**Звіт про виконання практичних завдань до лекцій з курсу Технології програмування на мові Python**

[Звіт до Теми №1 1](#_Toc1120961891)

[Перетворення рядка 2](#_Toc1739168984)

[Суть завдання: 2](#_Toc588206823)

[Хід виконання завдання: 2](#_Toc1706349503)

[Код завдання: 2](#_Toc1138980872)

[GitHub: 2](#_Toc63306627)

[Перевірка методів роботи зі строками 3](#_Toc1995492742)

[Суть завдання: 3](#_Toc1276687073)

[Хід виконання завдання: 3](#_Toc485371730)

[Код завдання: 3](#_Toc782348976)

[GitHub: 4](#_Toc1895072739)

[Функція знаходження дискримінанту 4](#_Toc250099506)

[Суть завдання: 4](#_Toc869270632)

[Хід виконання завдання: 5](#_Toc925736184)

[Код завдання: 5](#_Toc735818977)

[GitHub: 5](#_Toc667983672)

[Звіт до Теми №2 6](#_Toc1839590931)

[Корені квадратного рівняння 7](#_Toc1892815523)

[Суть завдання: 7](#_Toc1017243749)

[Хід виконання завдання: 7](#_Toc452936400)

[Код завдання: 8](#_Toc1434754639)

[GitHub: 8](#_Toc2012012569)

[Калькулятор if else 9](#_Toc1039084526)

[Суть завдання: 9](#_Toc1929447739)

[Хід виконання завдання: 9](#_Toc237066864)

[Код завдання: 10](#_Toc1964421748)

[GitHub: 10](#_Toc293872954)

[Калькулятор match 11](#_Toc534118111)

[Суть завдання: 11](#_Toc1476446686)

[Хід виконання завдання: 11](#_Toc1444490322)

[Код завдання: 12](#_Toc867149195)

[GitHub: 13](#_Toc1336570625)

[Звіт до Теми №3 15](#_Toc1878796017)

[Калькулятор із використанням циклу 15](#_Toc507876985)

[Суть завдання: 15](#_Toc1100641410)

[Хід виконання завдання: 15](#_Toc1943939838)

[Код завдання: 16](#_Toc28197936)

[GitHub: 16](#_Toc2048044753)

[Тестування функцій списків 17](#_Toc2026802398)

[Суть завдання: 17](#_Toc1350462358)

[Хід виконання завдання: 17](#_Toc1150338987)

[Код завдання: 17](#_Toc699102013)

[GitHub: 18](#_Toc1294923917)

[Тестування функцій словників 19](#_Toc641797906)

[Суть завдання: 19](#_Toc355304741)

[Хід виконання завдання: 19](#_Toc1998201164)

[Код завдання: 20](#_Toc388114738)

[GitHub: 20](#_Toc85038680)

[Пошук позиції у відсортованому списку 21](#_Toc1928644012)

[Суть завдання: 21](#_Toc1964686984)

[Хід виконання завдання: 21](#_Toc493033413)

[Код завдання: 22](#_Toc1749413249)

[GitHub: 22](#_Toc1838685457)

[Звіт до Теми №4 23](#_Toc1830261448)

[Розширений калькулятор 23](#_Toc674064328)

[Суть завдання: 23](#_Toc2028879889)

[Хід виконання завдання: 23](#_Toc1451325103)

[Код завдання: 24](#_Toc264422153)

[GitHub: 25](#_Toc247953738)

[Ділення на нуль 26](#_Toc1721239934)

[Суть завдання: 26](#_Toc562615313)

[Хід виконання завдання: 26](#_Toc1201657059)

[Код завдання: 26](#_Toc2108097109)

[GitHub: 27](#_Toc75248634)

[Виняткові ситуації 28](#_Toc2088084373)

[Суть завдання: 28](#_Toc1168000540)

[Хід виконання завдання: 28](#_Toc1911088035)

[Код завдання: 28](#_Toc480039744)

[GitHub: 29](#_Toc1562486612)

[Звіт до Теми №5 31](#_Toc1747477109)

[Гра з комп’ютером 31](#_Toc1022381038)

[Суть завдання: 31](#_Toc708822048)

[Хід виконання завдання: 32](#_Toc2022969279)

[Код завдання: 33](#_Toc612442859)

[GitHub: 34](#_Toc1601016652)

[Програма конвертування 35](#_Toc1505607228)

[Суть завдання: 35](#_Toc348462059)

[Хід виконання завдання: 35](#_Toc611668482)

[Код завдання: 36](#_Toc1098056829)

[GitHub: 36](#_Toc428509014)

[Програма калькулятор 36](#_Toc2051012127)

[Суть завдання: 36](#_Toc750125255)

[Хід виконання завдання: 36](#_Toc2107174692)

[Код завдання: 36](#_Toc1453993168)

[GitHub: 36](#_Toc1812447389)

# **Звіт до Теми №1**

**Функції та змінні**

Під час виконання практичного завдання до Теми №1 було надано варіанти рішення до наступних задач:

## Перетворення рядка

### **Суть завдання:**

Необхідно рядок, що має вигляд "abcdefg123" перетворити наступним чином "321gfedcba", вважаючи сталою довжину рядку в 10 символів.

### **Хід виконання завдання:**

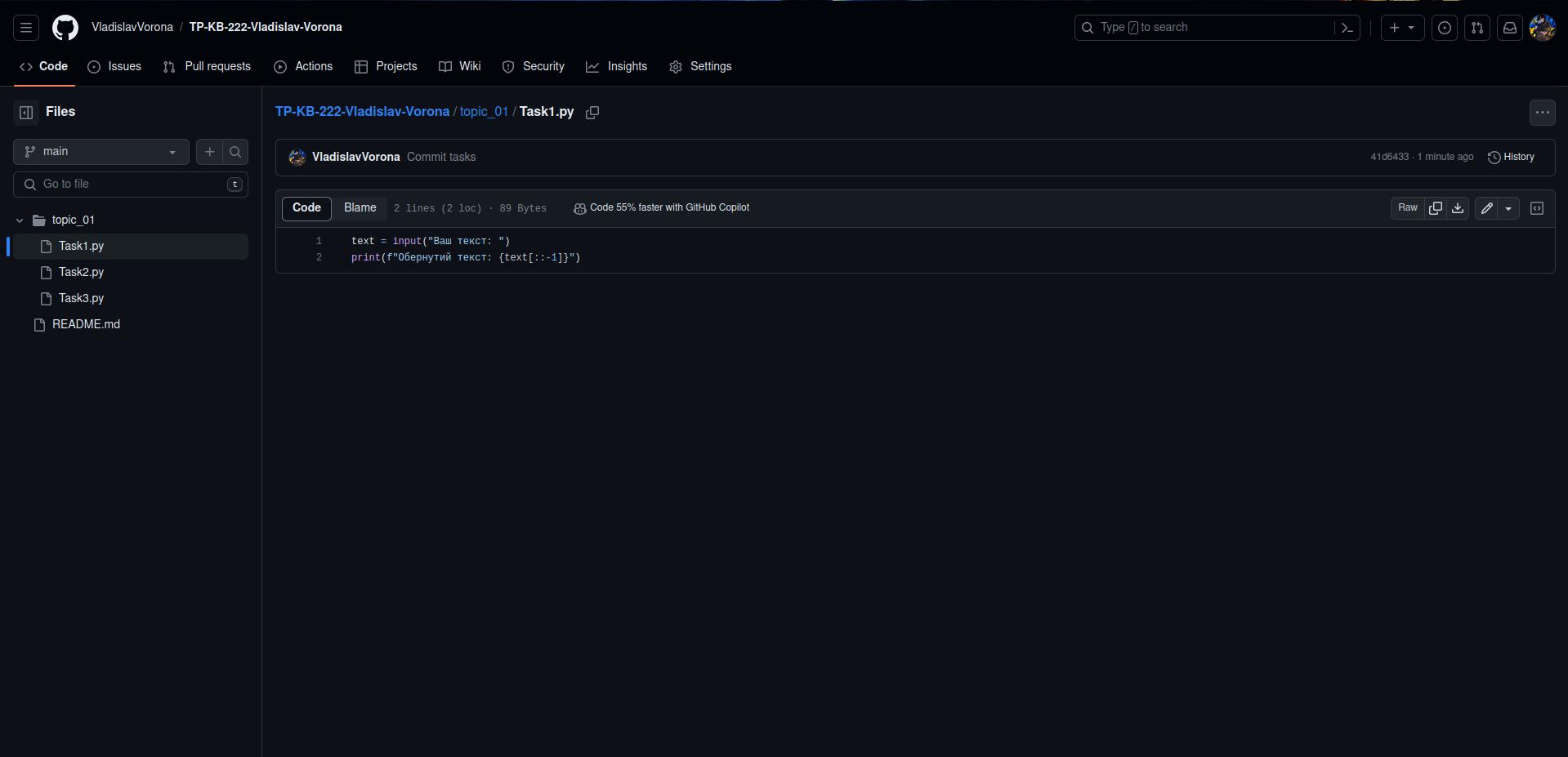
Для виконання цього завдання я використав input() для отримання тексту від користувача і потім записав цей текст в змінну text, після чого використав print() із форматування тексту для зручного відображення інформації. Для обернення рядка я використав оператор [::-1], який створює зріз рядка та включає всі символи в зворотньому порядку.

### **Код завдання:**



### **GitHub:**

Посилання на GitHub: https://github.com/VladislavVorona/TP-KB-222-Vladislav-Vorona/blob/main/topic\_01/Task1.py  
 Знімок екрану з GitHub:



## Перевірка методів роботи зі строками

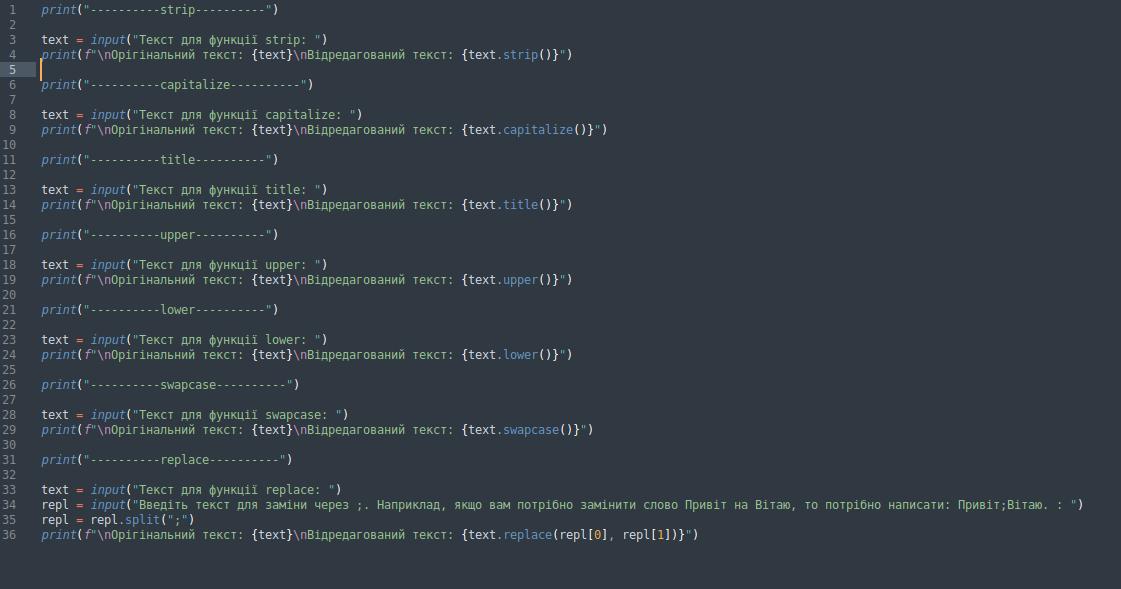
**Суть завдання:**

Необхідно перевірити роботу strip() , capitalize(), title(), upper(), lower(), swapcase(), replace().

### **Хід виконання завдання:**

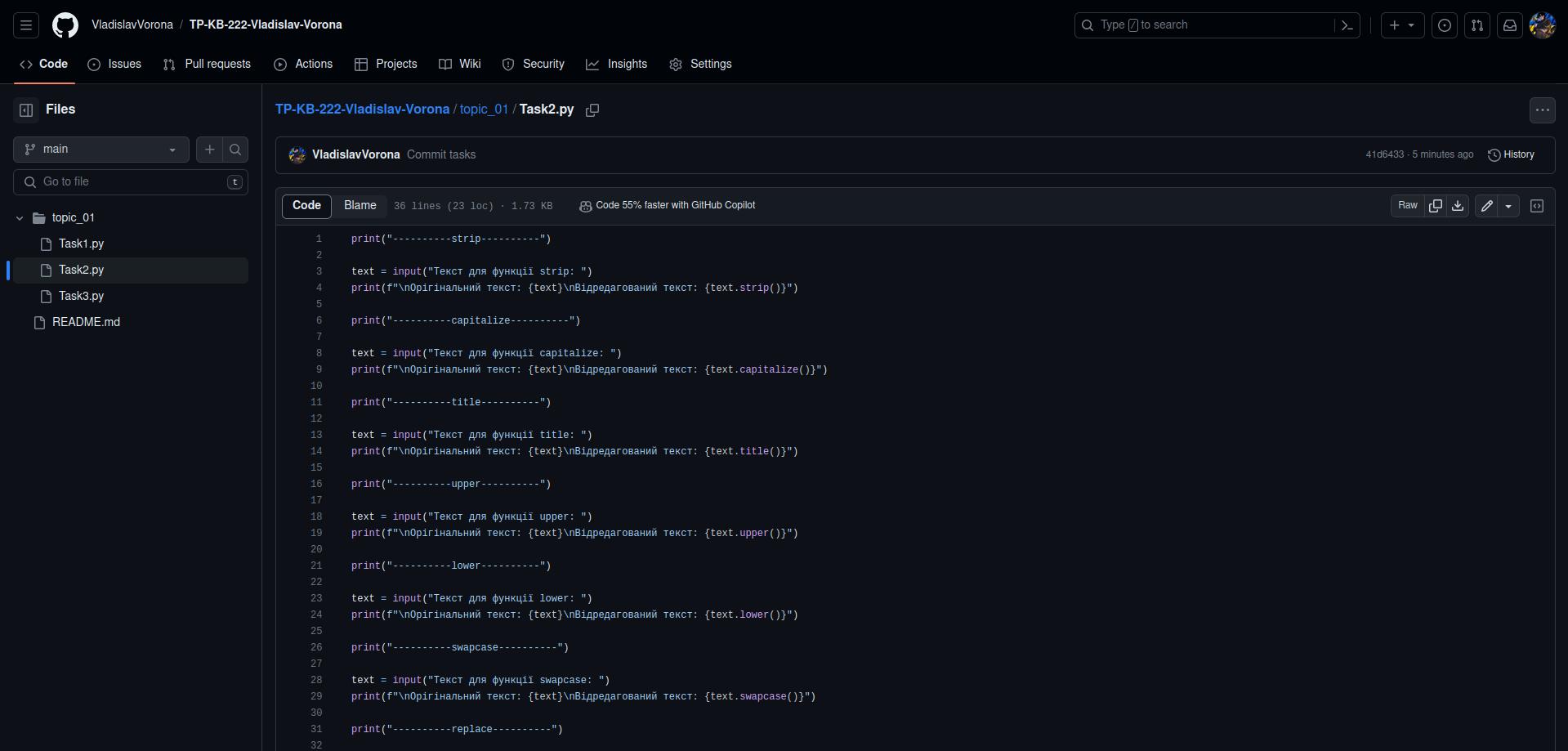
Для виконання цього завдання я використав input() для отримання тексту від користувача і потім записав цей текст в змінну text, після чого використав print() із форматування тексту та перенесенням рядку для зручного відображення інформації. В кожному print() використовується певний метод роботи зі строками, загалом було використано такі методи, як: strip() , capitalize(), title(), upper(), lower(), swapcase(), replace(). Також для демонстрації метода replace(), я використав split() для поділу отриманого тексту від користувача на елементи масиву.

### **Код завдання:**



### **GitHub:**

Посилання на GitHub: https://github.com/VladislavVorona/TP-KB-222-Vladislav-Vorona/blob/main/topic\_01/Task2.py  
 Знімок екрану з GitHub:



## Функція знаходження дискримінанту

**Суть завдання:**

Необхідно написати функцію яка приймає три параметри a,b,c та повертає значення дискримінанту.

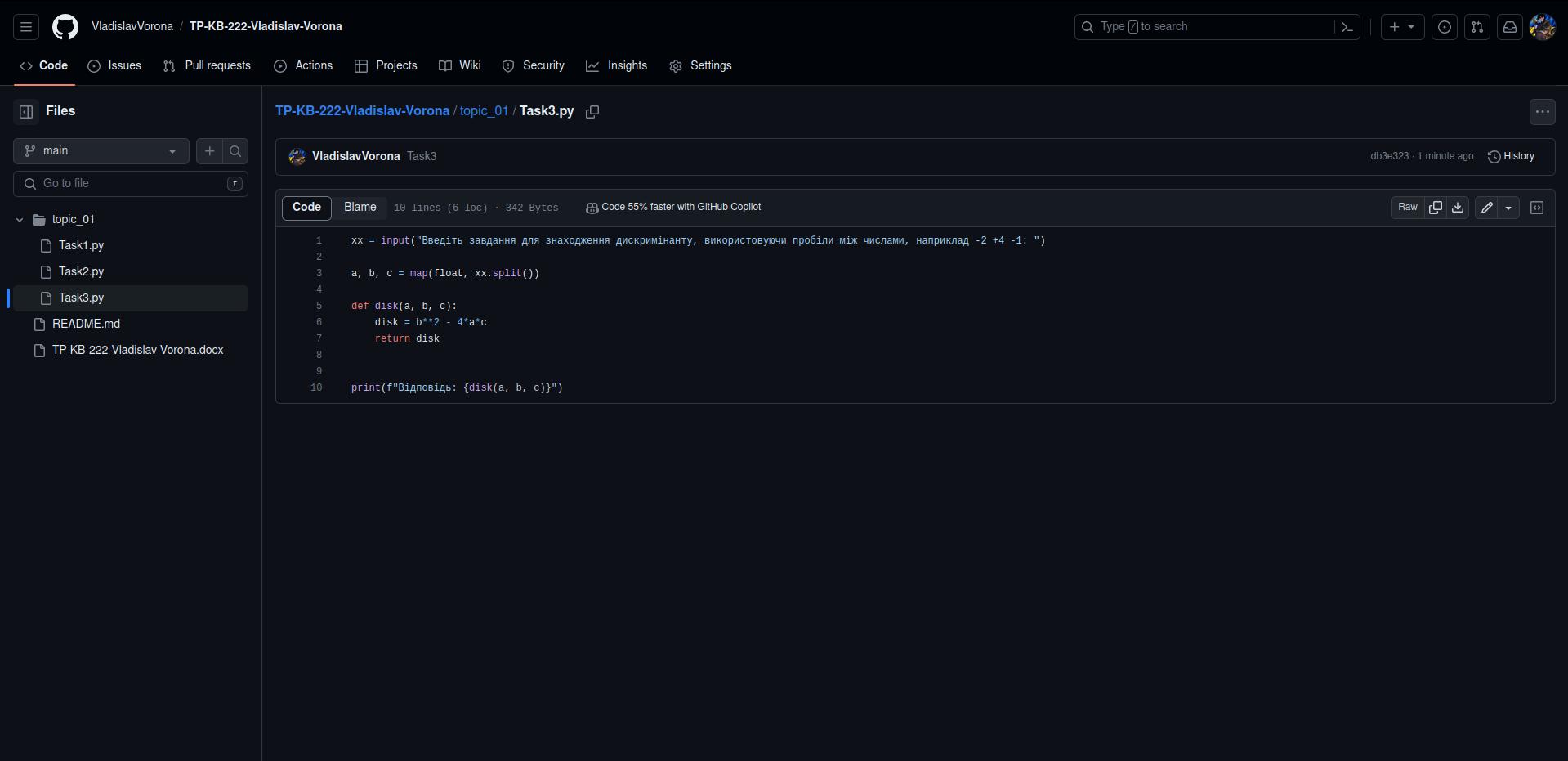
### **Хід виконання завдання:**

Для виконання цього завдання я використав input() для отримання тексту від користувача і потім записав цей текст в змінну xx, після чого я розділив отриманний текст на елементи масиву за допомогою split() та перевів всі значення в float за допомогою функції map() та типізатора float, після чого використав print() із форматуванням тексту для зручного відображення інформації. В середині print(), за допомогою форматування, викликав функцію disk та передав їй значення a, b, c, після чого виконав арифметичні дії, які потрібні для знаходження дискримінанту, використовуючи таку формулу: b^2-4ac, в середині функції, та повернув відповідь за допомогою return.

### **Код завдання:**

### **GitHub:**

Посилання на GitHub: https://github.com/VladislavVorona/TP-KB-222-Vladislav-Vorona/blob/main/topic\_01/Task3.py  
 Знімок екрану з GitHub:



# **Звіт до Теми №2**

**Умовний перехід**

Під час виконання практичного завдання до Теми №2 було надано варіанти рішення до наступних задач:

## Корені квадратного рівняння

### **Суть завдання:**

Потрібно написати функції пошуку коренів квадратного рівняння використовуючи функцію розрахунку дискримінанту з попередньої теми та умовні переходи

