**Звіт про виконання практичних завдань по Технологіях програмування на мові Python**

**Практичне завдання до теми 1**

1. **Перевертання рядка:**

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).

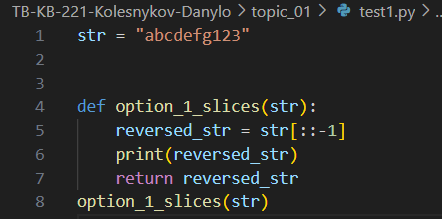


Рисунок 1.0 – Метод нарізання

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)

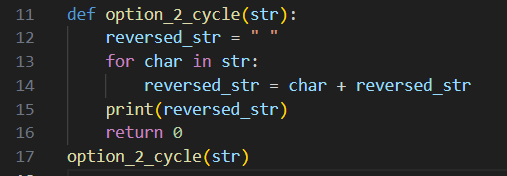


Рисунок 1.1

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, оськільки використовує команди, які напряму для цього призначені(рис. 1.2)

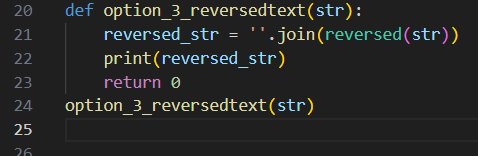


Рисунок 1.2

І результат виконання коду на рисунку 1.3.

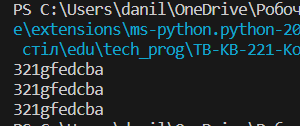


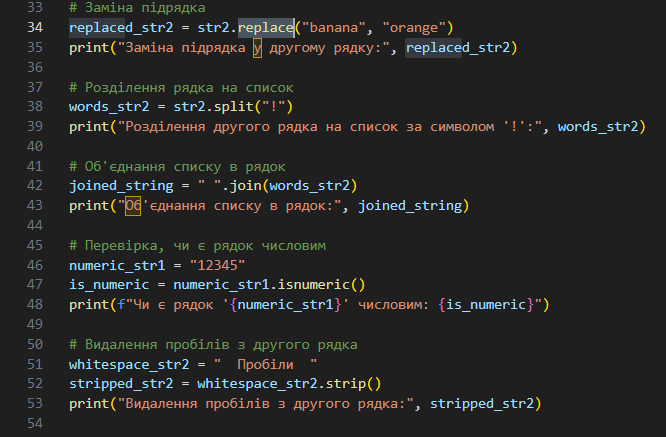
Рисунок 1.3

1. **Протестувати базові функції для рядків:**

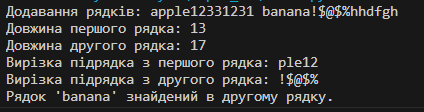
Команди які були використані для виконання даного завдання наведені нижче:

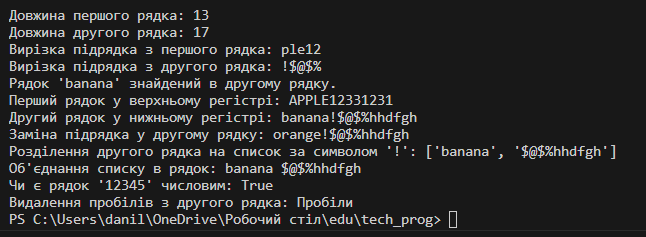
1. result = str1 + " " + str2 – додавання рядків. Використовуємо оператор +
2. len() – команда для визначення довжини рядку
3. substring\_str1 = str1[2:7] – вирізка з першого рядка. Вирізаємо символи у яких порядковий номер починається з 2 і до 7
4. if search\_string in str2: - пошук символів в рядку за допомогою оператора in
5. upper() – переведення символів в рядку в верхній регістр
6. lower() – переведення символів в нижній регістр
7. .replace() – заміна одного рядка в іншому.
8. Split() – розділення рядка на списки
9. Join() – об’єднання списку в рядок
10. Isnumeric() – перевірка чи є рядок числовим



