**Зміст**

[**Завдання до теми 1** 3](#_Toc147860258)

[1. Обернути рядок у зворотному порядку. 3](#_Toc147860259)

[1) Метод нарізання: 3](#_Toc147860260)

[2) За допомгою циклу for: 3](#_Toc147860261)

[3) За допомогою функції reversed(): 3](#_Toc147860262)

[**Результат**: На рисунку 1.3 3](#_Toc147860263)

[**2.** **Виконати тестування функцій, що працюють з рядками: strip(), capitalize(), title(), upper(), lower().** 4](#_Toc147860264)

[1) Тестування функції strip(): 4](#_Toc147860265)

[Результат виконання коду показаний на рисунку 2.1. 4](#_Toc147860266)

[2) Тестування функції capitalize(): 4](#_Toc147860267)

[Результат виконання коду показаний на рисунку 2.3 4](#_Toc147860268)

[3) Тестування функції title(): 4](#_Toc147860269)

[Результат виконання коду показано на рисунку 2.5 5](#_Toc147860270)

[4) Тестування функції upper(): 5](#_Toc147860271)

[Результат виконання коду показано на рисунку 2.7 5](#_Toc147860272)

[5) Тестування функції lower(): 5](#_Toc147860273)

[Результат виконання коду показано на рисунку 2.9 5](#_Toc147860274)

[3.**Написати функцію для пошуку дискримінанту квадратного рівняння** 5](#_Toc147860275)

[Результат виконання коду показано на рисунку 3.1. 6](#_Toc147860276)

[**Завдання до теми 2** 6](#_Toc147860277)

[**1.** **Написати функцію пошуку коренів квадратного рівняння використовуючи функцію розрахунку дискримінанту з попередньої теми та умовні переходи** 6](#_Toc147860278)

[Пояснення: 7](#_Toc147860279)

[**2.** **Написати програму калькулятор використовуючи if else конструкцію. Кожна операція має бути виконана в окремій функції.** 8](#_Toc147860280)

[Пояснення: 8](#_Toc147860281)

[**3.** **Написати програму калькулятор використовуючи match конструкцію. Кожна операція має бути виконана в окремій функції.** 9](#_Toc147860282)

[Пояснення: 9](#_Toc147860283)

[**Завдання до теми 3** 10](#_Toc147860284)

[**1.** **Написати програму калькулятор з постійними запитами на введення нових даних та операцій. За основу взяти програму калькулятор з попередньої теми. Реалізувати механізм завершення програми після отримання відповідної команди.** 10](#_Toc147860285)

[Пояснення: 10](#_Toc147860286)

[**2.** **Написати програму тестування функцій списків таких як: extend(), append(), insert(id, val), remove(val), clear(), sort(), reverse(), copy()** 11](#_Toc147860287)

[1) Функція sort: 11](#_Toc147860288)

[2) Функція reverse: 12](#_Toc147860289)

[3) Функція append: 12](#_Toc147860290)

[4) Функція extend: 12](#_Toc147860291)

[6) Функція remove: 12](#_Toc147860292)

[7) Функція copy: 13](#_Toc147860293)

[8) Функція clear: 13](#_Toc147860294)

[**3.** **Написати програму тестування функцій словників таких як: update(), del(), clear(), keys(), values(), items()** 13](#_Toc147860295)

[1) Функція update: 13](#_Toc147860296)

[2) Функція del: 13](#_Toc147860297)

[3) Функція clear: 14](#_Toc147860298)

[4) Функція keys: 14](#_Toc147860299)

[5) Функція values: 14](#_Toc147860300)

[6) Функція items: 14](#_Toc147860301)

[**4.** **Маючи відсортований список, написати функцію пошуку позиції для вставки нового елементу в список.** 15](#_Toc147860302)

[Пояснення: 15](#_Toc147860303)

# **Завдання до теми 1**

1. Обернути рядок у зворотному порядку.

Суть задачі: Якщо в нас є якийсь рядок st, наприклад "abcdefg123", то він має бути “ 321gfedcba”.

Я зробив декілька варіантів цієї роботи:

1. Метод нарізання:

Суть методу полягає в тому, що ми нарізаємо рядок на окремі символи, після чого всталяємо його в якусь змінну. В пайтоні є спеціальна команда яка це виконує(рис 1.0).

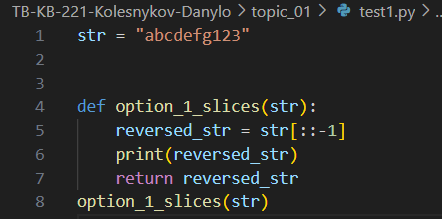


Рисунок 1.0 – Метод нарізання

1. За допомгою циклу for:

Суть полягає в тому, що ми переписуємо кожен символ в нову змінну, і так робимо по колу, поки не пройдемо весь стрінг(рис.1.1)

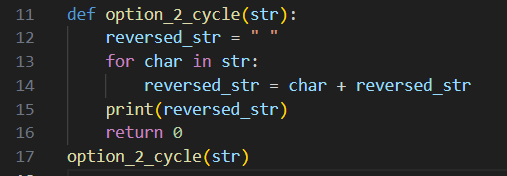


Рисунок 1.1

1. За допомогою функції reversed():

Суть полягає в тому, що за допомогою двох методів reversed та join, ми перевертаємо рядок, та записуємо його в пусту змінну. Даний код більш читабельний ніж попередній, оскільки використовує команди, які напряму для цього призначені(рис. 1.2)

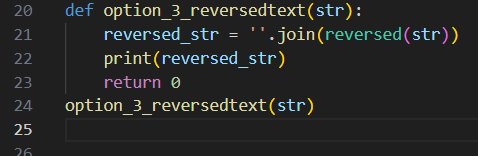


Рисунок 1.2

**Результат**: На рисунку 1.3

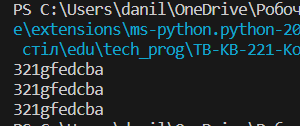


Рисунок 1.3

1. **Виконати тестування функцій, що працюють з рядками: strip(), capitalize(), title(), upper(), lower().**
2. Тестування функції strip():

Дана функція прибирає відступи на початку та в кінці рядку. Приклад коду показано на рисунку 2.0.

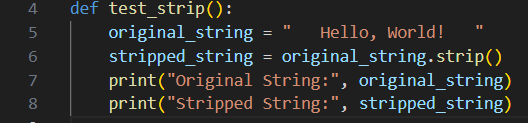


Рисунок 2.0

Результат виконання коду показаний на рисунку 2.1.



Рисунок 2.1

1. Тестування функції capitalize():

Дана команда переводить у верхній регістр першу літеру першого слова. Приклад використання команди показано на рисунку 2.2.

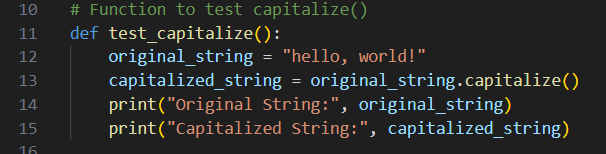


Рисунок 2.2

Результат виконання коду показаний на рисунку 2.3



Рисунок 2.3

1. Тестування функції title():

Дана функція перетворює всі перші літери кожного слова в верхній регістр. Приклад використання коду зазначено на рисунку 2.4.

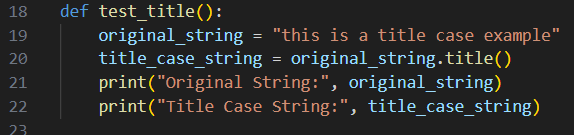


Рисунок 2.4

Результат виконання коду показано на рисунку 2.5



Рисунок 2.5

1. Тестування функції upper():

Дана функція повертає всі літери у верхньому регістрі. Приклад виконання коду показано на рисунку 2.6

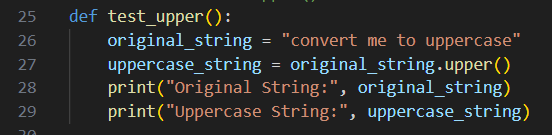


Рисунок 2.6

Результат виконання коду показано на рисунку 2.7



Рисунок 2.7

1. Тестування функції lower():

Дана функція повертає всі літери в нижньому регістрі. Приклад використання коду показано на рисунку 2.8.

