МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

**ЗВІТ**

**ПРО ВИКОНАННЯ ПРАКТИЧНОЇ РОБОТИ №4**

з дисципліни «Програмування мовою Python»

на тему «Обробка масивів»

Виконав студент групи КН-45-5

Письмак А.Ю

Перевірила [Кузнєцов Е. Г](https://mix.sumdu.edu.ua/info/userinfo/kuznetsov)

Варіант 13

Суми 2025

**Завдання 1**

1. **Постановка задачі**

Реалізувати одновимірний масив, довжину масива та сам масив користувач має ввести з клавіатури. Виконати над масивом обчислення, вказані  у Вашому варіанті.

****

1. **Текст програми**

# Письмак Архип Юрійович КН-45-5(13 варіант)

# Дано одномірний масив, що складається з N дійсних елементів. Масив користувач має ввести з клавіатури. Знайти мінімальний додатний елемент.

def read\_positive\_int(prompt: str) -> int:

"""

Зчитує з клавіатури додатне ціле число.

Повертає значення N, перевіряючи коректність введення.

"""

while True: # Повторюємо, поки користувач не введе коректне значення

try:

value = int(input(prompt).strip())

if value <= 0:

print("N має бути додатним цілим числом. Спробуйте ще раз.")

continue

return value

except ValueError:

print("Некоректне значення. Введіть додатне ціле число.")

def read\_float\_list(n: int) -> list[float]:

"""

Зчитує рівно n дійсних чисел, введених у один рядок.

Дозволяє використовувати як пробіли, так і коми як роздільники.

Повертає список з n чисел типу float.

"""

while True:

raw = input(f"Введіть {n} дійсних чисел через пробіл: ").strip()

# Замінюємо коми на пробіли для зручності введення: "1,2 3" -> "1 2 3"

parts = raw.replace(',', ' ').split()

if len(parts) != n:

print(f"Потрібно ввести рівно {n} чисел. Ви ввели {len(parts)}. Спробуйте ще раз.")

continue

try:

# Перетворюємо кожен фрагмент на дійсне число

return [float(x) for x in parts]

except ValueError:

print("Зустрілося нечислове значення. Переконайтесь, що вводите лише числа.")

def find\_min\_positive(arr: list[float]):

"""Повертає мінімальний додатний елемент масиву або None, якщо таких немає."""

# Відфільтровуємо лише додатні елементи (> 0)

positives = [x for x in arr if x > 0]

if not positives:

# Якщо додатніх елементів немає, повертаємо None

return None

return min(positives)

def main():

"""Точка входу в програму: зчитування даних та вивід результату."""

print("Завдання: Знайти мінімальний додатний елемент у масиві.")

n = read\_positive\_int("Введіть довжину масиву N: ")

arr = read\_float\_list(n) # Зчитуємо рівно N чисел

result = find\_min\_positive(arr)

if result is None:

print("У масиві немає додатних елементів.")

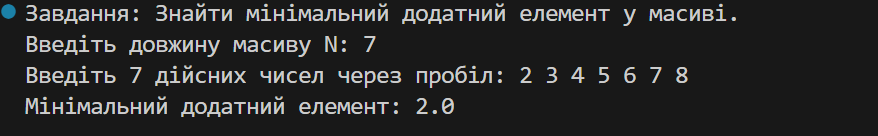
else:

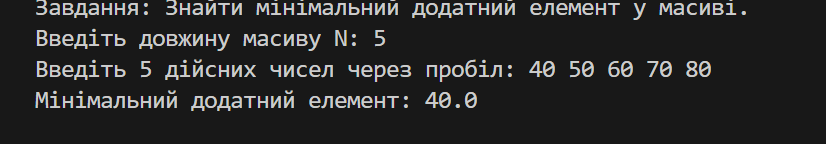
print(f"Мінімальний додатний елемент: {result}")

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

main()

1. **Приклад роботи програми**

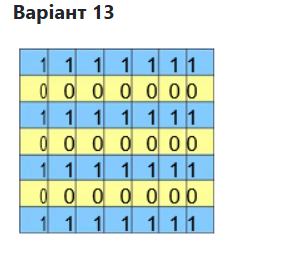
****

****

**Завдання 2**

1. **Постановка задачі**

Заповнити двовимірний масив розміром 7x7 таким чином, як показано на рисунку згідно з Вашим варіантом. Вивести масив на екран. Для виконання завдання використовуйте цикли.