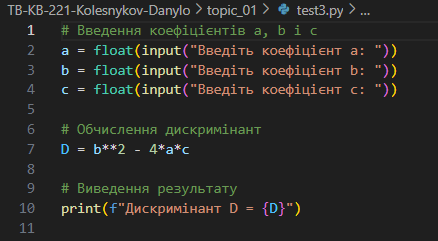


Результати наведені нижче:



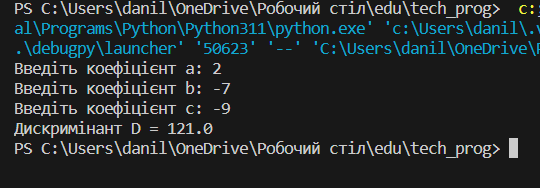


1. **Знайти Дискримінант**



Формула дискримінанту це: D = b^2 - 4ac

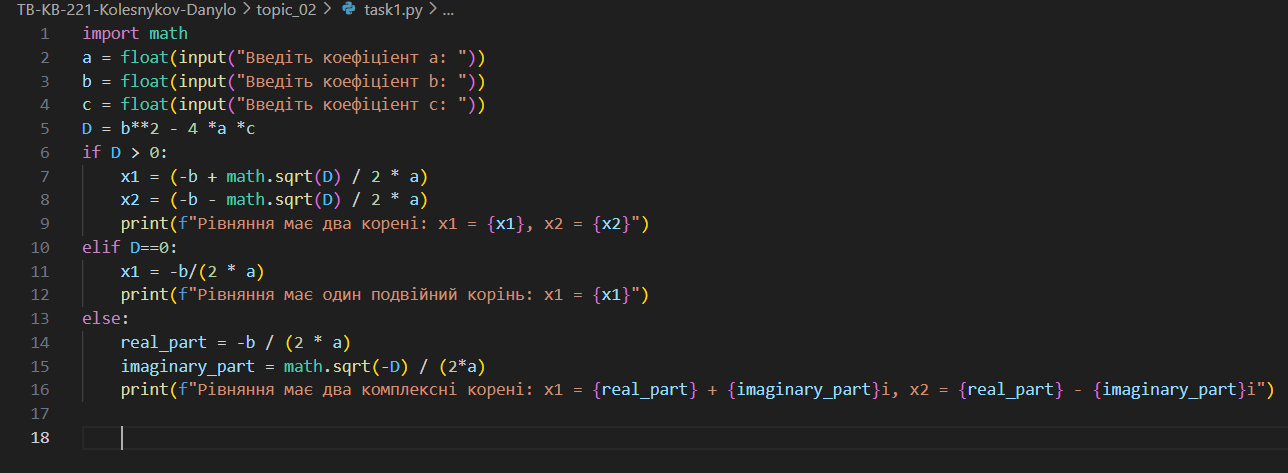
Отож, нам потрібні 3 коефіцієнти, після чого створимо ному змінну D, в якій за допомого двох зірок підносимо коефіцієнт b до степеня, який ми напишемо після зірок, в нашому випадку це 2. Далі пишемо – 4 і за допомогою зірочки множимо інші коефіцієнти. Далі просто виводимо результат.



**Практичне завдання до теми 2**

1. **Знайти значення коренів, враховуючи значення дискримінанту**

Беремо код нашого минулого завдання з дискримінантом, і трохи його модернізуємо.

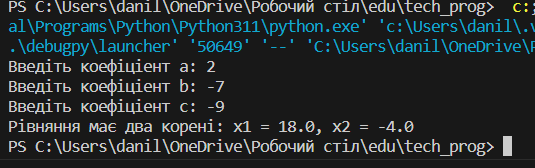


Створимо логічний оператор if, якщо дискримінант більше нуля, то він має два корені.

Якщо дискримінант дорівнює нулю, то дискримінант має один подвійний корінь.

В іншому випадку, рівняння буде мати 2 комплексні корені.

Результат:



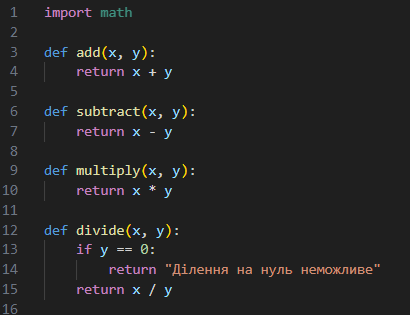
1. **Програма калькулятор на основі if elif else**



Тут я створив дві змінні в які потрібно буде вписати два числа, а також змінну для операції, тобто знаку операції. Після чого зробив прописав за допомогою операторів if elif else наступне: Якщо оператор плюс то два числа додаються, якщо -, віднімаються, якщо \* то множаться, якщо / то діляться. Також додав перевірку, якщо число 2 дорівнює 0, то виводиться помилка яка каже нам, що ділення на 0 неможливе. Вивів результат.

1. **Також зробити калькулятор, але на основі оператора match.**

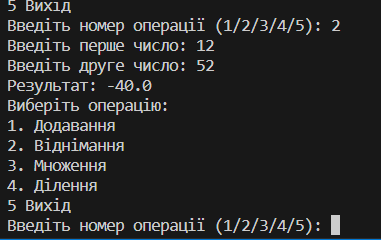
Оператор match є нововведенням в Python 3.10 і призначений для виконання збігу зі значеннями та виконання певних операцій на основі зіставлення значень з можливими варіантами. Основна властивість match полягає в тому, що він дозволяє вам визначати різні гілки зіставлення та виконувати код, пов'язаний з відповідним гілкам.





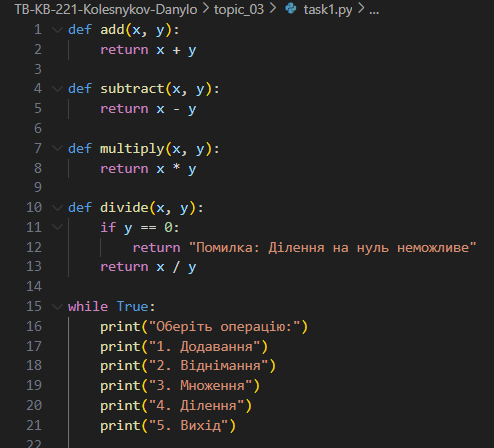
Спочатку я створив 4 функції, кожна з яких приймала дві змінні, і робила ту чи іншу математичну дію. Створив цикл, в якому вивід 5 дій, + - \* / та вихід. Створив змінну в яку потрібно вписвти номер дії. Після чого створив оператор match якому передав дані зі змінної choice. Після чого створив 5 кейсів, кожен з яких знову ж відповідав за якусь дію. Але в них я тільки створив змінні куди потрібно вписати числа, після чого викликав потрібну функцію та передав їх значення з змінних, які створив всередині цього кейсу. І якщо спрацьовує кейс 5 то за допомогою команди break виходжу з циклу.

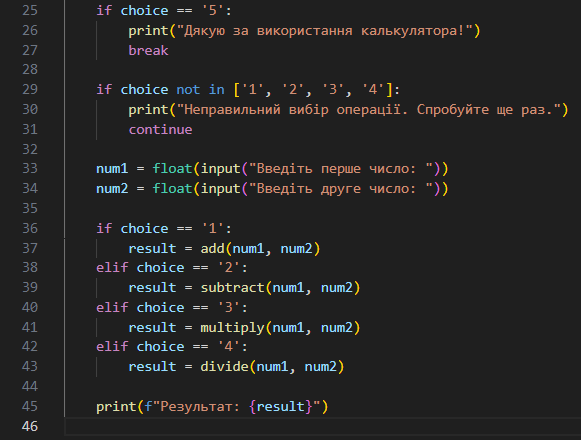
Результат:



**Практичне завдання до теми 3**

1. **Написати програму калькулятор з постійними запитами на введення нових даних та операцій.**





Спочатку я створив 4 функції, кожна з яких приймає два параметри та виконує ту чи іншу математичну дію. Після чого створив цикл в якому вивів всі доступні команди, створив змінну для того аби туди вписати номер потрібної нам команди. Після чого створив дві змінні які приймають два значення, це наші числа над якими потрібно зробити математичну дію. Після чого за допомогою логічних операторів if elif else зробив логіку, яка перенаправить нас на потрібну нам дію. І якщо ми не обираємо дію 5, то команда break не працює і ми маємо нескінченний калькулятор.

1. **Тестування списків**

Команди які я використовував в цьому завданні:

List.sort() – сортує список від найменшого до найбільшого

List.reverse() – перевертає список

List.append() – додає до списку значення в дужках

List.extend() – Розширює список

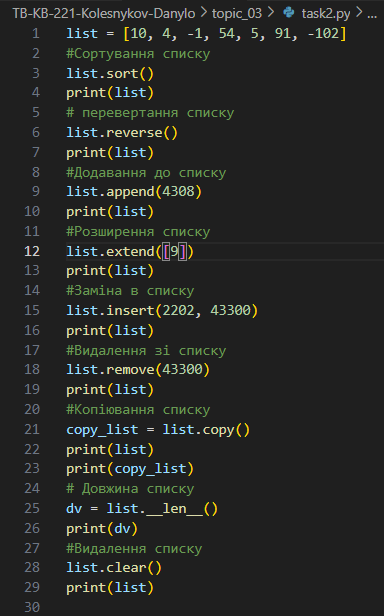
List.insert() – замінює одне значення на інше

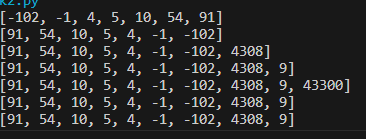
List.remove() – Видаляє значення в дужках

List,copy() – Копіює список у змінну

List.\_len\_() – рахує довжину списку

List.clear() – Повністю очищає список.



Результат: 

1. **Функції словника**

Функції які я використовував наведені нижче:

Assert значення in словник – перевіряє чи є таке значення в словнику

Assert словник[значення] == ключ – перевіряє чи такий ключ в значенні.

del my\_dict['job'] – видаляє значення в словнику

assert 'job' not in my\_dict – перевіряє чи немає такого значення в словнику

keys = list(my\_dict.keys())

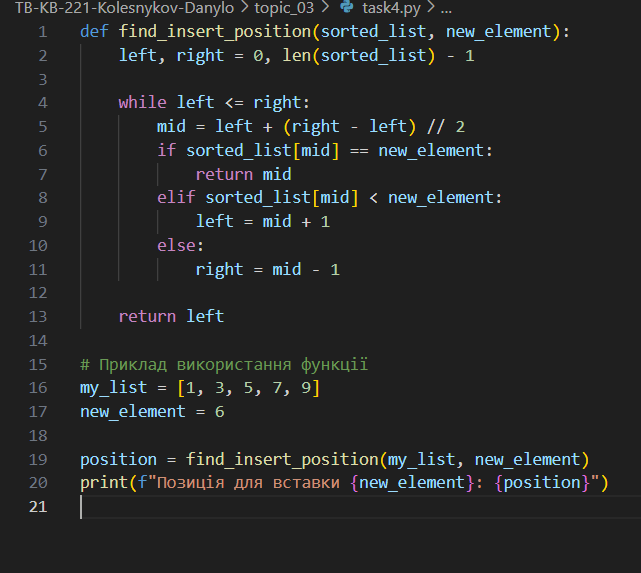
assert keys == ['name', 'age', 'city', 'country', 'email'] – перевірка значень або ключів в словнику

items = list(my\_dict.items()) assert items == [('name', 'John'), ('age', 31), ('city', 'New York'), ('country', 'USA'), ('email', 'john@example.com')] - Перевірка отримання пар ключ-значення у вигляді кортежів





1. **Написати функцію пошуку позиції для вставки нового елементу у відсортований список.**



У цій функції я використовував бінарний пошук для ефективного знаходження позиції вставки. Вона приймає відсортований список sorted\_list і новий елемент new\_element. Функція повертає позицію, на яку слід вставити new\_element, таким чином, щоб список залишився впорядкованим. У прикладі вище ми шукаємо позицію для вставки числа 6 до відсортованого списку my\_list і виводимо знайдену позицію.

Результат:

