`. It also includes a function `get_repo_info` and a `Result` container.
- Run Tab:** The "Run" tab is selected, showing the command: `C:\Users\ivano\PycharmProjects\PythonProject1\.venv\Scripts\python.exe C:\Users\ivano\AppData\Roaming\JetBrains\PyCharmCE2024.3\scratches\prac11.py`.
- Scratches:** A modal dialog titled "GitHub Repository Information" is displayed, prompting for a "Repository Name (owner/repo)". The input field contains "kubernetes/kubernetes". Below it is a green "Get Repository Info" button. The "Result" area is empty.
- Status Bar:** The status bar at the bottom shows "Ready", the current time "15:11", file encoding "CRLF", character set "UTF-8", and code style settings "4 spaces" and "Python 3.12 (PythonProject1)".



Практическая работа № 12. Рекурсия

Задание

4. Дано натуральное число N. Вычислите сумму его цифр. При решении этой задачи нельзя использовать строки, списки, массивы
4. Дано натуральное число $n > 1$. Проверьте, является ли оно простым. Программа должна вывести слово YES, если число простое и NO, если число составное. Алгоритм должен иметь сложность $O(\log n)$. Указание. Понятно, что задача сама по себе нерекурсивна, т.к. проверка числа n на простоту никак не сводится к проверке на простоту меньших чисел. Поэтому нужно сделать еще один параметр рекурсии: делитель числа, и именно по этому параметру и делать рекурсию.

Код программы Блок А

```
def sum_digits(n):  
    if n < 10:  
        return n  
    return n % 10 + sum_digits(n // 10)  
n = int(input("Введите натуральное число N: "))  
print(f"Сумма цифр числа {n}: {sum_digits(n)}")
```

Скриншот:

The screenshot shows the PyCharm IDE interface. On the left, there's a project tree for 'PythonProject1' containing files 'prac10.py', 'prac11.py', and 'scratch_1.py'. The 'scratch_1.py' file is open and contains the provided Python code for calculating the sum of digits of a number. In the bottom right pane, the terminal window shows the execution of the script: it prompts for a natural number (54), prints the sum of its digits (54), and then exits with code 0. The terminal path is 'C:\Users\ivano\PycharmProjects\PythonProject1\.venv\Scripts\python.exe'.

Код программы Блок Б

```
def is_prime(n, divisor=2):  
    if n < 2:  
        return False  
    if divisor * divisor > n:  
        return True  
    if n % divisor == 0:  
        return False  
    return is_prime(n, divisor + 1)  
  
# Ввод числа  
n = int(input("Введите натуральное число n > 1: "))
```

```
if n > 1:
    result = "YES" if is_prime(n) else "NO"
    print(f"Число {n} простое? {result}")
else:
    print("Число должно быть больше 1")
```

Скриншот:

The screenshot shows the PyCharm IDE interface. On the left is the Project tool window displaying a file tree for 'PythonProject1'. The 'Scratches' section contains files 'prac10.py', 'prac11.py', and 'scratch_2.py'. The 'scratch_2.py' file is open in the main editor area, showing the provided Python code. Below the editor is the Run tool window, which shows the output of running 'scratch_2.py'. The terminal output is:
C:\Users\ivano\PycharmProjects\PythonProject1\.venv\Scripts\python.exe C:\Users\ivano\AppData\Roaming\JetBrains\PyCharm
Введите натуральное число n > 1: 32
Число 32 простое? Нет
Process finished with exit code 0