МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

КАТЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Дискретна математика

Лабораторна робота №1

«Множини: основні властивості та операції над ними, діаграми Венна»

Виконав:
студент групи ІО-32
Крадожон М. Р.
Номер у списку групи: 16
Перевірив:
Пономаренко А. М.

Київ – 2023

Лабораторна робота №1

Тема: «Множини: основні властивості та операції над ними, діаграми Венна».

Мета: вивчити основні аксіоми, закони і теореми теорії множин, навчитися застосовувати їх на практиці. Обчислити логічний вираз шляхом послідовного застосування операцій над множинами.

Загальне завдання:

1. Повторити матеріал: «Бібліотека tkinter (віджети)» та виконати лабораторну роботу з застосуванням графічного інтерфейсу.
2. Спростити логічний вираз з застосуванням тотожностей алгебри множин.
3. В окремому модулі написати функцію обчислення початкового логічного виразу (1), вибраного відповідно до індивідуального варіанта.
4. В окремому модулі написати функцію обчислення спрощеного логічного виразу.
5. В окремому модулі написати функцію виконання логічної операції (2), вибраної відповідно до індивідуального варіанта.
6. В окремому модулі виконати порівняння результатів:
    А) обчислення початкового та спрощеного виразу
    Б) виконання логічної операції Вашою функцією та відповідною стандартною логічною операцією або функцією Python.

Теоретичні основи:

1.1 Властивості множин

Множина – є сукупність визначених об'єктів, різних між собою, об’єднаних за певною ознакою чи властивістю.

Множини позначають великими латинськими буквами. Об'єкти, що складають множини, називають елементами і позначають малими буквами латинського алфавіту.

Скінченна множина – це така множина, кількість елементів якої може бути виражена скінченним числом, причому не важливо, чи можемо ми порахувати це число в даний момент.

Нескінченна множина – це така множина, що не є скінченною.

Способи задавання множин:
- перерахуванням, тобто списком всіх елементів. Такий спосіб задавання прийнятний тільки при задаванні скінченних множин. Позначення списку – у

фігурних дужках. Наприклад, множина, що з перших п'яти простих чисел A = {2,3,5,7,11}. Множина спортсменів університетської хокейної команди: {} B= Іванов, Петров, Сидоров, Бубликов, Сироєжкін, Волосюк; - процедурою, що породжує і описує спосіб одержання елементів множини із уже отриманих елементів або з інших об'єктів.

Якщо A= х студентки групи ММ21, то Іванова ∈ A, а Петров ∉ A. Підмножина. Множину A називають підмножиною (або включенням) множини B (A ⊆ B), якщо кожен елемент множини A є елементом множини B, тобто, якщо x∈A, то x∈B. Якщо A⊆B й A≠B, то A називають строгою підмножиною й позначають A ⊂ B.

Рівність множин. Дві множини рівні (A = B), якщо всі їхні елементи збігаються. Множини A і B рівні,якщо A⊆B і B⊆A.

Потужність множини. Кількість елементів у скінченній множині A називають потужністю множини A і позначають A.

Універсальна множина U є множина, що має таку властивість, що всі розглянуті множини є її підмножинами.

Булеан. Множину всіх підмножин, що складаються з елементів множини A, називають булеаном P(A).

1.2. Операції над множинами

Об’єднання. Об'єднанням множин A і B називають множину, що складається із всіх тих елементів, які належать хоча б одній з множин A або B . Об'єднання множин A і B позначають A ∪ B . Це визначення рівносильне наступному: A ∪ B

Перетин. Перетином множин A і B називають множину, що складається із всіх тих елементів, які належать як множині A , так і множині B . Перетин множин A і B позначають A ∩ B . Це визначення рівносильне наступному: A ∩ B

Доповнення. Доповненням (або абсолютним доповненням) множини A називають множину, що складається із всіх елементів універсальної множини, які не належать A . Доповнення множини A позначають A . Це визначення рівносильне наступному: A=U−A={x∈U и x∉A}.

Різниця. Різницею множин A й B (або відносним доповненням) називають множину, що складається із всіх елементів множини A, які не належать B. Різницю множин A і B позначають A − B або A \ B . Це визначення рівносильне наступному: A−B={x∈A и x∉B}.

Симетрична різниця. Симетричною різницею множин A і B називають множину, що складається з об'єднання всіх елементів, що належать множині A і не містяться в B, і елементів, що належать множині B і не містяться в A. Симетричну різницю множин A і B позначають A + B або A Δ B . Це визначення рівносильне наступному: A Δ B=(A\B)∪(B\A).

Види операцій. Операції, які виконують над однією множиною, називають унарними. Операції, які виконують над двома множинами, називають бінарними. Прикладом унарної операції є знаходження доповнення. Прикладами бінарних операцій є об'єднання, перетин, різниця, симетрична різниця.

1.3. Діаграми Венна

Для графічної ілюстрації операцій над множинами даної універсальної множини U використовують діаграми Венна. Діаграма Венна – це зображення множини у вигляді геометричної множини, наприклад, кола. При цьому універсальну множину зображують у вигляді прямокутника. На рис.1.2 зображені діаграми Венна для розглянутих операцій над множинами.

Для будь-яких підмножин A, B, C універсальної множини U справедливо наступне:

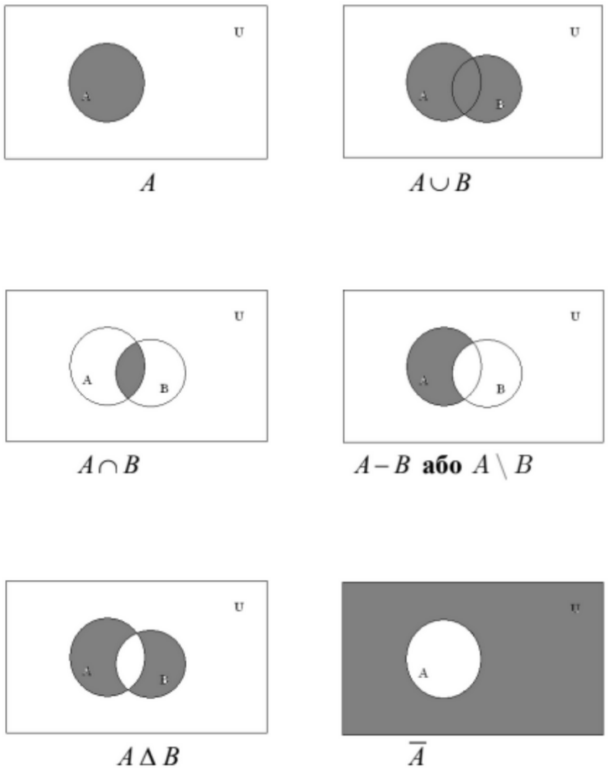


Рис. 1.2.