### **Хід виконання завдання:**

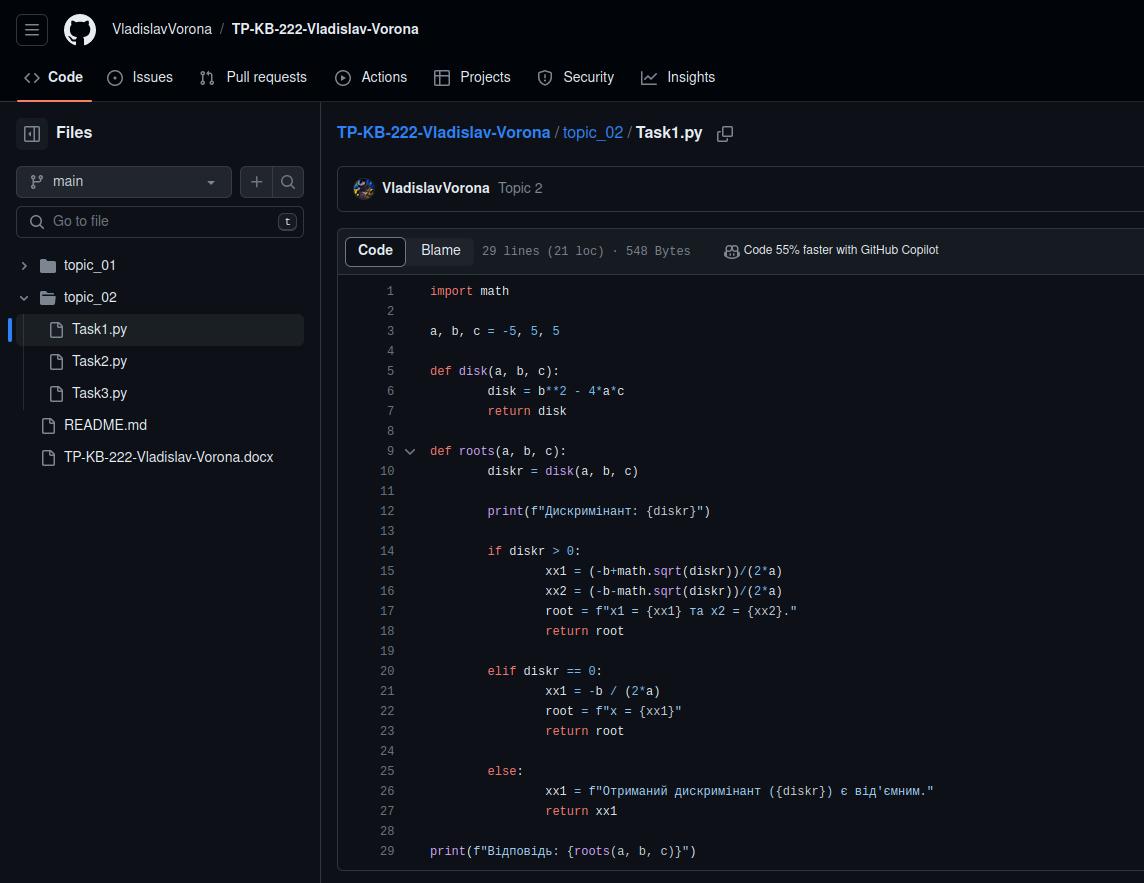
Для виконання цього завдання я імпортував модуль math, який додає можливість використовувати математичні функції, після чого створив змінні та заніс в їх числа. Потім я скопіював функцію розрахунку розрахування дискримінанту із минулої теми і додав її до коду, після чого почав написання функції для пошуку коренів, ця функція отримує змінні, які я вказав раніше, та викликає функцію розрахування дискримінанту та передає їй ці змінні, після чого заносить отриманий результат у змінну diskr, потім виводить за допомогою print() отриманий дискримінант, після чого я використав умову “if diskr > 0:”, якщо ця умова справджується, то починається розрахування коренів за формулою x = (-b±√D)/(2\*a), для розрахунків використовується функція модуля math - sqrt, яка розраховує корінь від числа. Після проведення розрахунків корінь заноситься в змінну і починається розрахунок другого корня, після чого я створив змінну із форматуванням тексту для гарного відображення інформації, після чого повернув цю змінну і за допомогою print() і форматування тексту, вивів корені. Якщо виконується умова “elif diskr == 0:”, то корінь вираховується за формулою -b/(2\*a) та заноситься в змінну, після чого я створив змінну із форматуванням тексту і повернув цю змінну в print() із форматуванням. Якщо жодна із попередніх умов не виконується, то виконується код, який повертає текст, де вказано, що отриманий дискримінант є від’ємним і не може бути розрахованим, після чого відобразив це за допомогою print() і форматування.

### **Код завдання:**



### **GitHub:**

Посилання на GitHub: https://github.com/VladislavVorona/TP-KB-222-Vladislav-Vorona/blob/main/topic\_02/Task1.py  
 Знімок екрану з GitHub:



## Калькулятор if else

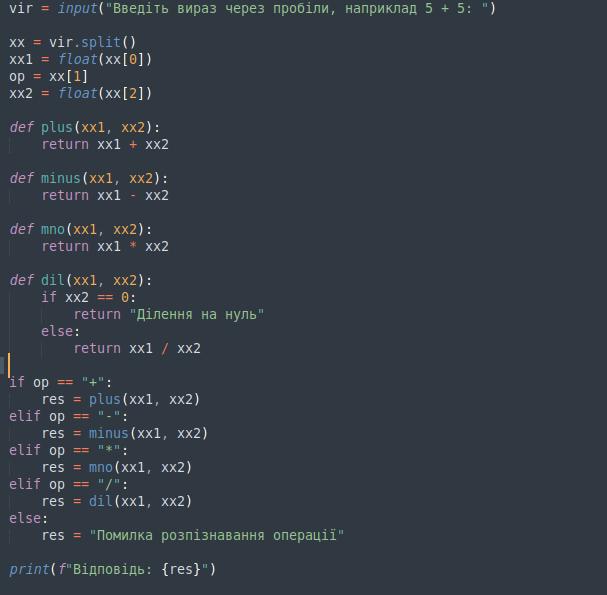
**Суть завдання:**

Потрібно написати програму калькулятор використовуючи **if else** конструкцію.

### **Хід виконання завдання:**

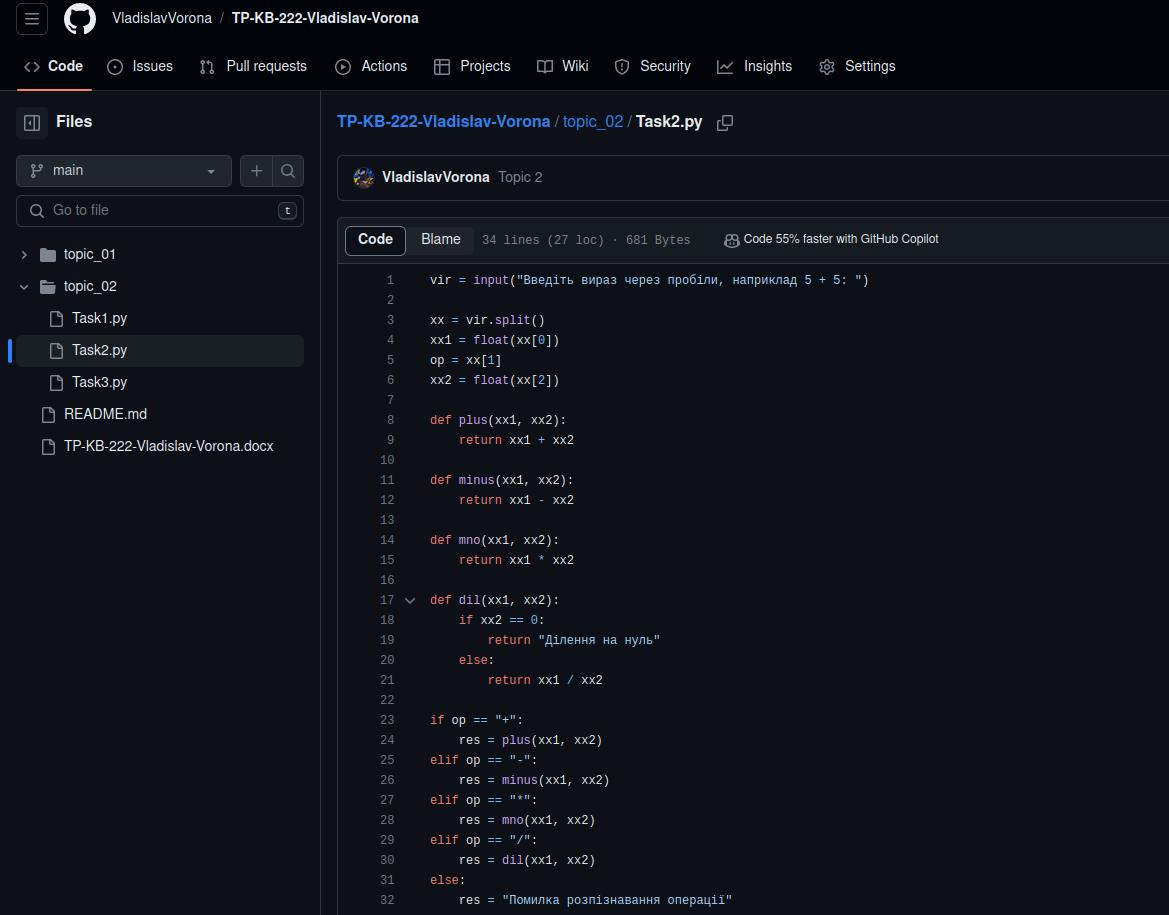
Для виконання цього завдання я спочатку створив змінну та використав input() для отримання завдання від користувача, після чого я отриманий текст перетворив в список значень за допомогою split() та заніс цей список в змінну, після чого почергово створив змінні, дві змінні для значень та одну змінну для операції, після чого заніс в ці змінні потрібну інформацію із списку і форматував текст в float, після чого виконується перевірка на тип операції, якщо змінна із операцією дорівнює "+", то виконується функція із додаванням змінних і після чого результат повертається та зберігається в змінну, теж саме відбувається для інших можливих операцій, таких як віднімання, множення та ділення. Також, якщо виконується операція ділення, то перевіряється чи дорівняє дільник нулю, якщо дільник дорівнює нулю, то в змінну записується текст, що відбулося ділення на ноль, якщо ні, то виконується ділення. Якщо було неправильно вказана операція, то в змінну із результатом записується інформація, що відбулася помилка розпізнавання операції. В кінці я використав print() із форматуванням для відображення результату.

### **Код завдання:**



### **GitHub:**

Посилання на GitHub: https://github.com/VladislavVorona/TP-KB-222-Vladislav-Vorona/blob/main/topic\_02/Task2.py  
 Знімок екрану з GitHub:



## Калькулятор match

**Суть завдання:**

Потрібно написати програму калькулятор використовуючи **match** конструкцію.

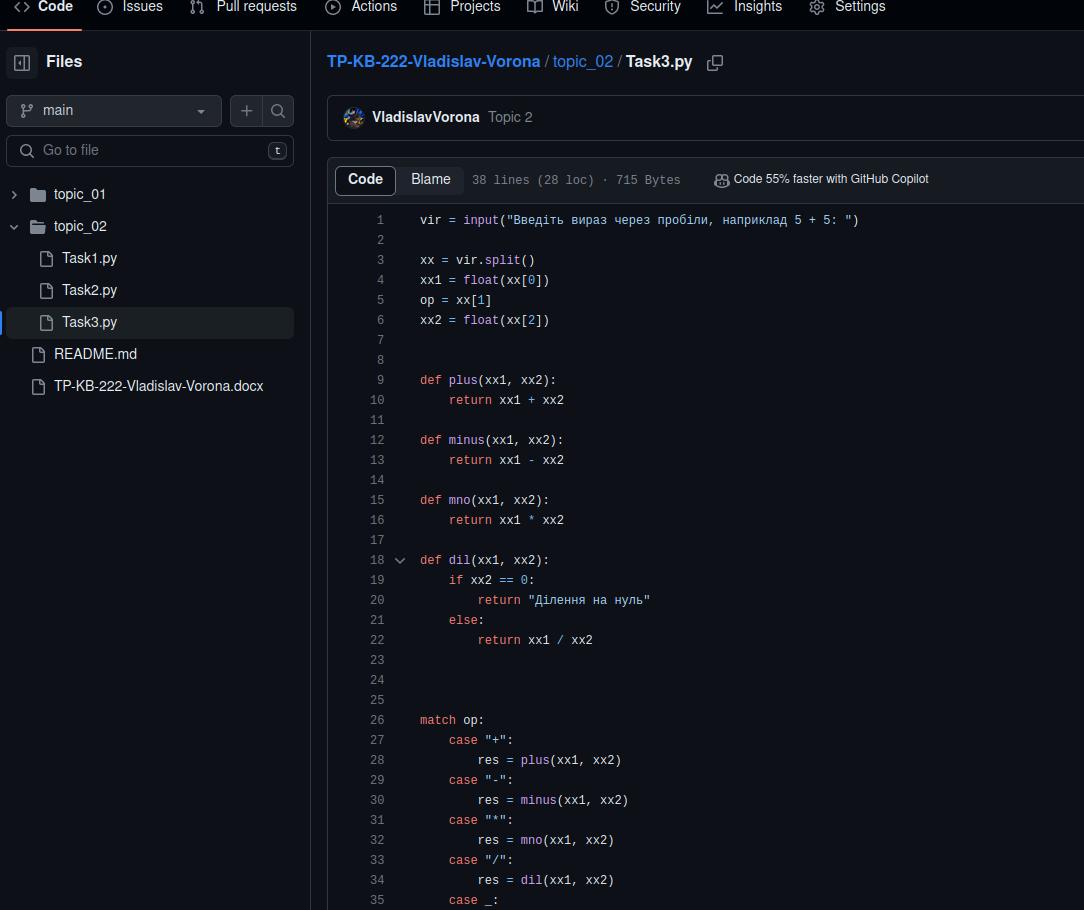
### **Хід виконання завдання:**

Для виконання цього завдання я спочатку створив змінну та використав input() для отримання завдання від користувача, після чого я отриманий текст перетворив в список значень за допомогою split() та заніс цей список в змінну, після чого почергово створив змінні, дві змінні для значень та одну змінну для операції, після чого заніс в ці змінні потрібну інформацію із списку і форматував текст в float. Далі я використав конструкцію match, яка дозволяє перевіряти значення змінної та виконувати відповідну операцію в залежності від значення змінної, наприклад, якщо змінна дорівнює "+", то виконується функція із додаванням змінних і після чого результат повертається та зберігається в змінну, теж саме відбувається для інших можливих операцій, таких як віднімання, множення та ділення. Також, якщо виконується операція ділення, то перевіряється чи дорівняє дільник нулю, якщо дільник дорівнює нулю, то в змінну записується текст, що відбулося ділення на ноль, якщо змінна не є нулем, то виконується ділення. Якщо було була вказана невідома операція, то в змінну із результатом записується інформація, що відбулася помилка розпізнавання операції. В кінці я використав print() із форматуванням для відображення результату.

### **Код завдання:**

### **GitHub:**

Посилання на GitHub: https://github.com/VladislavVorona/TP-KB-222-Vladislav-Vorona/blob/main/topic\_02/Task3.py  
 Знімок екрану з GitHub:



# **Звіт до Теми №3**

**Умовний перехід**

Під час виконання практичного завдання до Теми №3 було надано варіанти рішення до наступних задач:

## Калькулятор із використанням циклу

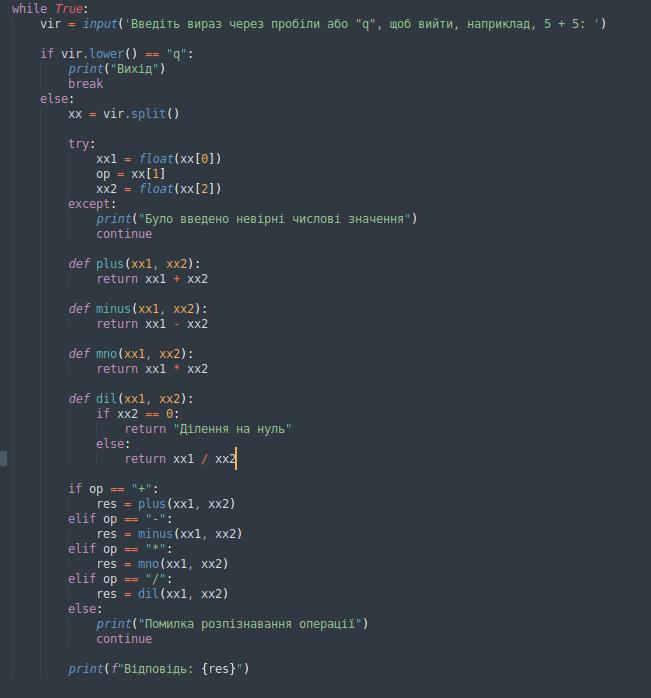
### **Суть завдання:**

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.

### **Хід виконання завдання:**

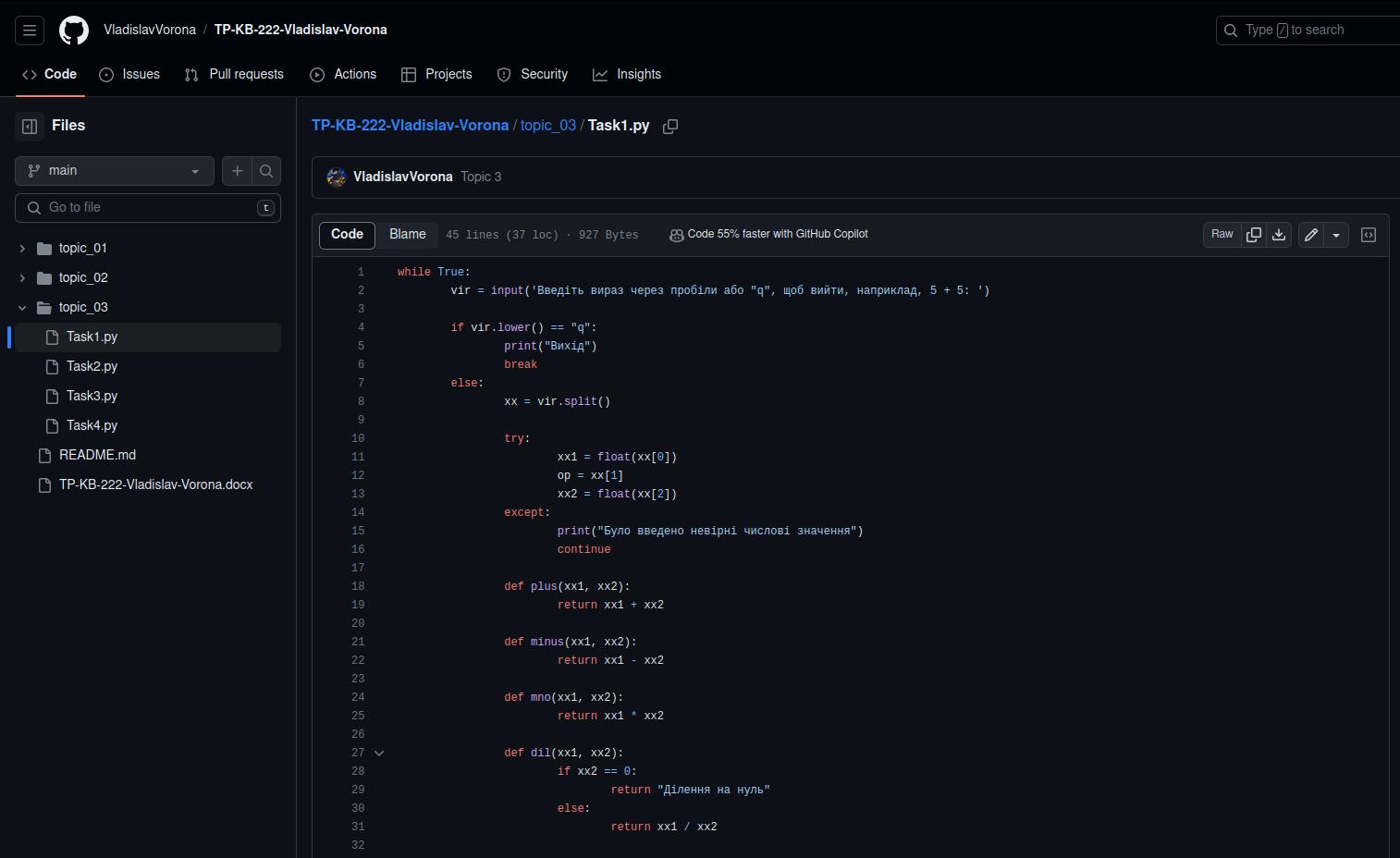
Для виконання цього завдання я скопіював код калькулятора із минулого завдання із заніс його в вічний цикл “while True:”, після чого додав перевірку змінної, в яку користувач вводить інформацію. Ця перевірка отримує змінну відредаговану методом lower(), який переводить увесь текст в нижній регістр, це було зроблено для зручності виходу. Сама умова перевіряє чи ввів користувач “q” і якщо умова справджується, то виводиться надпис “Вихід” і виконання циклу припиняється, що веде за собою закінчення роботи програми. Якщо ця умова не справджується, то далі проходить виконання коду калькулятора. Також я додав перевірку на введення вірних числових значень, вона побудована на блоках “try” та “except”. В середині блоку “try” виконується форматування елементів списку в float і якщо під час такого форматування відбудеться помилка, то активується блок “except”, де буде виведено інформацію, що було введено невірні числові значення і після чого цикл “while True” почнеться спочатку, теж саме відбудеться, якщо буде невірно вказана операція, тільки буде виведено надпис про помилку розпізнавання операції. Якщо все було вказано вірно, то після розрахунку виразу користувача, виконання циклу “while True” має початися спочатку.

### **Код завдання:**



### **GitHub:**

Посилання на GitHub: https://github.com/VladislavVorona/TP-KB-222-Vladislav-Vorona/blob/main/topic\_03/Task1.py  
 Знімок екрану з GitHub:



## Тестування функцій списків

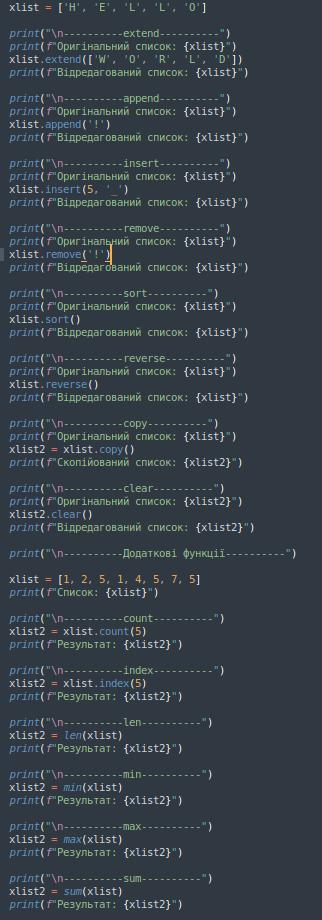
**Суть завдання:**

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

### **Хід виконання завдання:**

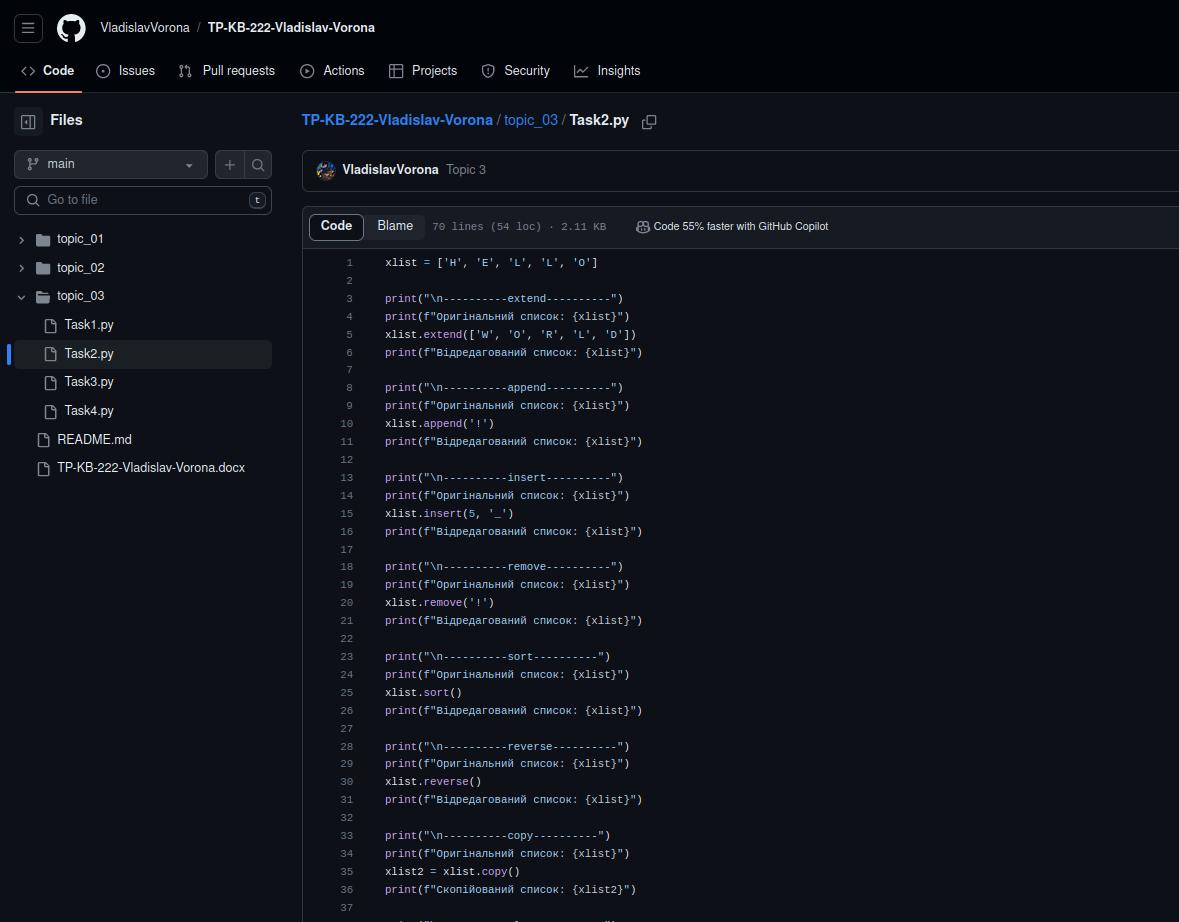
Для виконання цього завдання я спочатку створив список xlist і заніс в його тестові значення, після чого почав почергово тестувати функції списків, таких як: extend(), append(), insert(id, val), remove(val), clear(), sort, reverse, copy() і почергово виводити результат їх тестування за допомогою print() із форматуванням. Після тестування я також додав додаткові функції, які були протестовані на новоствореному списку xlist, цей список я створив числовим, оскільки робота цих функцій буде краще продемонстрована саме на числовому списку. Додатково я тестував такі функції, як count(), index(), len(), min(), max(), sum(), а результат їх виконання я виводив за допомогою print() із форматуванням.