1. **Текст програми**

# Письмак Архип Юрійович КН-45-5(13 варіант)

# Заповнити двовимірний масив 7x7 згідно рисунку.

ROWS = 7

COLS = 7

def value\_for\_position(i: int, j: int) -> int:

# Тут легко змінити під конкретний зразок.

return 1 if i % 2 == 0 else 0

def build\_matrix(rows: int, cols: int) -> list[list[int]]:

# Створює матрицю розміром rows x cols згідно правила value\_for\_position.

matrix: list[list[int]] = []

for i in range(rows): # зовнішній цикл за рядками

row: list[int] = []

for j in range(cols): # внутрішній цикл за стовпцями

row.append(value\_for\_position(i, j))

matrix.append(row)

return matrix

def print\_matrix(matrix: list[list[int]]) -> None:

# Друкує матрицю у зручному для читання вигляді.

for row in matrix:

# Вивід через пробіл без ком

print(" ".join(str(x) for x in row))

def main():

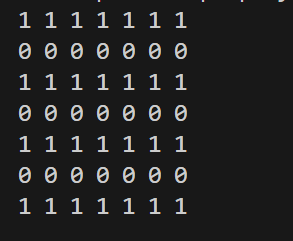
matrix = build\_matrix(ROWS, COLS)

print\_matrix(matrix)

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

main()

1. **Приклад роботи програми**

****

**Завдання 3**

1. **Постановка задачі**

Реалізувати функцію, яка виконує операції над списками – задану за варіантом та друк списку на екран. Список користувач має вводити з клавіатури.



1. **Текст програми**

# Письмак Архип Юрійович КН-45-5(13 варіант)

# Вставка нових елементів на парні позиції в списку.

def read\_list\_from\_input(prompt: str) -> list[str]:

#Зчитує список рядків, розділених пробілами. Коми також підтримуються як роздільники.

raw = input(prompt).strip()

# Дозволимо коми як роздільники: "a,b c" -> ["a","b","c"]

parts = raw.replace(',', ' ').split()

return parts

def insert\_on\_even\_positions(items: list[str], new\_item: str) -> list[str]:

"""Повертає новий список, де new\_item вставлено на всі парні позиції (1‑базово).

Схема формування результуючого списку (1‑базова нумерація позицій):

позиція 1 -> оригінал[0]

позиція 2 -> new\_item

позиція 3 -> оригінал[1]

позиція 4 -> new\_item

...

Отже, вставляємо new\_item після кожного елемента з індексом i, де i % 2 == 0 (0‑базово).

"""

result: list[str] = []

for i, val in enumerate(items):

result.append(val) # додаємо поточний елемент

if i % 2 == 0: # після 0‑го, 2‑го, 4‑го, ... вставляємо новий елемент

result.append(new\_item)

return result

def print\_list(items: list[str]) -> None:

# Друк списку у один рядок через пробіл.

print(' '.join(items))

def main() -> None:

print("Операція: вставка нового елемента на парні позиції списку (1‑базово).")

items = read\_list\_from\_input("Введіть елементи списку через пробіл (коми дозволені): ")

if not items:

print("Список порожній — нічого вставляти.")

return

new\_item = input("Введіть елемент, який потрібно вставляти на парні позиції: ").strip()

if new\_item == "":

print("Порожній елемент вставляти не будемо.")

return

result = insert\_on\_even\_positions(items, new\_item)

print("Початковий список:")

print\_list(items)

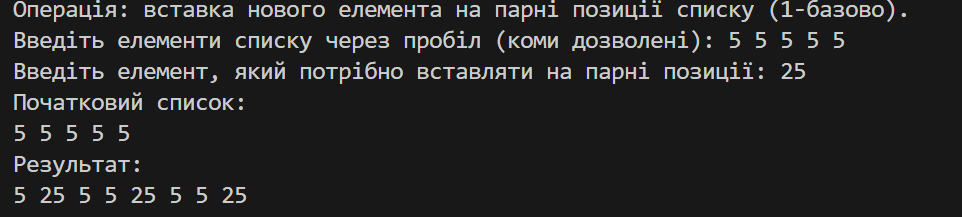
print("Результат:")

print\_list(result)

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

main()

1. **Приклад роботи програми**

****

**Завдання 4**

1. **Постановка задачі**

Реалізувати функцію, яка виконує операції над списками – задану за варіантом та друк списку на екран.



1. **Текст програми**

# Письмак Архип Юрійович КН-45-5(13 варіант)

# Поміняти місцями елементи списку з парними і непарними індексами.

def read\_list\_from\_input(prompt: str) -> list[str]:

# Зчитує список рядків, розділених пробілами. Коми також підтримуються як роздільники.

raw = input(prompt).strip()

parts = raw.replace(',', ' ').split()

return parts

def swap\_even\_odd\_indices(items: list[str]) -> list[str]:

# Повертає новий список, де елементи з парними і непарними індексами поміняні місцями попарно.

res = items.copy()

# Ітеруємося по парних індексах (0, 2, 4, ...), міняючи місцями i та i+1, якщо він існує

for i in range(0, len(res) - 1, 2):

res[i], res[i + 1] = res[i + 1], res[i]

return res

def print\_list(items: list[str]) -> None:

# Друк списку у один рядок через пробіл.

print(' '.join(items))

def main() -> None:

print("Операція: поміняти місцями елементи списку з парними і непарними індексами.")

items = read\_list\_from\_input("Введіть елементи списку через пробіл (коми дозволені): ")

if not items:

print("Список порожній — нічого міняти.")

return

result = swap\_even\_odd\_indices(items)

print("Початковий список:")

print\_list(items)

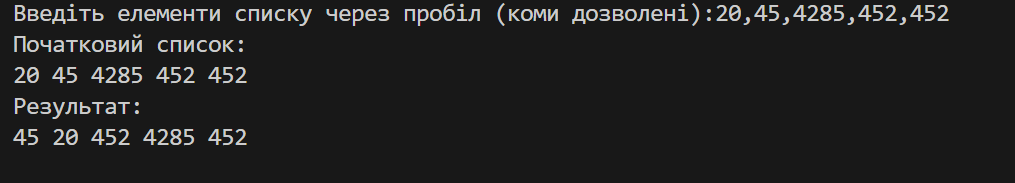
print("Результат:")

print\_list(result)

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

main()

1. **Приклад роботи програми**

****

**Завдання 5**

1. **Постановка задачі**

Реалізувати функцію, яка виконує операції над множинами – задану за варіантом та друк множини на екран. У випадку, якщо задану варіантом операцію над множиною виконати не можна, перетворіть множину у список, а потім при виведенні на екран результуючий список перетворіть на множину.



1. **Текст програми**

# Письмак Архип Юрійович КН-45-5(13 варіант)

# Задано текст з латинських літер. Символи цього тексту формують множину.Визначити та вивести всі літери, які входять у текст не менше двох разів.

from collections import Counter

import re

def read\_text(prompt: str) -> str:

# Зчитує довільний рядок від користувача

return input(prompt)

def filter\_latin\_letters(text: str) -> list[str]:

# Залишаємо тільки a-z/A-Z, перетворюємо у нижній регістр

letters = re.findall(r"[A-Za-z]", text)

return [ch.lower() for ch in letters]

def letters\_at\_least\_twice\_as\_set(text: str) -> set[str]:

# Обчислює множину літер, що зустрічаються не менше двох разів (без урахування регістру).

letters\_list = filter\_latin\_letters(text) # маємо СПИСОК латинських літер

counts = Counter(letters\_list) # підрахунок появ — операція над списком

result\_list = [ch for ch, c in counts.items() if c >= 2] # проміжний список результатів

return set(result\_list) # при виведенні перетворюємо на МНОЖИНУ

def print\_set\_as\_sorted(s: set[str]) -> None:

# Для зручності читацького сприйняття виведемо відсортовано, але як множину

print(set(sorted(s)))

def main() -> None:

print("Операція з множинами: вивести літери, які трапляються у тексті ≥ 2 разів (латиниця).")

text = read\_text("Введіть текст (латинські літери, інші символи ігноруються): ")

result\_set = letters\_at\_least\_twice\_as\_set(text)

if not result\_set:

print("Немає літер, що зустрічаються не менше двох разів.")

else:

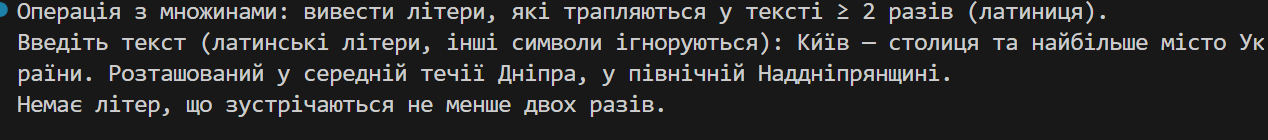
print("Множина літер, що зустрічаються ≥ 2 разів:")

print\_set\_as\_sorted(result\_set)

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

main()

1. **Приклад роботи програми**

****