МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

**ЗВІТ**

**ПРО ВИКОНАННЯ ПРАКТИЧНОЇ РОБОТИ №3**

з дисципліни «Програмування мовою Python»

на тему «Робота з рядками»

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**

Присвойте змінній довільний рядок в та отримайте з неї наступні зрізи, якщо довжина слова достатня для виконання операції зрізу.



1. **Текст програми**

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

# Передостанній символ рядка.

"""

Скрипт запитує у користувача рядок (якщо натиснути Enter — використається значення за замовчуванням)

та виводить різні зрізи з перевіркою на достатню довжину рядка.

"""

def show(label: str, condition: bool, value\_supplier):

# Допоміжна функція для безпечного виводу зрізів із перевіркою довжини.

if condition:

try:

value = value\_supplier()

except Exception as e:

print(f"{label}: помилка обчислення зрізу: {e}")

return

print(f"{label}: {value!r}")

else:

print(f"{label}: (довжина рядка недостатня)")

def main():

# Присвоюємо змінній довільний рядок (дозволяємо користувачу ввести свій)

user\_input = input("Введіть рядок (або натисніть Enter для прикладу): ").strip()

s = user\_input if user\_input else "Програмування мовою Python"

print(f"Рядок: {s!r} (довжина: {len(s)})\n")

# Базові зрізи/символи

show("1) Перший символ s[0]", len(s) >= 1, lambda: s[0])

show("2) Останній символ s[-1]", len(s) >= 1, lambda: s[-1])

show("3) Перші 3 символи s[:3]", len(s) >= 3, lambda: s[:3])

show("4) Останні 3 символи s[-3:]", len(s) >= 3, lambda: s[-3:])

# Декілька корисних фрагментів з умовами

show("5) Символи з 2-ї по 5-у s[1:5]", len(s) >= 5, lambda: s[1:5])

show("6) Середина без перших і останніх 3-х s[3:-3]", len(s) >= 6, lambda: s[3:-3])

show("7) Перші 8 символів, починаючи з 3-го s[2:10]", len(s) >= 3, lambda: s[2:10])

# Кроки та розворот

# Ці зрізи формально коректні для будь-якої довжини, але залишимо однаковий стиль виводу

show("8) Кожен 2-й символ s[::2]", len(s) >= 1, lambda: s[::2])

show("9) Кожен 2-й символ, починаючи з 2-го s[1::2]", len(s) >= 2, lambda: s[1::2])

show("10) Рядок у зворотному порядку s[::-1]", len(s) >= 1, lambda: s[::-1])

show("11) Кожен 2-й у зворотному порядку s[::-2]", len(s) >= 1, lambda: s[::-2])

# Індекси поза діапазоном безпечні для зрізів, але не для одиночних доступів

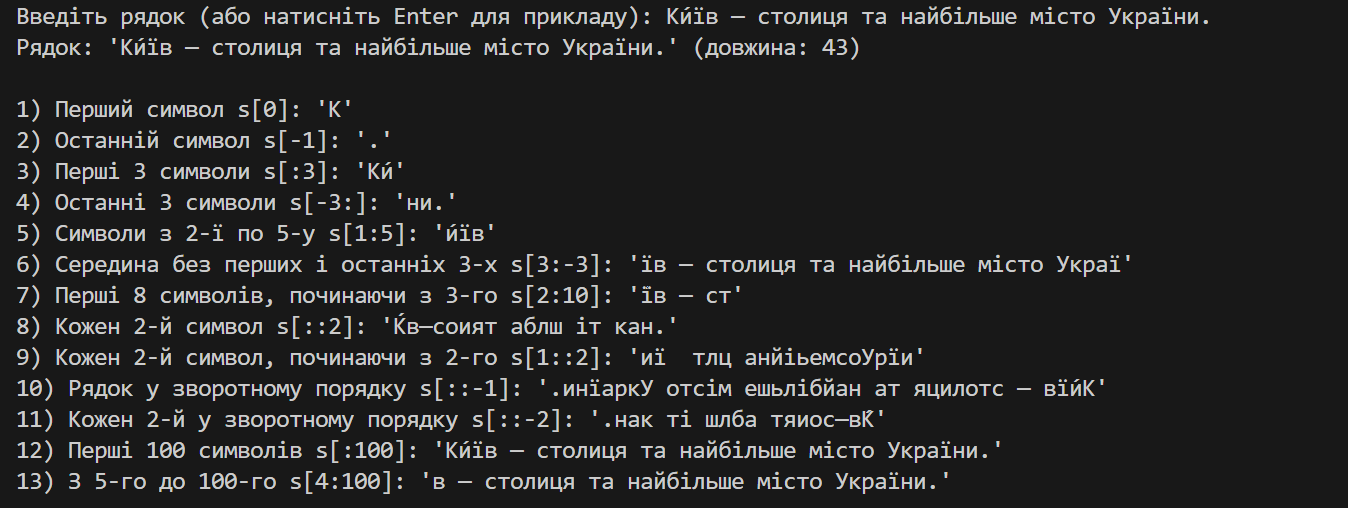
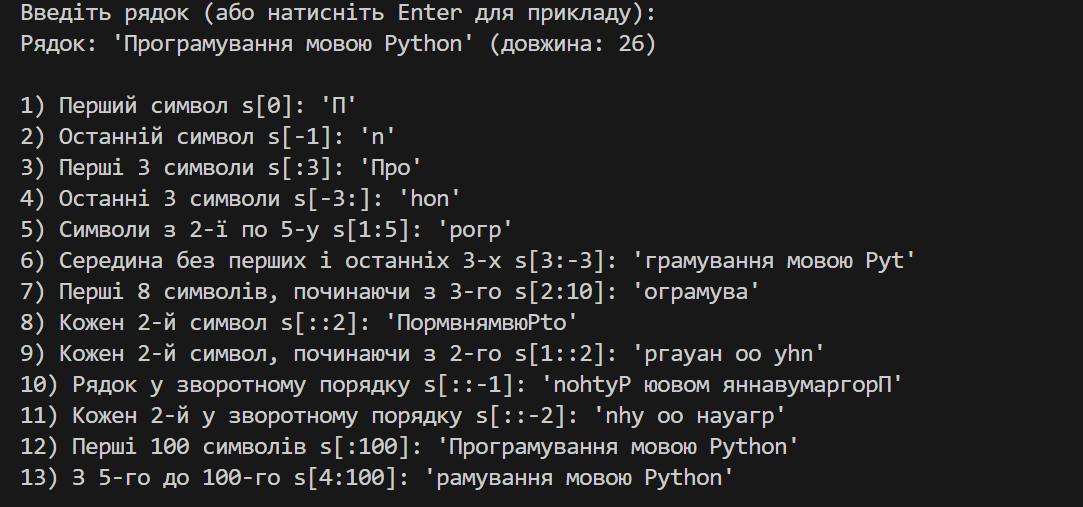
show("12) Перші 100 символів s[:100]", len(s) >= 1, lambda: s[:100])