### **Код завдання:**



### **GitHub:**

Посилання на GitHub: https://github.com/VladislavVorona/TP-KB-222-Vladislav-Vorona/blob/main/topic\_03/Task2.py  
 Знімок екрану з GitHub:



## Тестування функцій словників

**Суть завдання:**

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

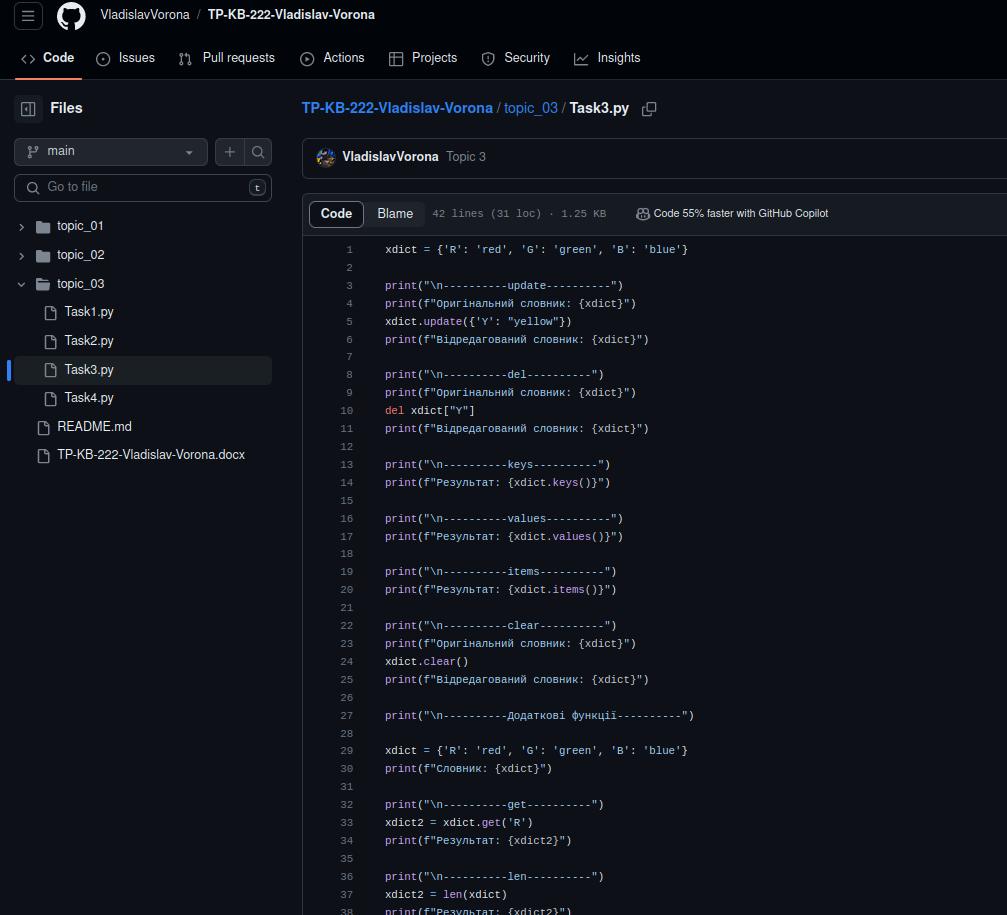
### **Хід виконання завдання:**

Для виконання цього завдання я спочатку створив словник і заніс в його тестові данні, після чого почав почергово тестувати на йому функції словників, таких як: update(), del(), clear(), keys(), values(), items() і після кожного тестування я виводив результат їхнього виконання за допомогою print() із форматуванням. Також я додатково розглянув такі функції, як: get(), len(), copy(). Для їх тестування було створено новий словник і в його було занесено тестові дані. Після тестування кожної із функцій я відображав результат їх роботи за допомогою print() із форматуванням.

### **Код завдання:**

### **GitHub:**

Посилання на GitHub: https://github.com/VladislavVorona/TP-KB-222-Vladislav-Vorona/blob/main/topic\_03/Task3.py  
 Знімок екрану з GitHub:



## Пошук позиції у відсортованому списку

**Суть завдання:**

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

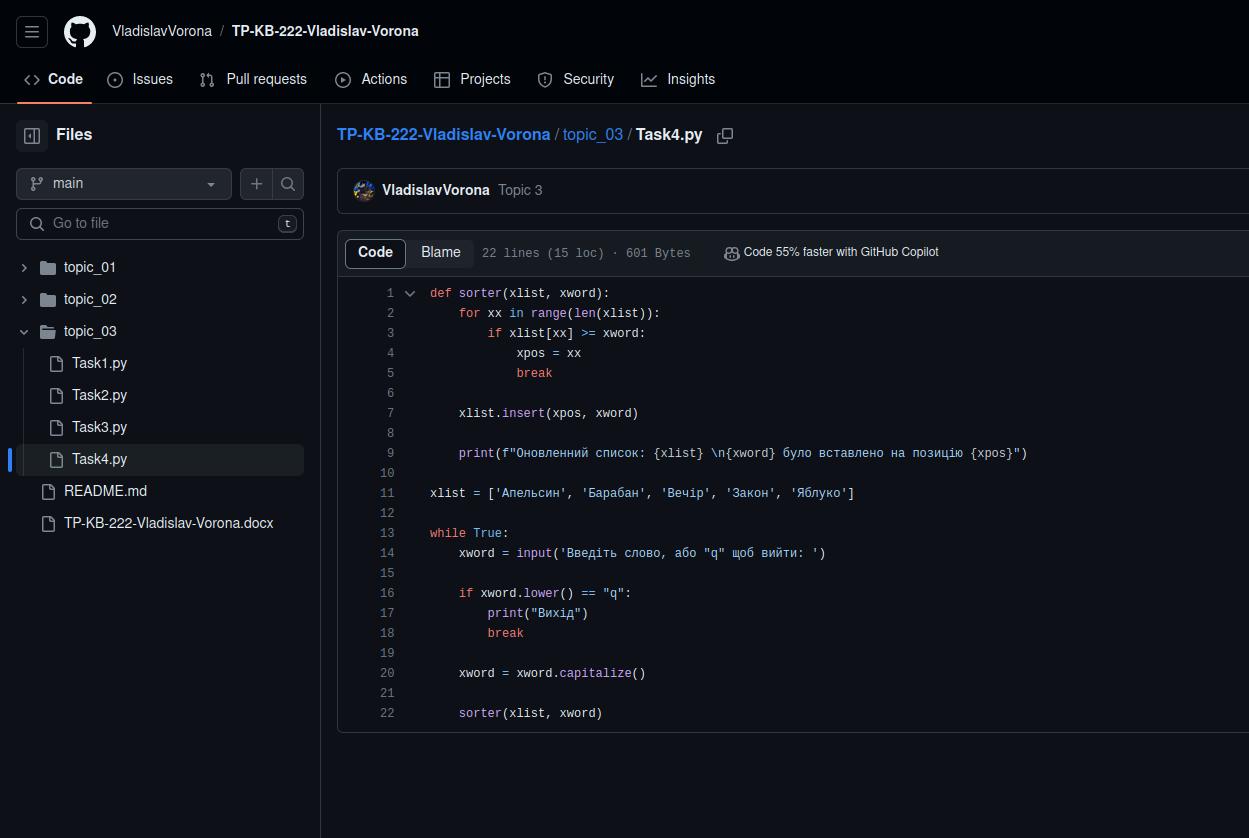
### **Хід виконання завдання:**

Для виконання цього завдання я створив список і заніс в його тестові данні, які були заздалегідь відсортовані, після чого я створив вічний цикл “while True:”, в якому я створив змінну xword в яку за допомогою input(), користувач може заносити значення. Після занесення значення в змінну проходить перевірка цієї змінної, яка була відредагована за допомогою методу lower(), який переводить увесь текст в нижній регістр. Якщо користувач написав “q”, то виконання циклу припиняється і тим самим закінчується виконанням програми. Якщо користувач написав не “q”, то далі відбувається перетворення змінної за допомогою методу capitalize(), який переводить першу букву в верхній регістр, а всі інші в нижній. Після збереження відредагованої змінної викликається функція “sorter”, в яку передається список та змінна із значенням користувача. В середині цієї функції виконується цикл “for”, який перебирає всі елементи у списку “xlist” і якщо цей елемент більший або рівний змінній, в яку заніс дані користувач, то зберігається індекс цього елемента в змінну “xpos” і після чого виконання циклу закінчується. Далі відбувається занесення змінної користувача в список по індексу, який ми занесли в змінну “xpos” і після чого відбувається виведення оновленого списку та індексу, куди було записано змінну користувача, за допомогою print() та форматування. Після виконання цих дій виконання коду циклу “while True” починається спочатку.

### **Код завдання:**

### **GitHub:**

Посилання на GitHub: https://github.com/VladislavVorona/TP-KB-222-Vladislav-Vorona/blob/main/topic\_03/Task4.py  
 Знімок екрану з GitHub:



# **Звіт до Теми №4**

**Умовний перехід**

Під час виконання практичного завдання до Теми №4 було надано варіанти рішення до наступних задач:

## Розширений калькулятор

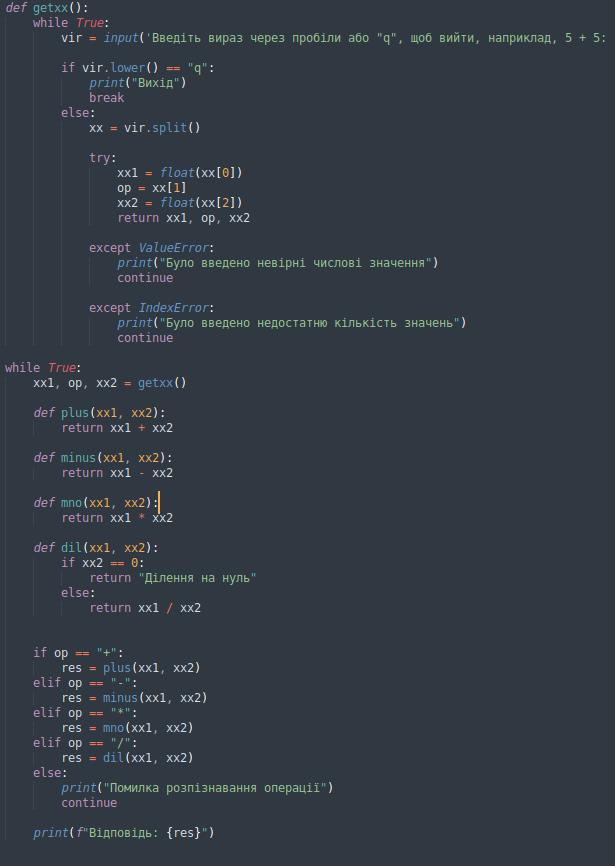
### **Суть завдання:**

Потрібно розширити програму калькулятор функцією запитів даних для виконання операцій від користувача, що обробляє виняткові ситуації.