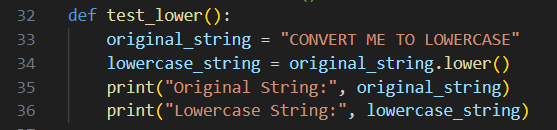


Рисунок 2.8

Результат виконання коду показано на рисунку 2.9



Рисунок 2.9

1. **Написати функцію для пошуку дискримінанту квадратного рівняння**

Формула дискримінанту:

D = b2 – 4ac

Код для розв’язання даної задачі показано на рисунку 3.0

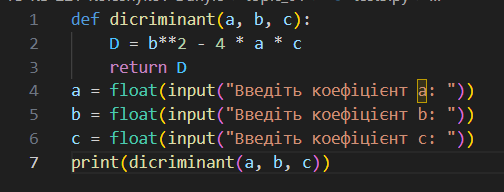


Рисунок 3.0

Пояснення: Я створив функцію яку назвав дискримінант, присвоїв їй 3 параметри, a, b та c, які є коефіцієнтами в рівнянні. Після чого написав формулу яка була зазначена вище. Та повернув результат обчислення в змінній D. Після чого створив 3 змінні які є коефіцієнтами з квадратного рівняння, та запитав користувача дані про ці коефіцієнти , після чого просто в функції print викликаю функцію з прив’язаними параметрами.

Результат виконання коду показано на рисунку 3.1.

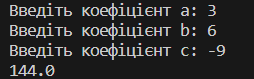


Рисунок 3.1

**Завдання до теми 2**

1. **Написати функцію пошуку коренів квадратного рівняння використовуючи функцію розрахунку дискримінанту з попередньої теми та умовні переходи**

Код для виконання даного завдання показано на рисунку 4.0.

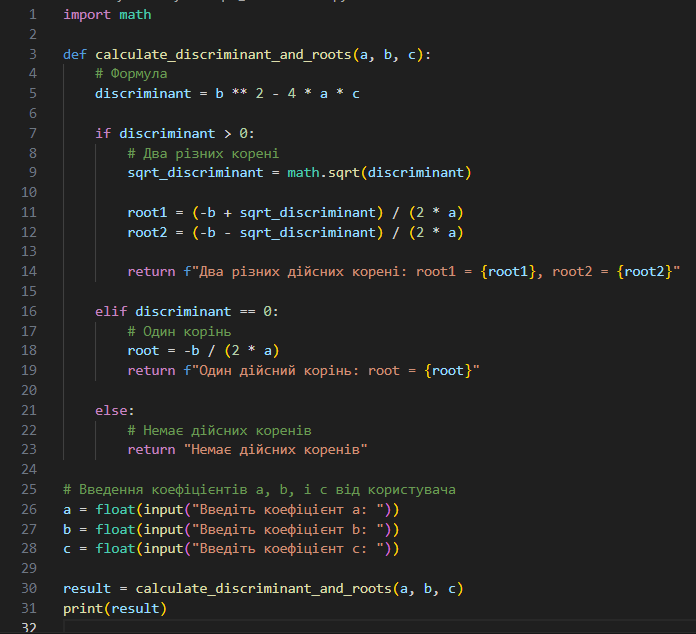


Рисунок 4.0

Пояснення:

Основна функція в програмі - це calculate\_discriminant\_and\_roots(a, b, c), яка приймає три аргументи a, b і c, які представляють коефіцієнти квадратного рівняння ax^2 + bx + c = 0. Спочатку обчислюється дискримінант за допомогою формули discriminant = b \*\* 2 - 4 \* a \* c, де \*\* - це оператор піднесення до ступеня. Потім використовується умовний оператор if, щоб перевірити значення дискримінанту: Якщо discriminant > 0, то це означає, що існують два різних дійсних корені квадратного рівняння. Обчислюються обидва корені (root1 і root2) за допомогою квадратного кореня (math.sqrt) і формули рішення квадратного рівняння. Результат повертається у вигляді рядка, що містить обидва корені. Якщо discriminant == 0, то це означає, що існує лише один дійсний корінь квадратного рівняння. Обчислюється цей корінь (root) і повертається у вигляді рядка з одним коренем. Якщо discriminant менше нуля, то квадратне рівняння не має дійсних коренів, і програма повертає рядок "Немає дійсних коренів". Після обчислення результату програма запитує користувача ввести значення коефіцієнтів a, b, і c. Викликається функція calculate\_discriminant\_and\_roots з введеними користувачем значеннями, і результат обчислень записується у змінну result. Наостанок, результат виводиться на екран за допомогою функції print.

Результат показано на рисунку 4.1.

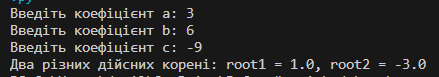


Рисунок 4.1

1. **Написати програму калькулятор використовуючи if else конструкцію. Кожна операція має бути виконана в окремій функції.**

Код для виконання даного коду зазначено на рисунку 4.2.

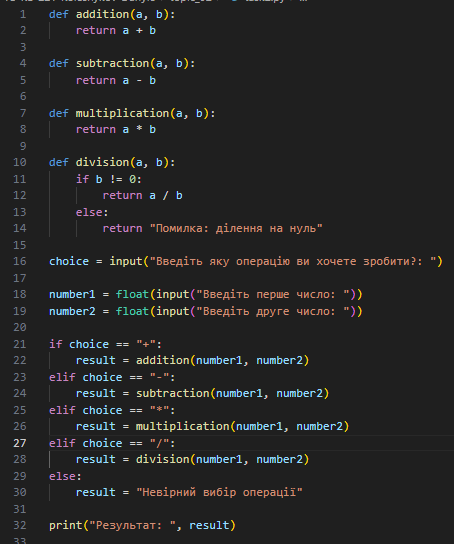


Рисунок 4.2

Пояснення:

Спочатку визначені чотири функції: addition(a, b) виконує додавання двох чисел a і b. subtraction(a, b) виконує віднімання b від a. multiplication(a, b) виконує множення двох чисел a і b. division(a, b) виконує ділення a на b. Перед виконанням ділення перевіряється, чи b не дорівнює нулю, щоб уникнути ділення на нуль. Користувачу пропонується ввести операцію, яку він бажає виконати (додавання, віднімання, множення або ділення), за допомогою рядка choice = input("Введіть яку операцію ви хочете зробити?: "). Користувач також запитується ввести два числа number1 і number2, над якими буде виконано вибрану операцію. За допомогою умовного оператора if-elif-else визначається, яку операцію обрано. Залежно від вибору, викликається відповідна функція (наприклад, якщо вибрано +, то викликається addition(number1, number2)), і результат обчислень записується у змінну result. Якщо користувач ввів невірну операцію (не "+", "-", "\*", або "/"), то в result записується повідомлення про невірний вибір операції. Наостанок, результат (результат операції або повідомлення про невірний вибір) виводиться на екран за допомогою функції print.

Результат виконання коду зазначений на рисунку 4.3.

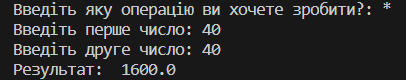


Рисунок 4.3

1. **Написати програму калькулятор використовуючи match конструкцію. Кожна операція має бути виконана в окремій функції.**

Код для виконання цієї задачі показано на рисунку 4.4.

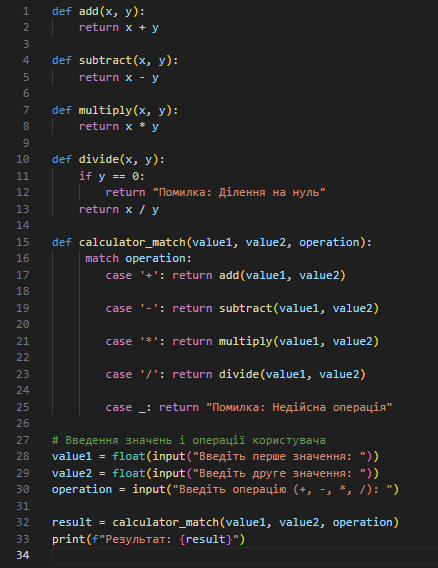


Рисунок 4.4

Пояснення:

Спочатку визначені чотири функції для виконання математичних операцій: add, subtract, multiply, і divide. Кожна з цих функцій приймає два аргументи x і y і виконує відповідну операцію (додавання, віднімання, множення або ділення). Функція calculator\_match(value1, value2, operation) приймає три аргументи: value1, value2, і operation. Вона використовує вираз match для визначення вибраної операції і викликає відповідну функцію для обчислення результату. Якщо операція недійсна, то повертається повідомлення про помилку. Користувачу пропонується ввести два значення value1 і value2 і вибрати операцію (operation) введенням "+", "-", "\*", або "/". За допомогою виразу match визначається вибрана операція. У вас є п'ять можливих варіантів: "+" для додавання, "-" для віднімання, "\*" для множення, "/" для ділення, і \_, що є варіантом за замовчуванням для недійсної операції. Викликається відповідна функція для виконання обраної операції над value1 і value2. Якщо операція недійсна, то програма повертає повідомлення про помилку. Наостанок, результат (результат операції або повідомлення про помилку) виводиться на екран за допомогою функції print.

**Завдання до теми 3**

1. **Написати програму калькулятор з постійними запитами на введення нових даних та операцій. За основу взяти програму калькулятор з попередньої теми. Реалізувати механізм завершення програми після отримання відповідної команди.**

Код для вирішення цієї задачі вказано на рисунку 5.0.

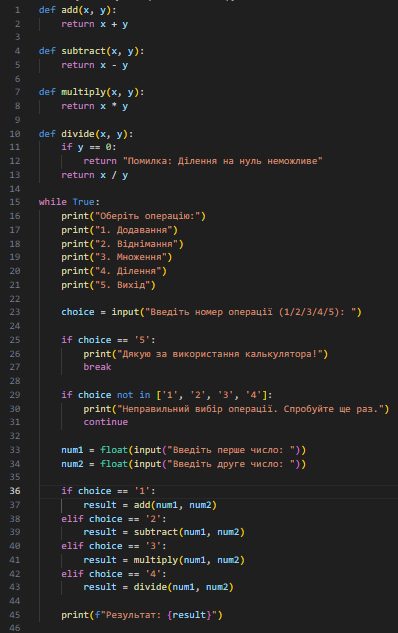


Рисунок 5.0

Пояснення:

Визначені чотири функції для виконання математичних операцій: add, subtract, multiply і divide. Кожна з цих функцій приймає два аргументи x і y і виконує відповідну операцію (додавання, віднімання, множення або ділення). Функція divide також перевіряє, чи y дорівнює нулю перед виконанням ділення. Введений цикл while True, який буде виконуватися безкінечно, доки користувач не обере опцію "Вихід" (5). У циклі виводяться доступні операції для користувача. Користувач вводить номер операції (1, 2, 3, 4 або 5) за допомогою функції input. Якщо користувач обирає "Вихід" (5), програма завершується за допомогою команди break. Якщо введений номер операції не відповідає жодному з варіантів (1, 2, 3 або 4), то програма виводить повідомлення про помилку і переходить до наступної ітерації циклу за допомогою команди continue. Користувач вводить два числа (num1 і num2) для виконання обраної операції. Відповідна функція викликається в залежності від обраної операції, і результат обчислення записується в змінну result. Результат виводиться на екран за допомогою функції print. Після виконання операції програма повертається до початку циклу, де користувач може обрати іншу операцію або вийти з програми.

Результат виконання коду зазначені на рисунках 5.1 та 5.2

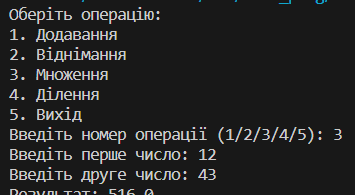


Рисунок 5.1

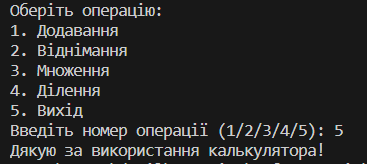


Рисунок 5.2

1. **Написати програму тестування функцій списків таких як: extend(), append(), insert(id, val), remove(val), clear(), sort(), reverse(), copy()**
2. Функція sort:

Дана функція сортує список від найменшого до найбільшого числа. Приклад на рисунку 5.3

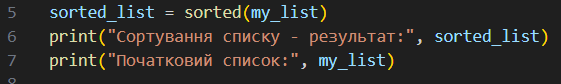
****

Рисунок 5.3

Результат на рисунку 5.4.



Рисунок 5.4

1. Функція reverse:

Дана функція повертає список у зворотному напрямку. Приклад на рисунку 5.5.

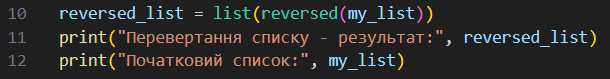


Рисунок 5.5

Результат на рисунку 5.6.



Рисунок 5.6

1. Функція append:

Дана функція додає елемент до списку. Приклад на рисунку 5.7.



Рисунок 5.7

Результат на рисунку 5.8.



Рисунок 5.8

1. Функція extend:

Дана функція розширює список. Працює приблизно як і попередня команда. Приклад на рисунку 5.9.

****

Рисунок 5.9

Результат на рисунку 5.10.



Рисунок 5.10

1. Функція insert:

Дана функція замінює елемент по його порядковому порядку на елемент який вказує користувач. Приклад на рисунку 5.11.



Рисунок 5.11

Результат на рисунку 5.12.



Рисунок 5.12

1. Функція remove:

Дана функція видаляє значення. Приклад на рисунку 5.13.

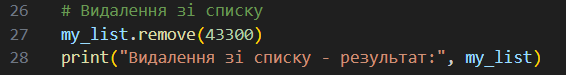
****

Рисунок 5.13

Результат на рисунку 5.14.



Рисунок 5.14

1. Функція copy:

Дана функція копіює список до змінної. Приклад вказано на рисунку 5.15.

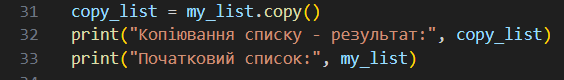
****

Рисунок 5.15

Результат на рисунку 5.16.



Рисунок 5.16

1. Функція clear:

Дана команда видаляє всі дані в списку. Приклад вказано на рисунку 5.17.

****

Рисунок 5.17

Результат на рисунку 5.18.



Рисунок 5.18

1. **Написати програму тестування функцій словників таких як: update(), del(), clear(), keys(), values(), items()**
2. Функція update:

Дана функція дозволяє оновити дані конкретного ключа. Приклад використання на рисунку 5.19.

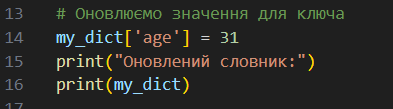


Рисунок 5.19

Результат представлений на рисунку 5.20.



Рисунок 5.20

1. Функція del:

Дана функція видаляє ключ та його значення. Приклад на рисунку 5.21.

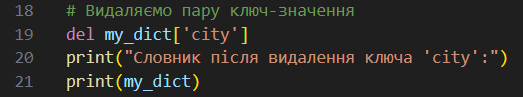


Рисунок 5.21

Результат на рисунку 5.22.



Рисунок 5.22

1. Функція clear:

Дана функція видаляє повністю список. Приклад використання на рисунку 5.23.

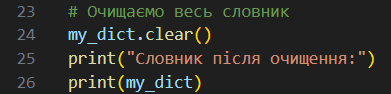


Рисунок 5.23

Результат на рисунку 5.24.



Рисунок 5.24

1. Функція keys:

Дана функція дозволяє нам отримати ключі цього словника. Приклад використання на рисунку 5.25.

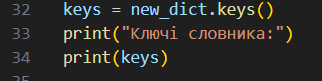


Рисунок 5.25

Результат на рисунку 5.26.



Рисунок 5.26

1. Функція values:

Дана функція дає нам можливість отримувати значення які зберігають ключі. Приклад на рисунку 5.27.

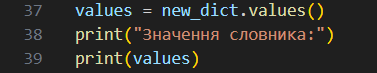


Рисунок 5.27

Результат на рисунку 5.28.



Рисунок 5.28

1. Функція items:

Дана функція дозволяє нам отримати пару Ключ – Значення. Приклад на рисунку 5.29.

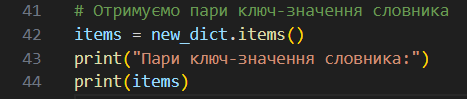


Рисунок 5.29

Результат на рисунку 5.30.



Рисунок 5.30

1. **Маючи відсортований список, написати функцію пошуку позиції для вставки нового елементу в список.**

Код для виконання даного завдання вказаний на рисунку 5.31.

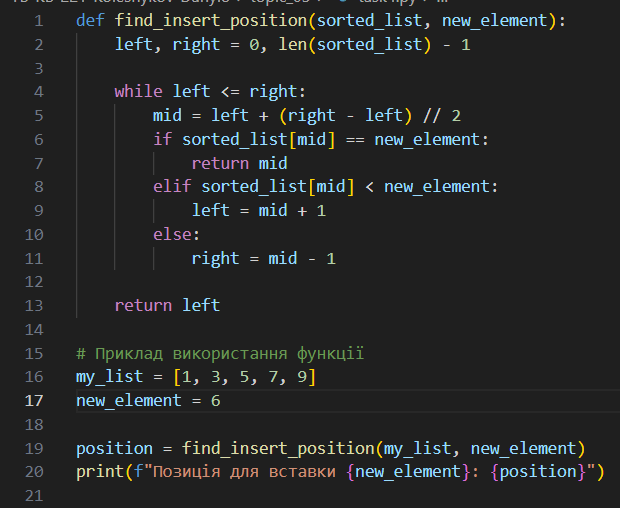


Рисунок 5.31

Пояснення:

Початкові значення змінних left і right встановлюються так, щоб охопити всі елементи від початку до кінця списку. left вказує на перший елемент списку, а right вказує на останній. У циклі while виконується пошук позиції для вставки new\_element. Поки left не більше або дорівнює right, виконуються наступні кроки: Обчислюється середній індекс mid як середнє значення між left і right. Порівнюється значення елементу sorted\_list[mid] з new\_element: Якщо sorted\_list[mid] дорівнює new\_element, то знайдена точна позиція для вставки, і функція повертає mid. Якщо sorted\_list[mid] менше за new\_element, то це означає, що new\_element повинен бути вставлений праворуч від поточного mid. Тому змінна left оновлюється на mid + 1, для того щоб вужчити область пошуку. Якщо sorted\_list[mid] більше за new\_element, то це означає, що new\_element повинен бути вставлений ліворуч від поточного mid. Тому змінна right оновлюється на mid - 1, для того щоб вужчити область пошуку. Якщо left стає більше за right, то це означає, що пошук завершився без знаходження точної позиції для вставки new\_element. У такому випадку функція повертає left, оскільки це позиція, на яку потрібно вставити new\_element.

Результат виконання коду зазначено на рисунку 5.32.



Рисунок 5.32

**Завдання до теми 4**

1. **Розширити програму калькулятор функцією запитів даних для виконання операцій від користувача, що обробляє виняткові ситуації.**
2. **Розширити функцію ділення обробкою виняткової ситуації ділення но нуль.**

Отож, попередній код показаний на рисунку 6.0. А вже модернізований код показано на рисунку 6.1. В попередній код були внесені наступні зміни:

1. **Запити від користувача та обробка некоректних виборів:**

* Додано головний цикл while True, який дозволяє користувачеві виконувати декілька операцій без виходу з програми.
* Запит користувача на вибір операції було розширено, та тепер користувач може обирати між додаванням, відніманням, множенням, діленням або виходом з програми.
* Додано перевірку коректності введеного вибору операції, і якщо вибір не входить у доступні опції (1-4), програма повідомляє користувача про помилку та просить ввести операцію знову.

1. **Обробка виняткової ситуації ділення на нуль:**

* У функцію divide додано блок try...except, який перехоплює можливий виняток ZeroDivisionError, що виникає при діленні на нуль.
* Якщо користувач введе друге число рівне нулю, програма повідомить про помилку "Помилка: Ділення на нуль неможливе" і продовжить виконання без аварійного завершення.

Загальна логіка коду наступна:

1. Користувач вибирає операцію та вводить два числа.
2. Програма виконує вибрану операцію та виводить результат на екран.
3. Якщо користувач введе некоректний вибір операції, виводиться відповідне повідомлення.
4. Якщо користувач введе друге число рівне нулю при діленні, виводиться повідомлення про помилку.