show("13) З 5-го до 100-го s[4:100]", len(s) >= 5, lambda: s[4:100])

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

main()

1. **Приклад роботи програми**

****

**Завдання 2**

1. **Постановка задачі**

Задане слово. Видалити з нього усі символи, що не є буквами.

1. **Текст програми**

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

# Задане слово. Видалити з нього усі символи, що не є буквами.

def clean\_word(s: str) -> str:

# Повертає рядок, у якому лишилися лише літери (Unicode).

return "".join(ch for ch in s if ch.isalpha())

def main():

s = input("Введіть слово (можна з будь-якими символами): ")

cleaned = clean\_word(s)

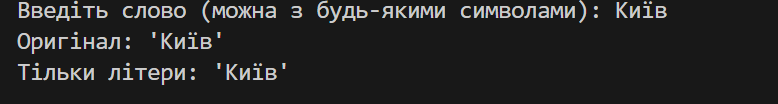
print(f"Оригінал: {s!r}")

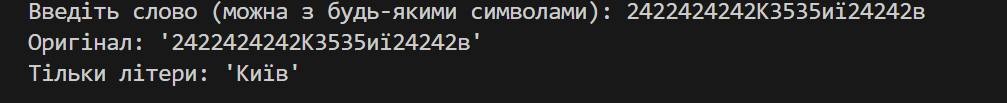
print(f"Тільки літери: {cleaned!r}")

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

main()

1. **Приклад роботи програми**

****

****

**Завдання 3**

1. **Постановка задачі**

Задано речення. Скласти програму, яка визначає і виводить на екран всі слова, які зустрічаються в реченні один раз.

1. **Текст програми**

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

# Задано речення. Скласти програму, яка визначає і виводить на екран всі слова, які зустрічаються в реченні один раз.

from collections import Counter

def extract\_words(s: str) -> list[str]:

"""

Повертає список слів як послідовностей літер (Unicode), без цифр і розділових знаків.

Словом вважаємо лише безперервну послідовність символів, для яких ch.isalpha() == True.

Напр., апостроф або дефіс не включаємо до слова, бо це не літера.

"""

words: list[str] = []

buff: list[str] = []

for ch in s:

if ch.isalpha():

buff.append(ch)

else:

if buff:

words.append("".join(buff))

buff.clear()

if buff:

words.append("".join(buff))

return words

def find\_unique\_words(sentence: str) -> list[str]:

# Повертає слова, що трапляються рівно один раз (реєстр нечутливий), у порядку появи, з оригінальним написанням.

words = extract\_words(sentence)

counts = Counter(w.lower() for w in words)

return [w for w in words if counts[w.lower()] == 1]

def main():

sentence = input("Введіть речення: ")

uniques = find\_unique\_words(sentence)

if uniques:

print("Слова, що зустрічаються один раз:")

for w in uniques:

print(w)

else:

print("Немає слів, що зустрічаються лише один раз.")

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

main()

1. **Приклад роботи програми**