### **Хід виконання завдання:**

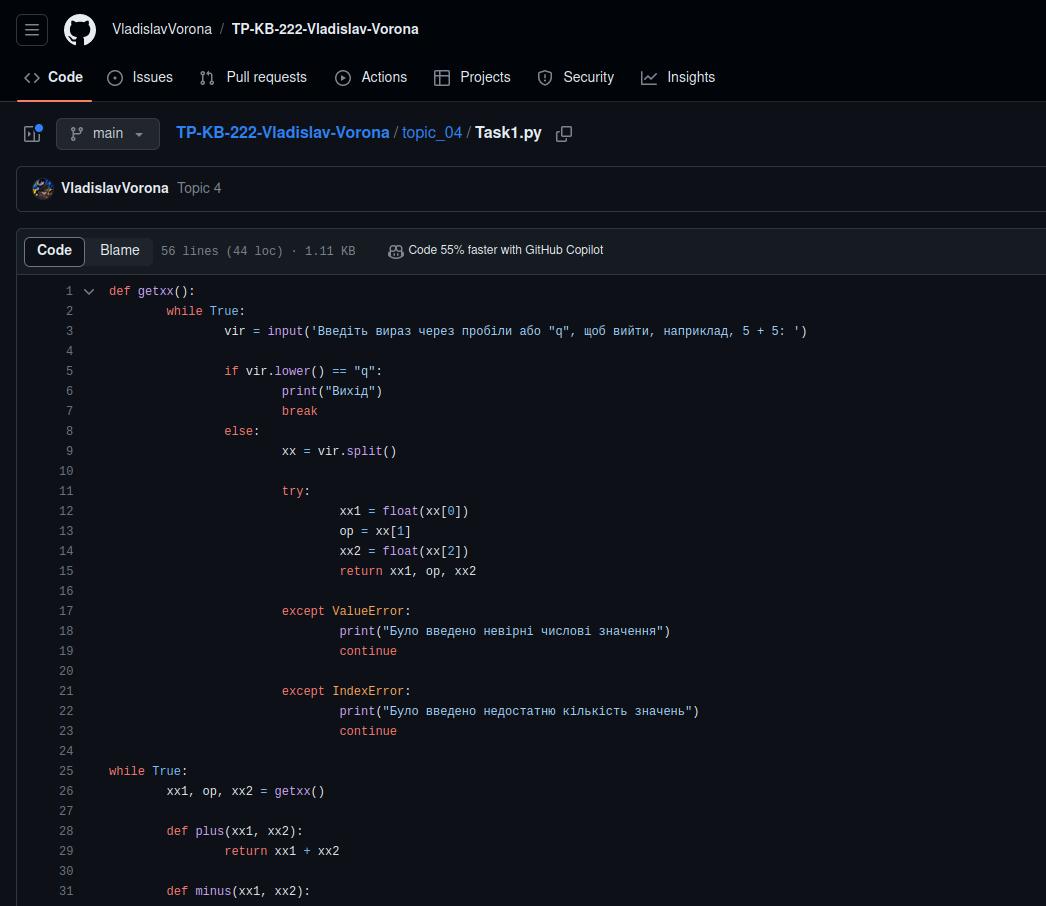
Для виконання цього завдання я скопіював код калькулятору із минулих завдань і створив функцію getxx, яка отримую та оброблю данні, які вводить користувач. Я перемістив код отримання та перетворення значень, які були введені користувачем, в цю функцію та додав такі виняткові ситуації, як ValueError та IndexError. ValueError було додано для слідкуванням за форматуванням значень, тобто, якщо код виводить помилку під час форматування значень користувача, то за допомогою цієї виняткової ситуації виводиться повідомлення, що було введено невірні значення і починає виконання циклу, із отримання і обробки данних користувача, із початку. IndexError було додано для слідкуванням за кількістю отриманих даних від користувача, якщо під час розподілення даних користувача по змінним не буде вистачить даних в списку, то виникне ця виняткова ситуація, яка повідомить користувача, що було введено недостатню кількість значень і почне виконання циклу, отримання та обробки даних від користувача, спочатку. Якщо все буде виконано правильно, то функція повертає 3 змінні, які одразу зберігаються в коді обрахунку завдання.

### **Код завдання:**



### **GitHub:**

Посилання на GitHub: https://github.com/VladislavVorona/TP-KB-222-Vladislav-Vorona/blob/main/topic\_04/Task1.py  
 Знімок екрану з GitHub:



## Ділення на нуль

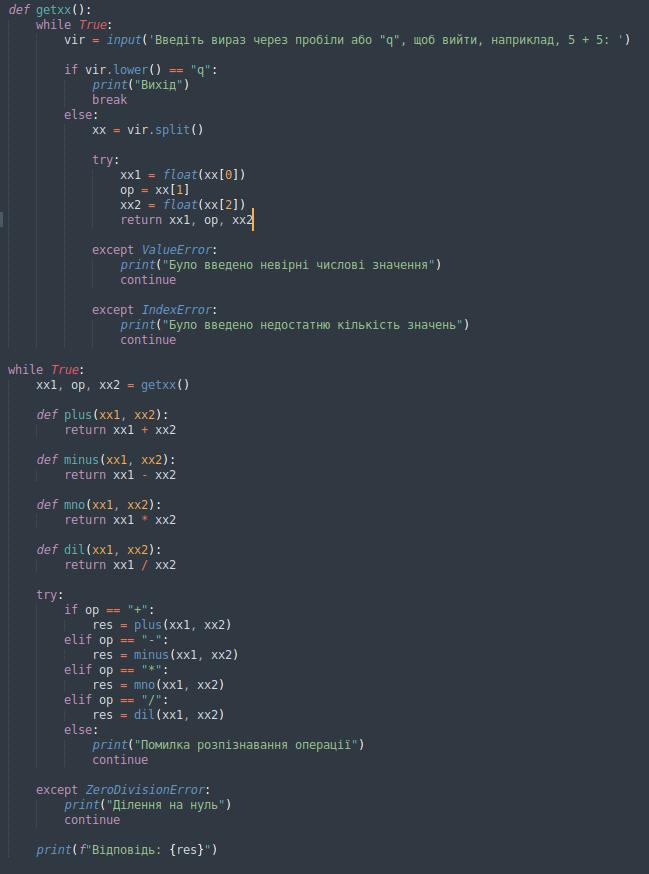
**Суть завдання:**

Потрібно розширити функціонал ділення обробкою виняткової ситуації ділення на нуль.

### **Хід виконання завдання:**

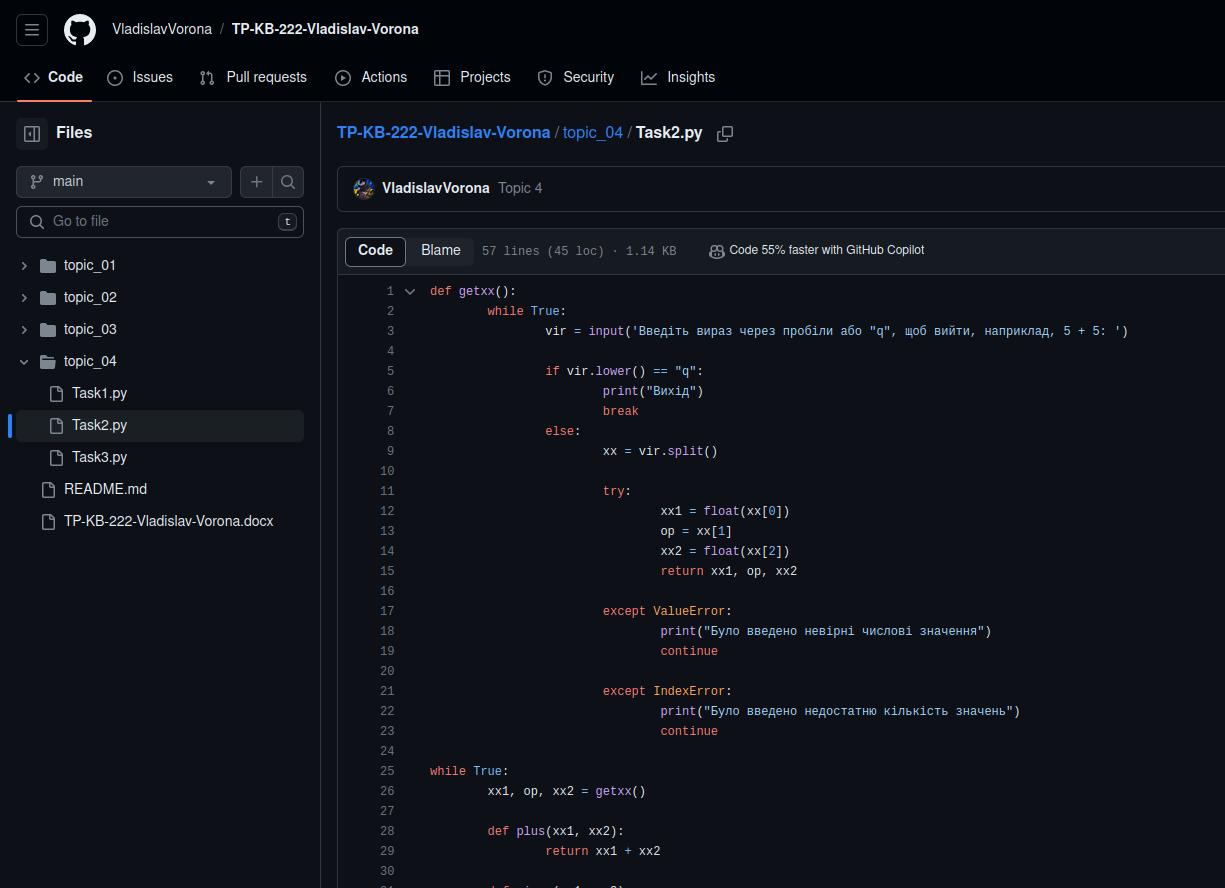
Для виконання цього завдання я скопіював код калькулятору із минулого завдання і додав виняткову подію ZeroDivissionError до коду, де виконується обрахунок завдання, яке було отримане від користувача. Саме ця виняткова подія стосується ділення, оскільки виникає при діленні на нуль. Якщо була спроба поділити на нуль, то виводиться помилка, де вказано, що відбулося ділення на нуль і після чого виконання всього циклу отримання, обробки і розрахунку починається спочатку.

### **Код завдання:**



### **GitHub:**

Посилання на GitHub: https://github.com/VladislavVorona/TP-KB-222-Vladislav-Vorona/blob/main/topic\_04/Task2.py  
 Знімок екрану з GitHub:



## Виняткові ситуації

**Суть завдання:**

Потрібно ознайомитись зі списком виняткових ситуацій за [посиланням](https://docs.python.org/3/library/exceptions.html).