1.4. Тотожності алгебри множин

1. Комутативність об'єднання $A \cup B = B \cup A$	1. Комутативність перетину $A \cap B = B \cap A$
2. Асоціативність об'єднання $A \cup (B \cup C) = (A \cup B) \cup C$	2. Асоціативність перетину $A \cap (B \cap C) = (A \cap B) \cap C$
3. Дистрибутивність об'єднання відносно перетину $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$	3. Дистрибутивність перетину відносно об'єднання $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
4. Закони дій з пустою та універсальною множинами $A \cup \varnothing = A$ $A \cup \overline{A} = U$ $A \cup U = U$	4. Закони дій з пустою та універсальною множинами $A \cap U = A$ $A \cap \overline{A} = \varnothing$ $A \cap \varnothing = \varnothing$
5.Закон ідемпотентності об'єднання Термін **ідемпотентність** означує властивість математичного об'єкта, яка проявляється в тому, що повторна дія над об'єктом *не змінює* його $A \cup A = A$	5. Закон ідемпотентності перетину $A \cap A = A$
6. Закон де Моргана $\overline{A \cup B} = \overline{A} \cap \overline{B}$	6. Закон де Моргана $\overline{A \cap B} = \overline{A} \cup \overline{B}$
7. Закон поглинання $A \cup (A \cap B) = A$	7. Закон поглинання $A \cap (A \cup B) = A$
8. Закон склеювання $(A \cap B) \cup (A \cap \overline{B}) = A$	8. Закон склеювання $(A \cup B) \cap (A \cup \overline{B}) = A$
9. Закон Порецького $A \cup (\overline{A} \cap B) = A \cup B$	9. Закон Порецького $A \cap (\overline{A} \cup B) = A \cap B$
10. Закон подвійного доповнення $\overline{\overline{A}} = A$	
11. Визначення операції «різниця»: $A \setminus B = A \cap \overline{B}$.	
12. Визначення операції «симетрична різниця»: $A \Delta B = (A \setminus B) \cup (B \setminus A)$.	

Індивідуальне завдання

Загальний порядок виконання лабораторної роботи:

А) Вибрати номер Z індивідуального варіанта відповідно до виразу:
`Z=(i+G%60)%30+1`, де i – номер у списку групи, G – числова складова назви групи.

Програма обчислення варіанта:

```
G=64
N=1
print("Моя група: IO -",G)
print("Мій номер у групі:",N)
print("Мій варіант:",(N+G%60)%30+1)
```

Б) Максимально спростити логічний вираз. Для спрощення використати тотожності алгебри множин та визначення логічних операцій. Спрощення є максимальним, якщо формула містить тільки одне входження кожної множини.

В) Створити блок-схему послідовності обчислення початкового логічного виразу.

Г) Створити блок-схему послідовності обчислення спрощеного логічного виразу.

Ґ) З використанням блок-схем створити проект, який містить модуль з функцією для обчислення початкового виразу (1) та модуль з функцією для обчислення спрощеного виразу.

Д) З використанням блок-схем, заданих у лабораторній роботі, створити модуль з функцією для виконання логічної операції (2), вибраної відповідно до варіанта.

Е) В основному файлі виконати перевірку правильності спрощення виразу з виводом відповідного повідомлення.

Є) Як елементи множин можуть бути використані числа від 0 до 255.

Ж) Лабораторну роботу виконувати з застосуванням мови Python та бібліотеки tkinter.

Вимоги до інтерфейсу

А) Програма повинна складатися з 5-ти вікон:

Вміст вікна No1

1. Головне меню, яке повинно включати виклик вікна No2, вікна No3, вікна No4 та вікна No5.

2. Віджети виводу П.І.Б студента, номера групи, номера у списку та віджет виводу результатів обчислення варіанту відповідно да програми, що задана у завданні – пункт (А) загального порядку виконання лабораторної роботи).

3. Віджети для задавання потужності множин A, B і C.

4. Віджети для формування випадковим чином множин A,B і C заданої потужності.

5. Віджети, що дають можливість ручного вводу множин A,B і C.

6. Віджет для задавання діапазону цілих чисел, які будемо вважати універсальною множиною.

Вміст вікна No2

1. Віджети для відображення елементів множин A, B і C.

2. Віджети запуску покрокового виконання початкового виразу (1). Одним кроком вважати виконання однієї логічної операції.

3. Віджети відображення множин-операндів та множини-результату кожної логічної операції.

4. Віджет відображення множини D та віджет для виконання команди збереження даного результату у файлі.

Вміст вікна No3

1. Віджети для відображення елементів множин A, B і C.

2. Віджети запуску покрокового виконання спрощеного логічного виразу. Одним кроком вважати виконання однієї логічної операції.

3. Віджети відображення множин-операндів та множини-результату кожної логічної операції.

4. Віджет відображення множини D, та віджет для виконання команди збереження даного результату у файлі.

Вміст вікна No4

1. Віджети для відображення елементів множин X і Y.

2. Віджет відображення результату Z виконання заданої у індивідуальному варіанті логічної операції (2) над множинами X і Y, яка виконана за допомогою написаної вами функції.

Вміст вікна No5

1. Віджети для зчитування з файлу та відображення множини, яка є результатом обчислення початкового виразу.

2. Віджети для зчитування з файлу та відображення множини, яка є результатом обчислення спрощеного виразу.

3. Віджети для зчитування з файлу та відображення множини Z, яка є результатом обчислення операції над множинами X і Y, яка виконана за допомогою написаної вами функції.

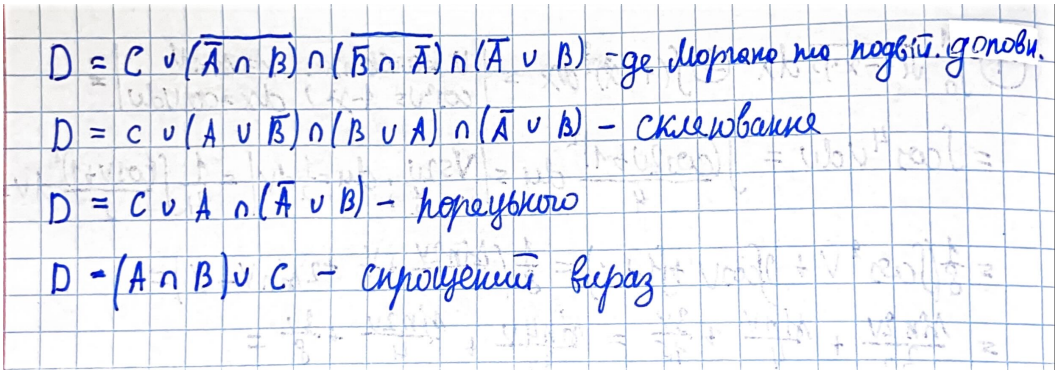
4. Віджети для запуску та відображення логічної операції над множинами X і Y, яка є результатом обчислення Z з використанням стандартної функції Python.

5. Віджети для порівняння результатів виводу у пунктах 1 та 2 та у пунктах 3 та 4.

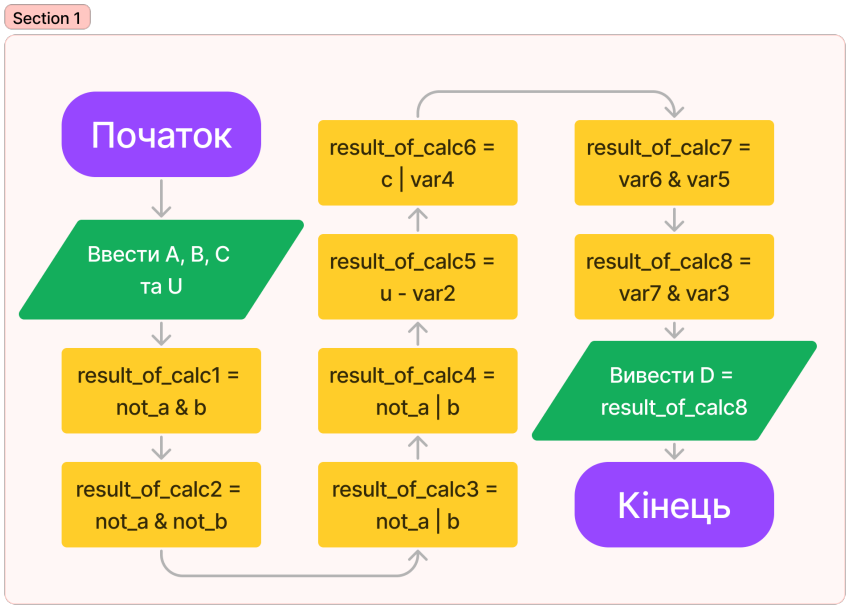
Варіант:

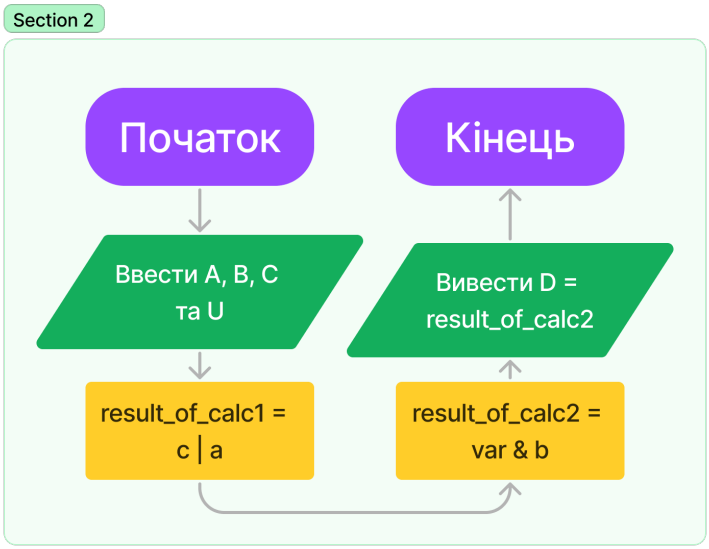
19	(1)	$D = C \cup \overline{(\bar{A} \cap B)} \cap \overline{(\bar{B} \cap \bar{A})} \cap (\bar{A} \cup B)$
	(2)	$X = C; Y = \bar{A}; Z = X \cup Y$

Спрощення виразу:



Блок-схеми:





Роздруківка коду:

Файл із функціями:

```
# Обчислення другої логічної операції за допомогою вбудованих функцій Python
def calc_union(not_a, c):  # Знаходить об'єднання комплемента множини A та множини C
    return not_a | c


# Обчислення другої логічної операції власною функцією
def custom_calc_union(not_a, c):  # Знаходить об'єднання комплемента множини A та множини C власною функцією
    for i in c:
        if i not in not_a:
            not_a.add(i)
    return not_a


# Обчислення даного виразу
def step1(a, b, u):  # Обчислення першого кроку виразу: ¬A ∩ B
    not_a = u - a
    result_of_calc = not_a & b
    return result_of_calc


def step2(a, b, u):  # Обчислення другого кроку виразу: ¬B ∩ ¬A
    not_a = u - a
    not_b = u - b
    result_of_calc = not_a & not_b
    return result_of_calc


def step3(a, b, u):  # Обчислення третього кроку виразу: ¬A ∪ B
    not_a = u - a
    result_of_calc = calc_union(not_a, b)
    return result_of_calc


def step4(var, u):  # Обчислення четвертого кроку виразу: ¬(¬A ∩ B)
    result_of_calc = u - var
    return result_of_calc


def step5(var2, u):  # Обчислення п'ятого кроку виразу: ¬(¬B ∩ ¬A)
    result_of_calc = u - var2
    return result_of_calc


def step6(c, var4):  # Обчислення шостого кроку виразу: C ∪ ¬(¬A ∩ B)
    result_of_calc = calc_union(c, var4)
    return result_of_calc


def step7(var6, var5):  # Обчислення сьомого кроку виразу: C ∪ ¬(¬A ∩ B) ∩ ¬(¬B ∩ ¬A)
    result_of_calc = var6 & var5
    return result_of_calc
```

```
def step8(var7, var3):  # Обчислення восьмого кроку виразу: C ∪ ¬(¬A ∩ B) ∩ ¬(¬B ∩ ¬A) ∩ (¬A ∪B)
    result_of_calc = var7 & var3
    return result_of_calc


# Обчислення спрощеного виразу
def first_short_step(a, c):  # Знаходить об'єднання множин C та A
    return calc_union(c, a)


def second_short_step(var, b):  # Обчислення спрощеного виразу: (C ∪A) ∩ B
    return var & b
```

Основний файл:

```
import functions
import random
from tkinter import *


# Функції, які генерують множини
def universal_set():
    u.clear()  # Очищає універсальну множину та генерує нову на основі введення користувача
    global left_border
    global right_border
    try:
        left_border = int(left_universal_data.get())
        right_border = int(right_universal_data.get())
        if left_border <= 0 or right_border <= 0:
            raise ValueError("Введені числа повинні бути додатніми.")

        universal_range = range(left_border, right_border + 1, 1)
        for i in universal_range:
            u.add(i)
        print(u)
    except ValueError as e:
        print("Помилка:", e)


def gen_set_A():
    a.clear()  # Очищає множину A та генерує нову заданого розміру
    try:
        print("Генерується множина A")
        power = int(set_a_data.get())
        if power <= 0:
            raise ValueError("Розмір множини повинен бути додатнім числом.")

        while len(a) != power:
            number = random.randint(left_border, right_border)
            if number not in a:
                a.add(number)
        print(a)
    except ValueError as e:
        print("Помилка:", e)


# Інші подібні функції для генерації множин B та C визначаються аналогічно
def gen_set_B():
    b.clear()
```

```
    try:
        print("Генерується множина B")
        power = int(set_b_data.get())
        if power <= 0:
            raise ValueError("Розмір множини повинен бути додатнім числом.")

        while len(b) != power:
            number = random.randint(left_border, right_border)
            if number not in b:
                b.add(number)
        print(b)
    except ValueError as e:
        print("Помилка:", e)


def gen_set_C():
    c.clear()
    try:
        print("Генерується множина C")
        power = int(set_c_data.get())
        if power <= 0:
            raise ValueError("Розмір множини повинен бути додатнім числом.")

        while len(c) != power:
            number = random.randint(left_border, right_border)
            if number not in c:
                c.add(number)
        print(c)
    except ValueError as e:
        print("Помилка:", e)


# Функції, які дають можливість ручного вводу
def manual_input_set_a():
    a.clear()  # Ручне введення множини A
    try:
        a_pool = manual_data_set_a.get().split(",")
        for i in a_pool:
            num = int(i)
            if num <= 0:
                raise ValueError("Елементи множини повинні бути додатніми числами.")
            a.add(num)
        print("A: ", a)
    except ValueError as e:
        print("Помилка:", e)


# Інші подібні функції для ручного введення множин B та C визначаються аналогічно
def manual_input_set_b():
    b.clear()
    try:
        b_pool = manual_data_set_b.get().split(",")
        for i in b_pool:
            num = int(i)
            if num <= 0:
                raise ValueError("Елементи множини повинні бути додатніми числами.")
            b.add(num)
        print("B: ", b)
    except ValueError as e:
        print("Помилка:", e)
```

```
def manual_input_set_c():
    c.clear()
    try:
        c_pool = manual_data_set_c.get().split(",")
        for i in c_pool:
            num = int(i)
            if num <= 0:
                raise ValueError("Елементи множини повинні бути додатніми числами.")
            c.add(num)
        print("C: ", c)
    except ValueError as e:
        print("Помилка:", e)


# Вікно 2
def second_window():
    def save_txt_file():
        f = open(r"D.txt", "w")
        f.write(str(step_eighth_of_calculation))
        f.close()

    # Включає кроки для обчислення множини D за допомогою визначених функцій
    def step1():
        Label(root2, text=f"¬A ∩ B: {step_first_of_calculation}", font='Arial 12').place(x=10, y=60+10)

    def step2():
        Label(root2, text=f"¬B ∩ ¬A: {step_second_of_calculation}", font='Arial 12').place(x=10, y=80+10)

    def step3():
        Label(root2, text=f"¬A ∪ B: {step_third_of_calculation}", font='Arial 12').place(x=10, y=100+10)

    def step4():
        Label(root2, text=f"¬(¬A ∩ B): {step_fourth_of_calculation}", font='Arial 12').place(x=10, y=120+10)

    def step5():
        Label(root2, text=f"¬(¬B ∩ ¬A): {step_fifth_of_calculation}", font='Arial 12').place(x=10, y=140+10)

    def step6():
        Label(root2, text=f"C ∪ ¬(¬A ∩ B): {step_sixth_of_calculation}", font='Arial 12').place(x=10, y=160+10)

    def step7():
        Label(root2, text=f"C ∪ ¬(¬A ∩ B) ∩ ¬(¬B ∩ ¬A): {step_seventh_of_calculation}",
              font='Arial 12').place(x=10, y=180+10)

    def step8():
        Label(root2, text=f"C ∪ ¬(¬A ∩ B) ∩ ¬(¬B ∩ ¬A) ∩ (¬A ∪ B): {step_eighth_of_calculation}",
              font='Arial 12').place(x=10, y=200+10)
        Label(root2, text=f'Результат D: {step_eighth_of_calculation}', font='Arial 12').place(x=500, y=320)

    step_first_of_calculation = functions.step1(a, b, u)
    step_second_of_calculation = functions.step2(a, b, u)
    step_third_of_calculation = functions.step3(a, b, u)
    step_fourth_of_calculation = functions.step4(step_first_of_calculation, u)
    step_fifth_of_calculation = functions.step5(step_second_of_calculation, u)
    step_sixth_of_calculation = functions.step6(c, step_fourth_of_calculation)
    step_seventh_of_calculation = functions.step7(step_sixth_of_calculation, step_fifth_of_calculation)
    step_eighth_of_calculation = functions.step8(step_seventh_of_calculation, step_third_of_calculation)
```

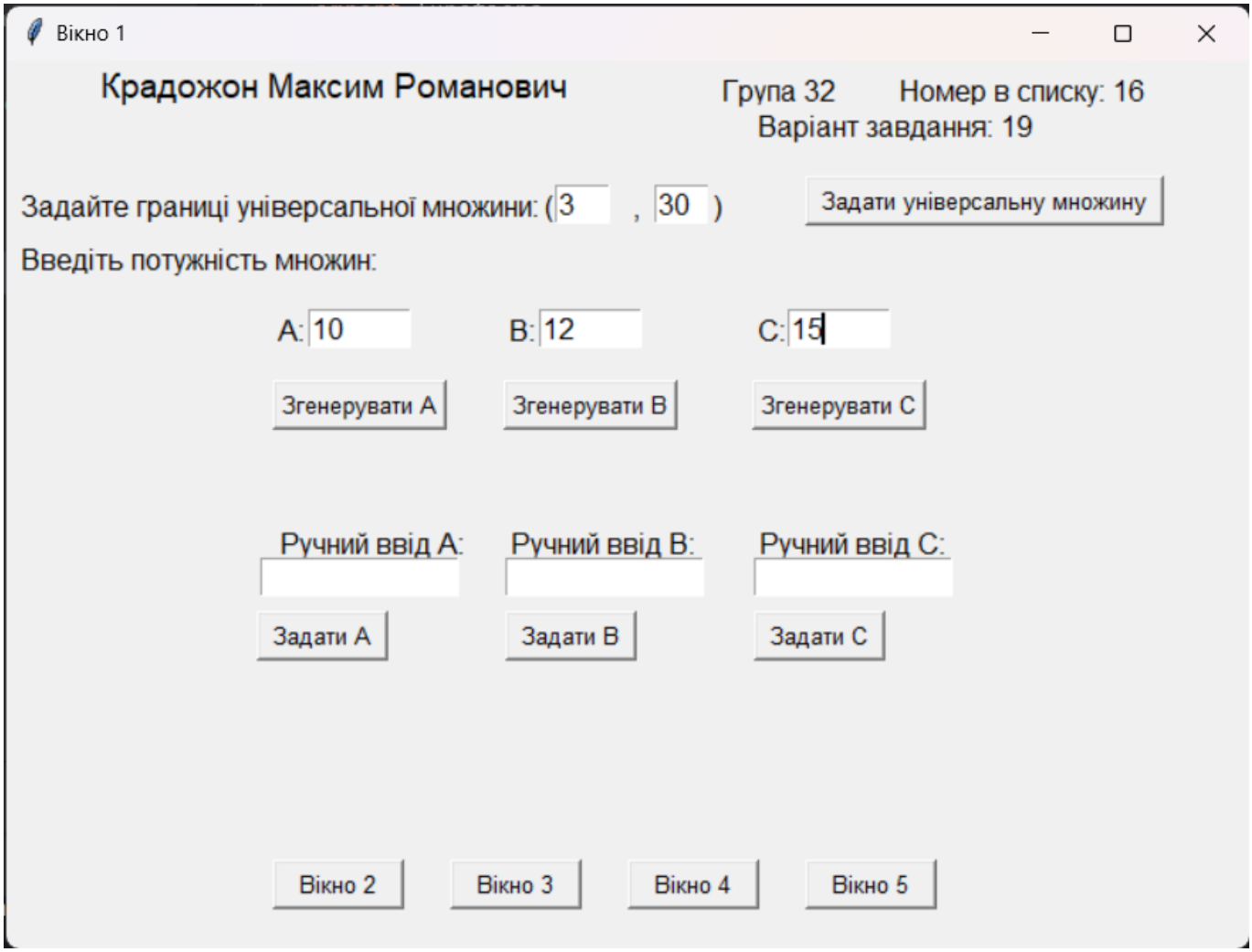
```
    # Відображає результати в вікні GUI
    root2 = Tk()
    root2.title("Вікно 2")
    root2.title("Вікно 2")
    root2.geometry("900x400")
    Label(root2, text=f'A: {a}  ¬A: {u - a}', font='Arial 12').place(x=0)
    Label(root2, text=f'B: {b}  ¬B: {u - b}', font='Arial 12').place(x=0, y=20)
    Label(root2, text=f'C: {c}', font='Arial 12').place(x=0, y=40)
    Button(root2, width=8, text="Крок 1", font="Arial 10", command=step1).place(x=0+30, y=320)
    Button(root2, width=8, text="Крок 2", font="Arial 10", command=step2).place(x=80+30, y=320)
    Button(root2, width=8, text="Крок 3", font="Arial 10", command=step3).place(x=160+30, y=320)
    Button(root2, width=8, text="Крок 4", font="Arial 10", command=step4).place(x=240+30, y=320)
    Button(root2, width=8, text="Крок 5", font="Arial 10", command=step5).place(x=0+30, y=350)
    Button(root2, width=8, text="Крок 6", font="Arial 10", command=step6).place(x=80+30, y=350)
    Button(root2, width=8, text="Крок 7", font="Arial 10", command=step7).place(x=160+30, y=350)
    Button(root2, width=8, text="Крок 8", font="Arial 10", command=step8).place(x=240+30, y=350)
    Button(root2, width=24, text="Завантажити D у файл на ПК", font="Arial 10",
           command=save_txt_file).place(x=500, y=350)


# Вікно 3
# Інші функції вікон (third_window, fourth_window, fifth_window) визначаються аналогічно
def third_window():
    step_first_of_calculation = functions.first_short_step(a, c)
    step_second_of_calculation = functions.second_short_step(step_first_of_calculation, b)

    def save_simplified_txt_file():
        f = open(r"D_simplified.txt", "w")
        f.write(str(step_second_of_calculation))
        f.close()

    def step1():
        Label(root3, text=f"C ∪ A: {step_first_of_calculation}", font='Arial 12').place(x=10, y=60+10)

    def step2():
        Label(root3, text=f"C ∪ A ∩ B: {step_second_of_calculation}", font='Arial 12').place(x=10, y=80+10)
        Label(root3, text=f'Результат D: {step_second_of_calculation}', font='Arial 12').place(x=200, y=180)

    root3 = Tk()
    root3.title("Вікно 3")
    root3.geometry("900x300")
    Label(root3, text=f'A: {a}', font='Arial 12').place(x=0)
    Label(root3, text=f'B: {b}', font='Arial 12').place(x=0, y=20)
    Label(root3, text=f'C: {c}', font='Arial 12').place(x=0, y=40)
    Button(root3, width=8, text="Крок 1", font="Arial 10", command=step1).place(x=0 + 30, y=220)
    Button(root3, width=8, text="Крок 2", font="Arial 10", command=step2).place(x=80 + 30, y=220)
    Button(root3, width=24, text="Завантажити D у файл на ПК", font="Arial 10",
           command=save_simplified_txt_file).place(x=200, y=220)


# Вікно 4
def fourth_window():
    step_result = functions.custom_calc_union(u - a, c)

    def saver3():
        f = open(r"customZ.txt", "w")
        f.write(str(step_result))
        f.close()

    def step():
```

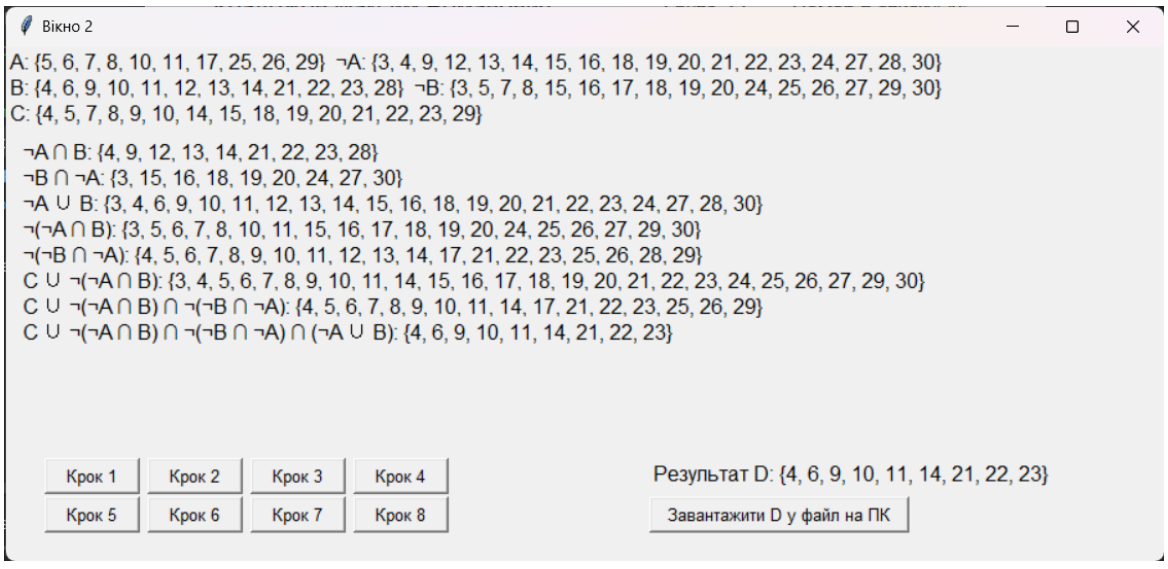
```
        Label(root4, text=f'X ∪ Y: {step_result}', font='Arial 12').place(x=0, y=60)
        Label(root4, text=f'Результат Z: {step_result}', font='Arial 12').place(x=20, y=120)

    root4 = Tk()
    root4.title("Вікно 4")
    root4.geometry("900x300")
    Label(root4, text=f'X: {c}', font='Arial 12').place(x=0)
    Label(root4, text=f'Y: {u - a}', font='Arial 12').place(x=0, y=20)
    Button(root4, width=12, text="Розрахувати", font="Arial 10", command=step).place(x=0 + 30, y=220)
    Button(root4, width=24, text="Завантажити Z у файл на ПК", font="Arial 10", command=saver3).place(x=120 + 30,
y=220)


# Вікно 5
def fifth_window():
    def data_read():
        usual_d_file = open(r"D.txt", "r")
        global data_usual_d_file
        data_usual_d_file = usual_d_file.read()
        usual_d_file.close()
        Label(root5, text=f'D: {data_usual_d_file}', font='Arial 12').place(x=10)

        simple_d_file = open(r"D_simplified.txt", "r")
        global data_simple_d_file
        data_simple_d_file = simple_d_file.read()
        simple_d_file.close()
        Label(root5, text=f'Спрощене D: {data_simple_d_file}', font='Arial 12').place(x=10, y=20)

        z1 = open(r"customZ.txt", "r")
        global z1_data
        z1_data = z1.read()
        z1.close()
        Label(root5, text=f'З використанням функції, яку я сам написав для Z: ', font='Arial 12').place(x=10, y=40)
        Label(root5, text=f'{z1_data}', font='Arial 12').place(x=10, y=60)

    z2_data = str(functions.calc_union(u - a, c))

    def step():
        Label(root5, text=f'Z обчислене функціями Python: ', font='Arial 12').place(x=10, y=80)
        Label(root5, text=f'{z2_data}', font='Arial 12').place(x=10, y=100)

    def compare_d():
        if data_usual_d_file == data_simple_d_file:
            Label(root5, text='Результати D є однаковими', font='Arial 12').place(x=500)
        else:
            Label(root5, text='Результати D є різними', font='Arial 12').place(x=500)

    def compare_z():
        if z1_data == z2_data:
            Label(root5, text='Результати Z є однаковими', font='Arial 12').place(x=500, y=20)
        else:
            Label(root5, text='Результати Z є різними', font='Arial 12').place(x=500, y=20)

    root5 = Tk()
    root5.title("Вікно 5")
    root5.geometry("900x300")
    Button(root5, width=18, text="Зчитати результати", font="Arial 10", command=data_read).place(x=15, y=200)
    Button(root5, width=34, text="Обчислити Z за допомогою функцій Python",
           font="Arial 10", command=step).place(x=15, y=230)
    Button(root5, width=12, text="Порівняти D", font="Arial 10", command=compare_d).place(x=15, y=260)
```

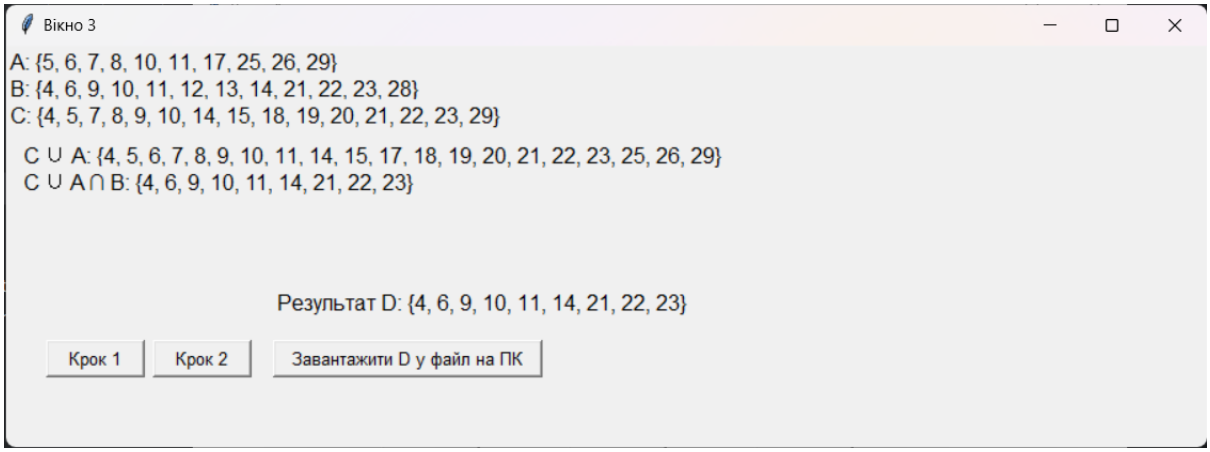
```
    Button(root5, width=12, text="Порівняти Z", font="Arial 10", command=compare_z).place(x=125, y=260)


u = set()
a = set()
b = set()
c = set()

# Інформація про мене
academic_group = 32
number_of_list = 16
variant = (number_of_list + academic_group % 60) % 30 + 1

# Вікно 1
root = Tk()
root.title("Вікно 1")
root.geometry("700x500")

Label(root, text='Крадожон Максим Романович', font='Arial 14').place(x=50)
Label(root, text=f'Група {academic_group}', font='Arial 12').place(x=400, y=5)
Label(root, text=f'Номер в списку: {number_of_list}', font='Arial 12').place(x=500, y=5)
Label(root, text=f'Варіант завдання: {variant}', font='Arial 12').place(x=420, y=25)

Label(root, text='Задайте границі універсальної множини:', font='Arial 12').place(x=5, y=70)  # Універсальна множина
Label(root, text='(', font='Arial 12').place(x=300, y=70)
left_universal_data = Entry(root, width=3, font="Arial 12")
left_universal_data.place(x=309, y=70)
Label(root, text=',', font='Arial 12').place(x=350, y=70)
right_universal_data = Entry(root, width=3, font="Arial 12")
right_universal_data.place(x=365, y=70)
Label(root, text=')', font='Arial 12').place(x=395, y=70)
Button(root, width=24, text="Задати універсальну множину", font="Arial 10", command=universal_set).place(x=450, y=65)

Label(root, text='Введіть потужність множин:', font='Arial 12').place(x=5, y=100)
Label(root, text='A:', font='Arial 12').place(x=10+140, y=140)  # Множина A
set_a_data = Entry(root, width=6, font="Arial 12")
set_a_data.place(x=30+140, y=140)
Button(root, width=11, text="Згенерувати A", font="Arial 10", command=gen_set_A).place(x=10+140, y=180)

Label(root, text='B:', font='Arial 12').place(x=140+140, y=140)  # Множина B
set_b_data = Entry(root, width=6, font="Arial 12")
set_b_data.place(x=160+140, y=140)
Button(root, width=11, text="Згенерувати B", font="Arial 10", command=gen_set_B).place(x=140+140, y=180)

Label(root, text='C:', font='Arial 12').place(x=280+140, y=140)  # Множина C
set_c_data = Entry(root, width=6, font="Arial 12")
set_c_data.place(x=300+140, y=140)
Button(root, width=11, text="Згенерувати C", font="Arial 10", command=gen_set_C).place(x=280+140, y=180)

Label(root, text='Ручний ввід A:', font='Arial 12').place(x=10+141, y=260)  # Ручний ввід A
manual_data_set_a = Entry(root, width=12, font="Arial 12")
manual_data_set_a.place(x=2+141, y=280)
Button(root, width=8, text="Задати A", font="Arial 10", command=manual_input_set_a).place(x=0+141, y=310)

Label(root, text='Ручний ввід B:', font='Arial 12').place(x=140+141, y=260)  # Ручний ввід B
manual_data_set_b = Entry(root, width=12, font="Arial 12")
manual_data_set_b.place(x=140+141, y=280)
Button(root, width=8, text="Задати B", font="Arial 10", command=manual_input_set_b).place(x=140+141, y=310)

Label(root, text='Ручний ввід C:', font='Arial 12').place(x=280+141, y=260)  # Ручний ввід C
```

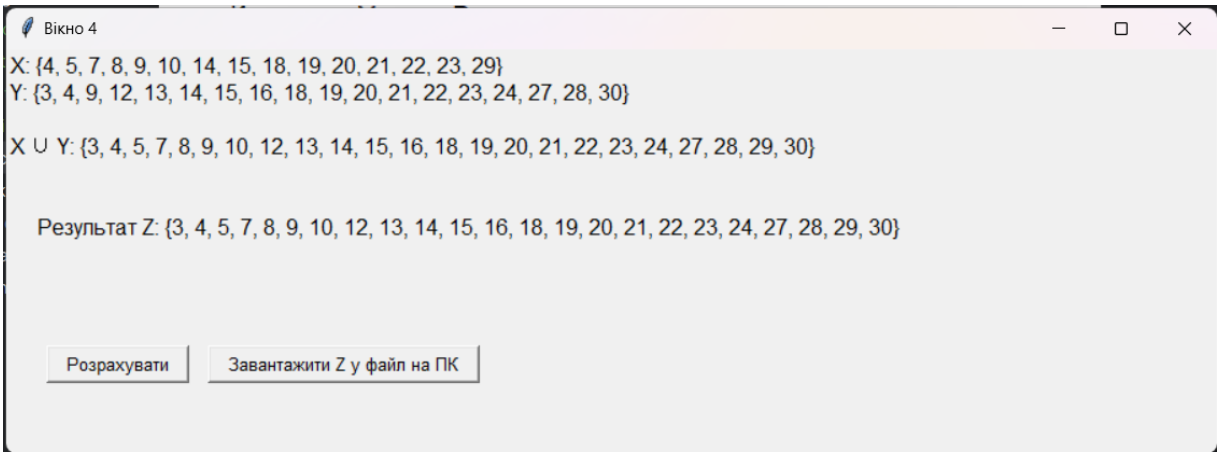
```
manual_data_set_c = Entry(root, width=12, font="Arial 12")
manual_data_set_c.place(x=280+141, y=280)
Button(root, width=8, text="Задати C", font="Arial 10", command=manual_input_set_c).place(x=280+141, y=310)

Button(root, width=8, text="Вікно 2", font="Arial 10", command=second_window).place(x=200-50, y=450)
Button(root, width=8, text="Вікно 3", font="Arial 10", command=third_window).place(x=300-50, y=450)
Button(root, width=8, text="Вікно 4", font="Arial 10", command=fourth_window).place(x=400-50, y=450)
Button(root, width=8, text="Вікно 5", font="Arial 10", command=fifth_window).place(x=500-50, y=450)
root.mainloop()
```

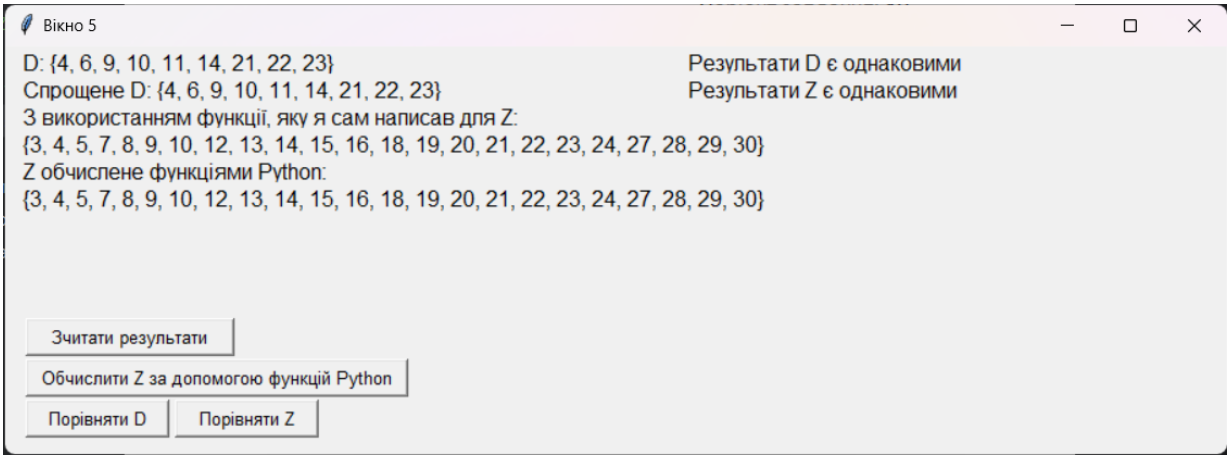
Скриншоти:

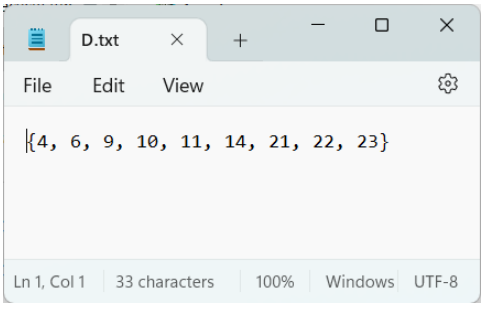


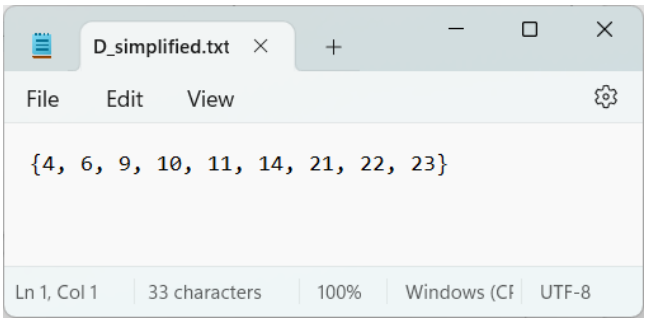


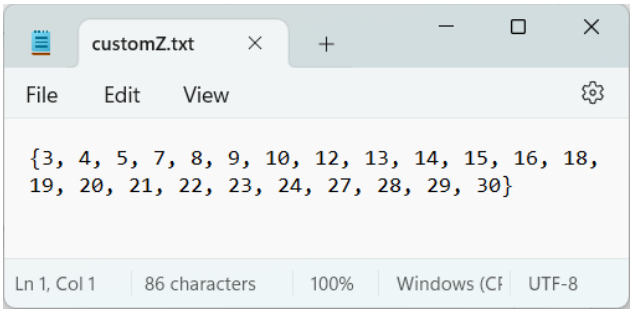

Вікно 3
A: {5, 6, 7, 8, 10, 11, 17, 25, 26, 29}
B: {4, 6, 9, 10, 11, 12, 13, 14, 21, 22, 23, 28}
C: {4, 5, 7, 8, 9, 10, 14, 15, 18, 19, 20, 21, 22, 23, 29}
C ∪ A: {4, 5, 6, 7, 8, 9, 10, 11, 14, 15, 17, 18, 19, 20, 21, 22, 23, 25, 26, 29}
C ∪ A ∩ B: {4, 6, 9, 10, 11, 14, 21, 22, 23}
Результат D: {4, 6, 9, 10, 11, 14, 21, 22, 23}
Крок 1
Крок 2
Завантажити D у файл на ПК


Вікно 4
X: {4, 5, 7, 8, 9, 10, 14, 15, 18, 19, 20, 21, 22, 23, 29}
Y: {3, 4, 9, 12, 13, 14, 15, 16, 18, 19, 20, 21, 22, 23, 24, 27, 28, 30}
X ∪ Y: {3, 4, 5, 7, 8, 9, 10, 12, 13, 14, 15, 16, 18, 19, 20, 21, 22, 23, 24, 27, 28, 29, 30}
Результат Z: {3, 4, 5, 7, 8, 9, 10, 12, 13, 14, 15, 16, 18, 19, 20, 21, 22, 23, 24, 27, 28, 29, 30}
Розрахувати
Завантажити Z у файл на ПК


Вікно 5
D: {4, 6, 9, 10, 11, 14, 21, 22, 23}
Спрощене D: {4, 6, 9, 10, 11, 14, 21, 22, 23}
З використанням функції, яку я сам написав для Z:
{3, 4, 5, 7, 8, 9, 10, 12, 13, 14, 15, 16, 18, 19, 20, 21, 22, 23, 24, 27, 28, 29, 30}
Z обчислене функціями Python:
{3, 4, 5, 7, 8, 9, 10, 12, 13, 14, 15, 16, 18, 19, 20, 21, 22, 23, 24, 27, 28, 29, 30}
Результати D є однаковими
Результати Z є однаковими
Зчитати результати
Обчислити Z за допомогою функцій Python
Порівняти D
Порівняти Z


D.txt
File
Edit
View
{4, 6, 9, 10, 11, 14, 21, 22, 23}
Ln 1, Col 1
33 characters
100%
Windows
UTF-8


D_simplified.txt
File
Edit
View
{4, 6, 9, 10, 11, 14, 21, 22, 23}
Ln 1, Col 1
33 characters
100%
Windows (CF
UTF-8


customZ.txt
File
Edit
View
{3, 4, 5, 7, 8, 9, 10, 12, 13, 14, 15, 16, 18,
19, 20, 21, 22, 23, 24, 27, 28, 29, 30}
Ln 1, Col 1
86 characters
100%
Windows (CF
UTF-8

Висновок: Виконавши цю лабораторну роботу, я зміг здобути відповідні навички в множинах: основні властивості та операції над ними, діаграми Венна. Під час виконання лабораторної роботи проблем не виникало, а складність була в структуруванні коду та приведенні його до більш гарного вигляду.