### **Хід виконання завдання:**

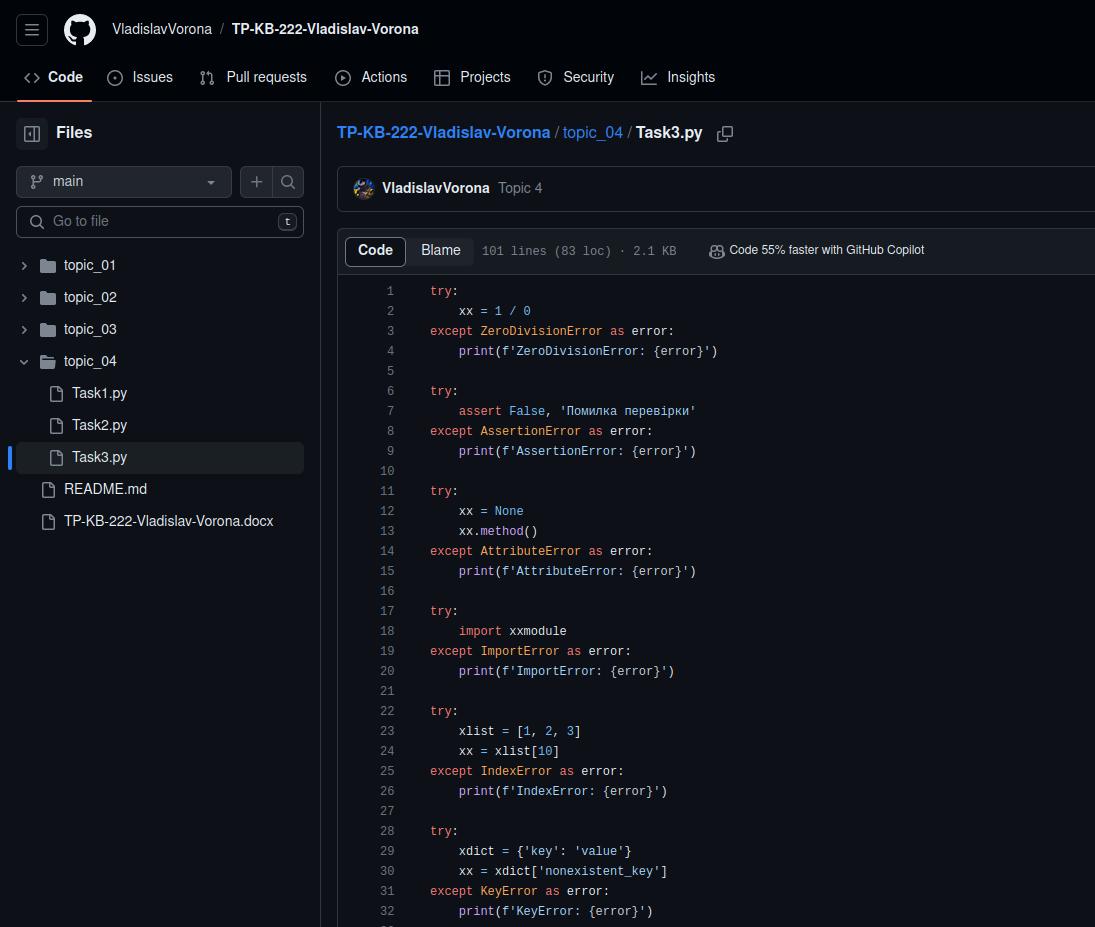
Для виконання цього завдання я перейшов по посиланню та ознайомився зі списком виняткових ситуацій, після чого захотів перевірити їх роботу. Я написав код, в якому почергово перевіряється кожна із виняткових функцій і після чого виводиться повідомлення із подробицями про помилку.

### **Код завдання:**

### 

### **GitHub:**

Посилання на GitHub: https://github.com/VladislavVorona/TP-KB-222-Vladislav-Vorona/blob/main/topic\_04/Task3.py  
 Знімок екрану з GitHub:



# **Звіт до Теми №5**

**Бібліотеки**

Під час виконання практичного завдання до Теми №5 було надано варіанти рішення до наступних задач:

## Гра з комп’ютером

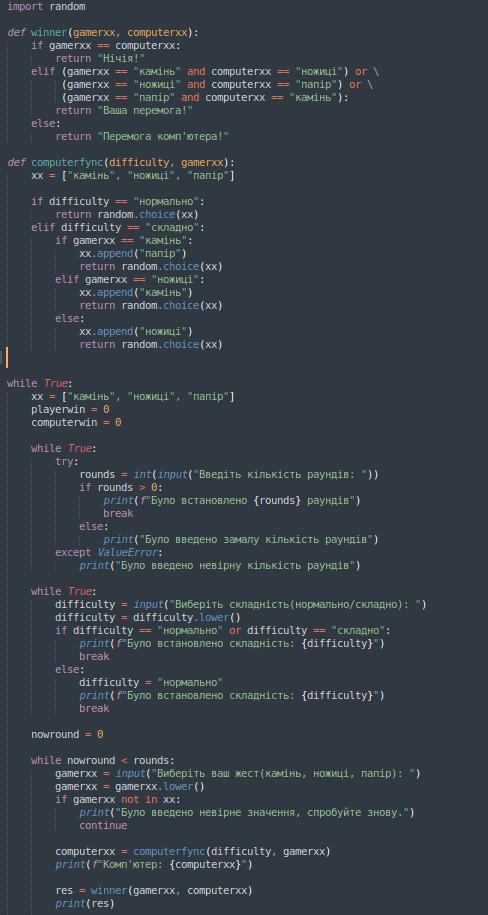
### **Суть завдання:**

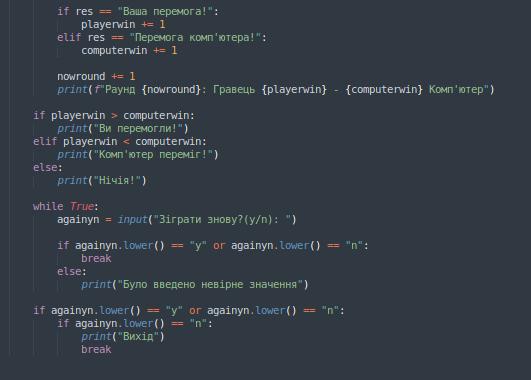
Гра з комп’ютером: камінь, ножиці, папір. Програма виконує запит від користувача на введення одного із значень ["rock", "scissor", "paper"]. Наступним кроком, використовуючи модуль random, програма у випадковому порядку вибирає одне із значень ["rock", "scissor", "paper"]. В залежності від умови, що камінь перемагає ножиці, ножиці перемагають папір, а папір перемагає камінь визначити переможця.

### **Хід виконання завдання:**

Для виконання цього завдання я імпортував модуль random та створив вічний цикл. В середині цього циклу створив список, в який помістив всі можливі жести для гри, після чого створив 2 змінні, які відповідають за кількість перемог комп’ютера та гравця. Потім я створив вічний цикл, який відповідає за введення користувачем кількості раундів для гри і відразу перевіряє правильність введених значень. Потім я створив цикл, який відповідає за введення користувачем рівня складності, я створив 2 рівні, нормальний та складний, якщо користувач пропускає цей вибір, то автоматично виставляється нормальна складність. Після чого я створив змінну, яка відповідає за раунд, який зараз проходить і потім написав цикл, який виконується до тих пір, поки активний раунд не дійде до введеного користувачем значення раундів. В середині цього циклу відбувається введення жесту користувачем і перевірка введеного жесту, після чого відбувається розрахування ходу комп’ютера за допомогою функції computerfync, в яку передається складність та хід гравця. В середині цієї функції є список зі всіма можливими ходами, якщо була передана нормальна складність, то вибирається випадкове значення із цього списку, а якщо складно, то береться хід гравця і до списку додається ще одне значення, яке перемогає жест гравця і після чого вибирається випадкове значення зі списку і повертається до місця виклику функції. Після цього викликається функція winner, в яку передається хід гравця і хід комп’ютера, в середині за допомогою умов вирішується, хто переміг, а якщо два ходи одинакові, то це нічія і після чого повертається відповідь. Потім відбувається перевірка того, хто виграв і додається 1 бал для переможця і додається одиниця до змінної поточного раунду. Після кожного раунду виводиться кількість раундів та кількість балів гравця та комп’ютера. Якщо цикл раундів закінчиться, то відбувається перевірка, у кого більше балів і після чого в виводиться переможець. Потім я додав цикл, який перевіряє, чи хоче гравець зіграти знову, відбувається обробка невірних значень і якщо гравець вводить ‘n’, то закінчується виконання циклу.

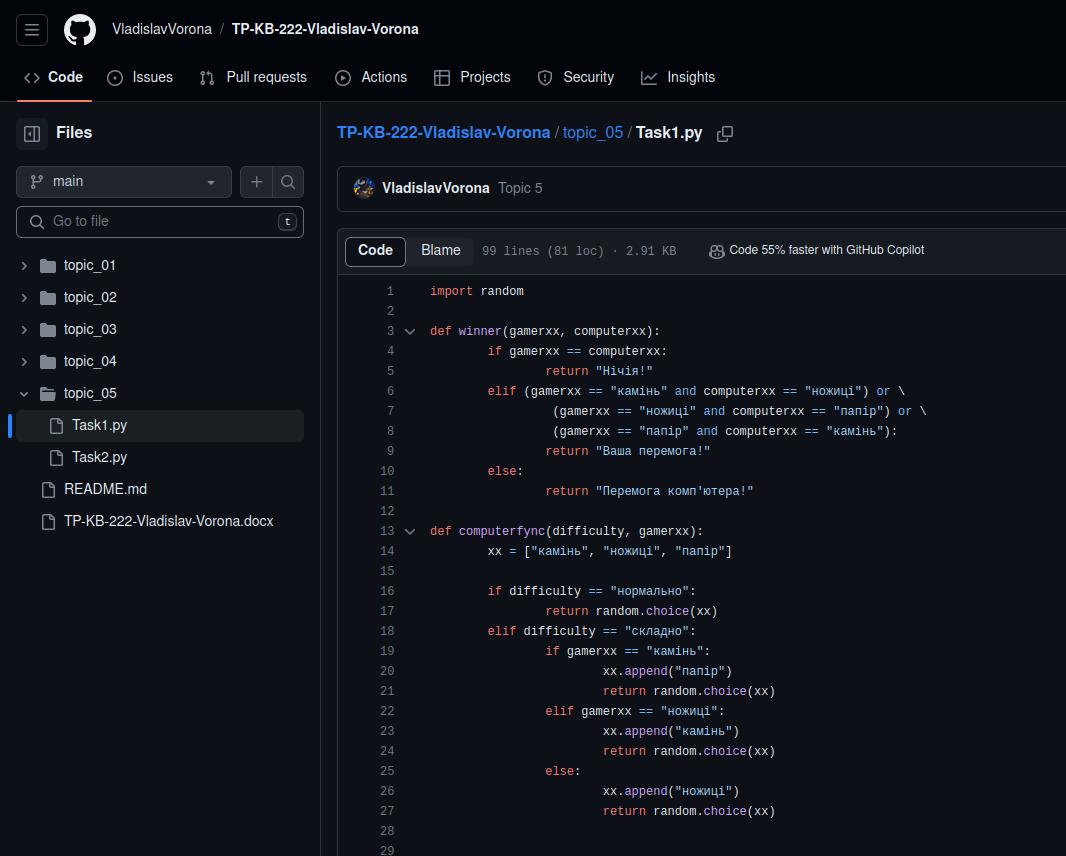
### **Код завдання:**





### **GitHub:**

Посилання на GitHub: https://github.com/VladislavVorona/TP-KB-222-Vladislav-Vorona/blob/main/topic\_05/Task1.py  
 Знімок екрану з GitHub:



## Програма конвертування

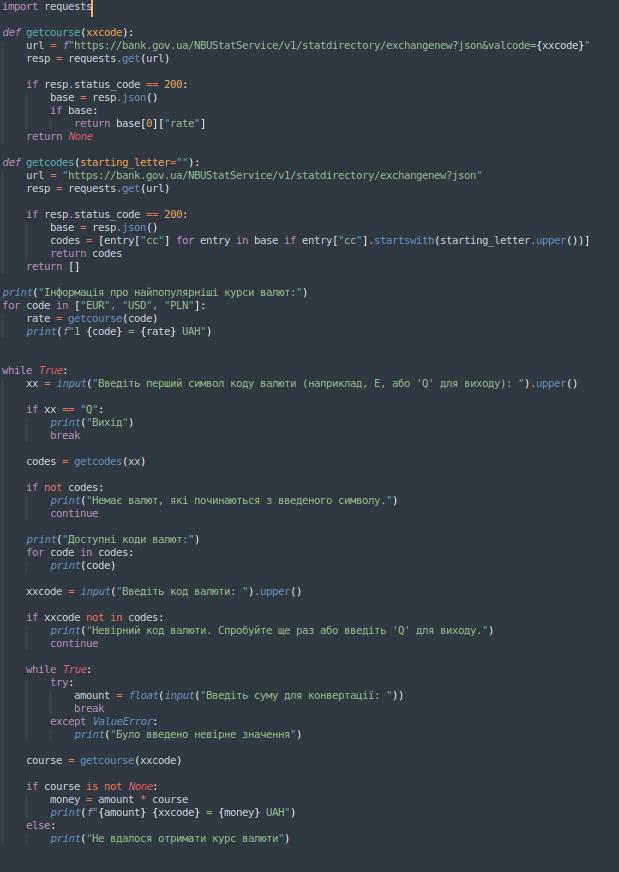
**Суть завдання:**

Програма конвертування іноземної валюти в українську гривню. Для отримання актуальних курсів валют необхідно використовувати API НБУ та модуль, що надає можливість виконувати запити до сторонніх сервісів requests. Достатня умова роботи – можливість конвертації для трьох іноземних валют EUR, USD, PLN. Користувачу надається можливість введення кількості та типу валюти, результат роботи програми – конвертоване значення в українських гривнях.

### **Хід виконання завдання:**

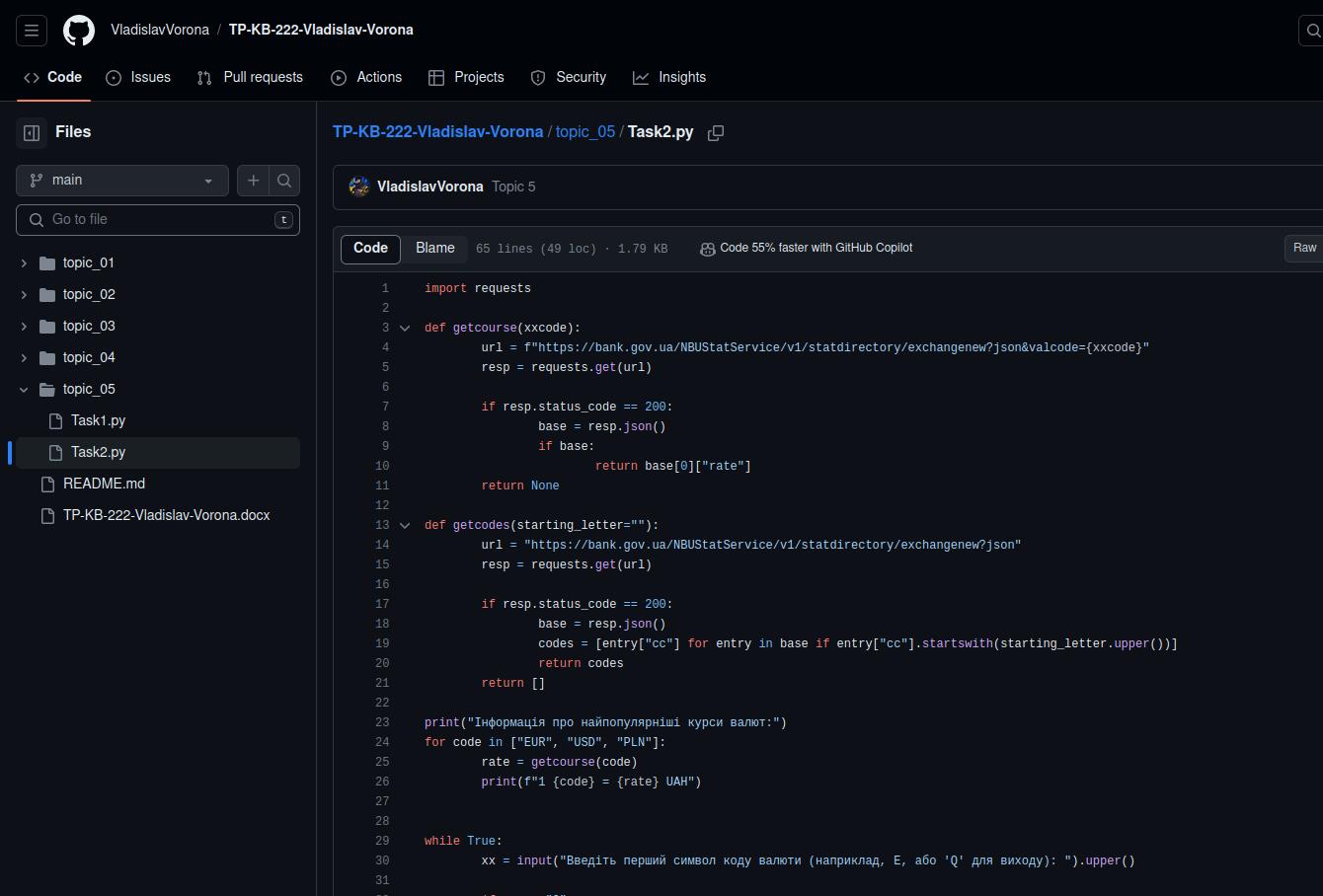
Для виконання цього завдання я імпортував бібліотеку requests для роботи із HTTP-запитами. Спочатку я вирішив вивести на екран за допомогою циклу найбільш популярні курси валют, для цього я написав функцію getcourse, в яку передається код валюти. В цій функції створюється змінна із URL для запиту до сайту Національного банку України із переданим кодом валюти, після чого використовується бібліотека requests для виконання HTTP-запиту, після чого перевіряється статус відповіді, якщо код відповіді дорівнює 200, то відбувається парсинг отриманих даних у формат JSON і після чого повертається курс першої валюти у списку. Після виконання циклу виведення найбільш популярних курсів валют я створив вічний цикл, в якому відбувається введення користувачем перших літер коду валюти, а якщо було введено ‘Q’ то закінчувати виконання цього циклу. Після введення перших символів валюти я створив функцію getcodes, яка отримує всі можливі коди валют, які починаються із вказаних користувачем символів. В средині цієї функції ми створюємо змінну, в яку заніс URL для запиту до сайту Національного банку України і після чого ми виконуємо цей HTTP-запит і парсимо отримані дані у формат JSON, після чого відбувається формування списку кодів валют, які починаються із вказаних користувачем символів і після чого повертає ці коди, після чого, якщо немає кодів, то виводиться повідомлення, що немає валют, які починаються з введених символів і цикл починається спочатку, якщо коди є, то далі вони виводяться за допомогою циклу і після чого в користувача запитується валюта, яка його цікавить, якщо такого коду немає в списку кодів, то виводити інформацію, що невірний код валюти, а якщо код є, то далі виконується цикл для введення суми для конвертації із перевіркою введених значень. Після введення вірних значень знову виконується функція getcourse для отримання курсу валюти за кодом. В цій функції створюється змінна із URL для запиту до сайту Національного банку України із переданим кодом валюти, після чого використовується бібліотека requests для виконання HTTP-запиту, після чого перевіряється статус відповіді, якщо код відповіді дорівнює 200, то відбувається парсинг отриманих даних у формат JSON і після чого повертається курс першої валюти у списку. Потім перевіряється, чи було отримано курс валюти і якщо його було отримано, то відбуваєтсья множення курсу на суму користувача і виведення результату, а якщо курс не було отримано, то виводиться повідомлення, що не вдалося отримати курс валюти. Після виконання цієї умови виконання циклу починається спочатку.

### **Код завдання:**



### **GitHub:**

Посилання на GitHub: https://github.com/VladislavVorona/TP-KB-222-Vladislav-Vorona/blob/main/topic\_05/Task2.py  
 Знімок екрану з GitHub:



## Програма калькулятор

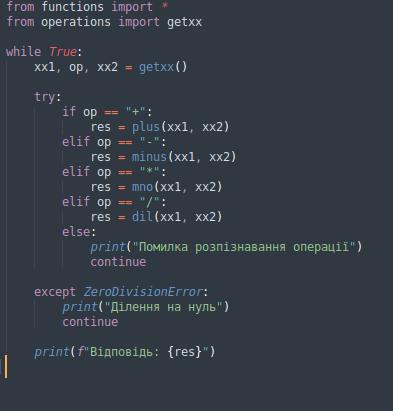
**Суть завдання:**

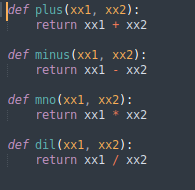
Використання модулів для програми калькулятор. Функції додавання, віднімання, множення та ділення перенести в файл functions.py. Функції запиту на введення даних для операцій та самих операцій перемістити в файл operations.py. Програму калькулятор реалізувати в файлі calc.py, до якого підключають файл functions.py та operations.py.

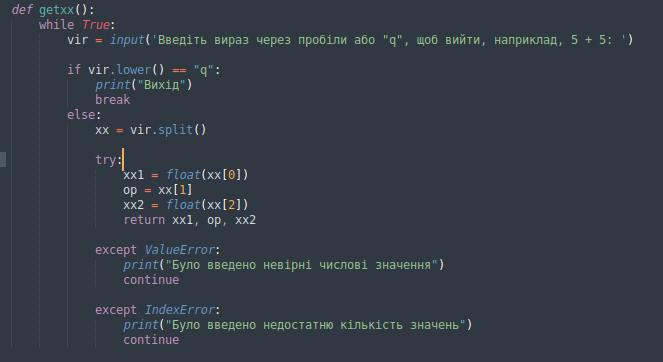
### **Хід виконання завдання:**

Для виконання цього завдання я скопіював код калькулятору із минулого завдання і створив 3 файли, calc.py, functions.py, operations.py. Спочатку я переніс всі функції, які виконують певні математичні дії, до файлу functions.py, після чого переніс функцію отримання числових значень та операції getxx до файлу operations.py. Решту коду скопіював до файлу calc.py, після чого додав імпортування всіх функцій із файлу functions.py та імпортував функцію getxx із файлу operations.py, після чого починається виконання стандартного циклу калькулятору.

### **Код завдання:**







### **GitHub:**

Посилання на GitHub: [https://github.com/VladislavVorona/TP-KB-222-Vladislav-Vorona/blob/main/topic\_05/calc.py](https://github.com/VladislavVorona/TP-KB-222-Vladislav-Vorona/blob/main/topic_05/Task3.py)

Посилання на GitHub: [https://github.com/VladislavVorona/TP-KB-222-Vladislav-Vorona/blob/main/topic\_05/operations.py](https://github.com/VladislavVorona/TP-KB-222-Vladislav-Vorona/blob/main/topic_05/Task3.py)

Посилання на GitHub: [https://github.com/VladislavVorona/TP-KB-222-Vladislav-Vorona/blob/main/topic\_05/functions.py](https://github.com/VladislavVorona/TP-KB-222-Vladislav-Vorona/blob/main/topic_05/Task3.py) Знімок екрану з GitHub:

