**Звіт про виконання завдань для практики з курсу «Технології програмування»**

[1. Звіт до теми №1 2](#_Toc149820673)

[1.1. Обернення рядка. 2](#_Toc149820674)

[1.2. Тестування різних методів для роботи зі строками. 2](#_Toc149820675)

[1.3. Функція для знаходження дискримінанту. 4](#_Toc149820676)

[2. Звіт до теми №2 6](#_Toc149820677)

[2.1. Написати функцію для знаходження коренів рівняння, використовуючи умови. 6](#_Toc149820678)

[2.2. Написання калькулятора за допомогою умов if, elif. Написання калькулятора за допомогою оператора match. 8](#_Toc149820679)

[3. Звіт до теми №3 13](#_Toc149820680)

[3.1. Постійно працююча програма за допомогою циклу while. 13](#_Toc149820681)

[3.2. Експерименти з методами масиву. 15](#_Toc149820682)

[3.3. Експерименти з методами для словарів. 18](#_Toc149820683)

[3.4. Написання функції для вставки нових елементів у відсортований масив. 20](#_Toc149820684)

[4. Звіт до теми №4 23](#_Toc149820685)

[4.1. Обробка виняткових ситуацій. 23](#_Toc149820686)

[4.2. Ознайомлення зі списком виняткових ситуацій. 25](#_Toc149820687)

[5. Звіт до теми №5 26](#_Toc149820688)

[5.1. Гра «Камінь, Ножиці, Папір». 26](#_Toc149820689)

[5.2. Конвертер валют. 28](#_Toc149820690)

[5.3. Модулі для програми «Калькулятор». 30](#_Toc149820691)

[6. Звіт до теми №6 35](#_Toc149820692)

[6.1. Використання lamda-функції. 35](#_Toc149820693)

[6.2. Логування всіх дій у застосунку «Калькулятор». 37](#_Toc149820694)

[7. Звіт до теми №7 44](#_Toc149820695)

[7.1. Перегляд документації про Class. 44](#_Toc149820696)

[7.2. Перегляд документації про \_\_func\_\_. 44](#_Toc149820697)

[7.3. Перепис коду програми із завдання 6.1. (про сортування списку і ламбду-функцію) із використанням ООП. 44](#_Toc149820698)

[7.4. Переписати код програми «Калькулятор» за допомогою ООП. 46](#_Toc149820699)

# Звіт до теми №1

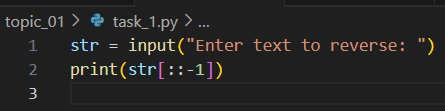
## Обернення рядка.

Завдання: Необхідно перевернути рядок **abcdefg123** так, щоб він був таким: **321gfedcba**.

Виконання завдання:

Створив змінну із запитом користувача, який запитує, яку строку обернути. Далі виводимо через print, у середині за допомогою оператора [::-1] я повернув обернений рядок. Перше та друга крапки в операторі [::-1] означають, що потрібно брати всі елементи від початку до кінця, а число «-1» означає, що відлік має починатися з кінця.

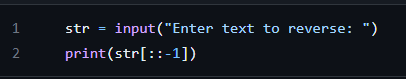
Код завдання:



GitHub:

Посилання на GitHub на цей файл: <https://github.com/danylosykylinda/TP-KB-221-Danylo-Sykylinda/blob/main/topic_01/task_1.py>

Скріншот коду з GitHub:



## Тестування різних методів для роботи зі строками.

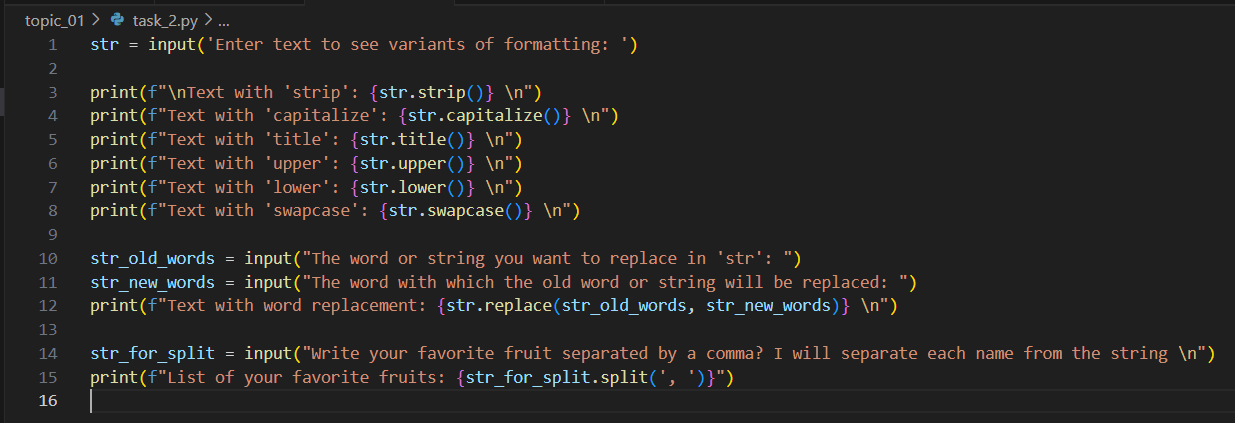
Завдання: Перевірити методи для роботи зі строками strip(), capitalize(), title(), upper(), lower(), swapcase(), replace, split().

Виконання завдання:

Створив змінну із запитом про текст у користувача, потім через функцію print виводжу відформатований текст різними методами(strip(), capitalize(), title(), upper(), lower(), swapcase()) і таким чином показую, як ці методи працюють. Далі для методу replace () я створив дві змінні з запитом до користувача. У першій я запитую, яке слово чи текст хоче замінити в строці, яка є в змінній **str**, а в другій – на яке слово чи текст буде замінений попереднє слово чи текст. Далі виводжу відформатований текст через print, у ньому всередині replace() заміняє старі слова або текст на нові слова або текст.

Далі я вирішив випробувати метод split(). Для цього створив змінну із запитом до користувача про його улюблені фрукти і попрохав перечислити його їх через кому. Потім через print виводжу список улюблених фруктів, список цей видає метод split(), за допомогою нього я виділив із строки назви фруктів, вказавши в split параметр ‘, ‘, тобто, шукати кому з пробілом. У підсумку цей метод повертає список елементів – улюблених фруктів користувача.

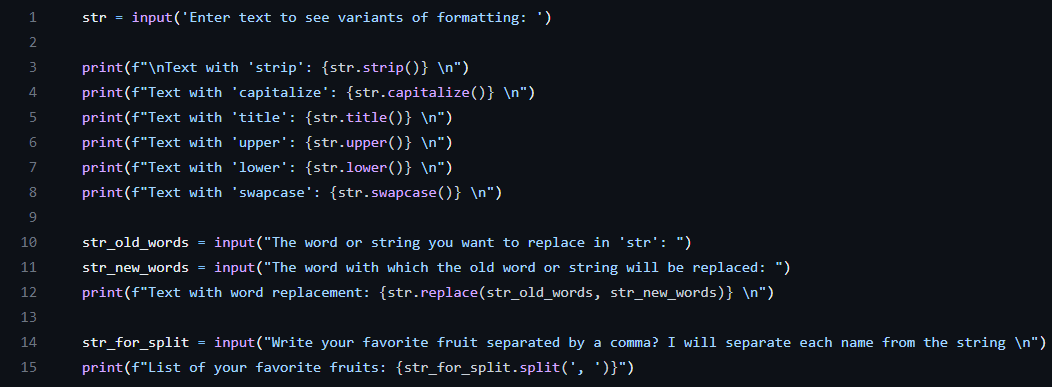
Код завдання:



GitHub:

Посилання на GitHub на цей файл: <https://github.com/danylosykylinda/TP-KB-221-Danylo-Sykylinda/blob/main/topic_01/task_2.py>

Скріншот коду з GitHub:



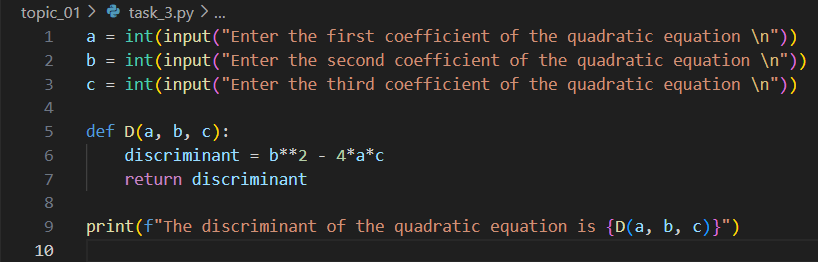
## Функція для знаходження дискримінанту.

Завдання: Потрібно написати функцію, яка приймає три параметри a, b, c – коєфіцієнти квадратного рівняння та повертає значення дискримінанту.

Виконання завдання:

Я створив три змінні із запитом до користувача, щоб отримати коєфіцієнти рівняння а, b, c. Далі створив функцію D, що приймає три аргументи a, b, c. У всередині функції створив змінну discriminant, якій присвоїв формулу b2-4ac. І через return повернув результат функції. Потім виводжу через print, при цьому всередині неї викликаю функцію D і передаю в якості параметрів змінні зі значеннями, що надав користувач. І тоді ми отримуємо результат.

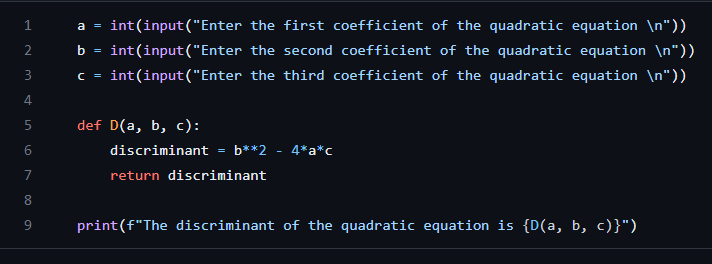
Код завдання:



GitHub:

Посилання на GitHub на цей файл: <https://github.com/danylosykylinda/TP-KB-221-Danylo-Sykylinda/blob/main/topic_01/task_3.py>

Скріншот коду з GitHub:



# Звіт до теми №2

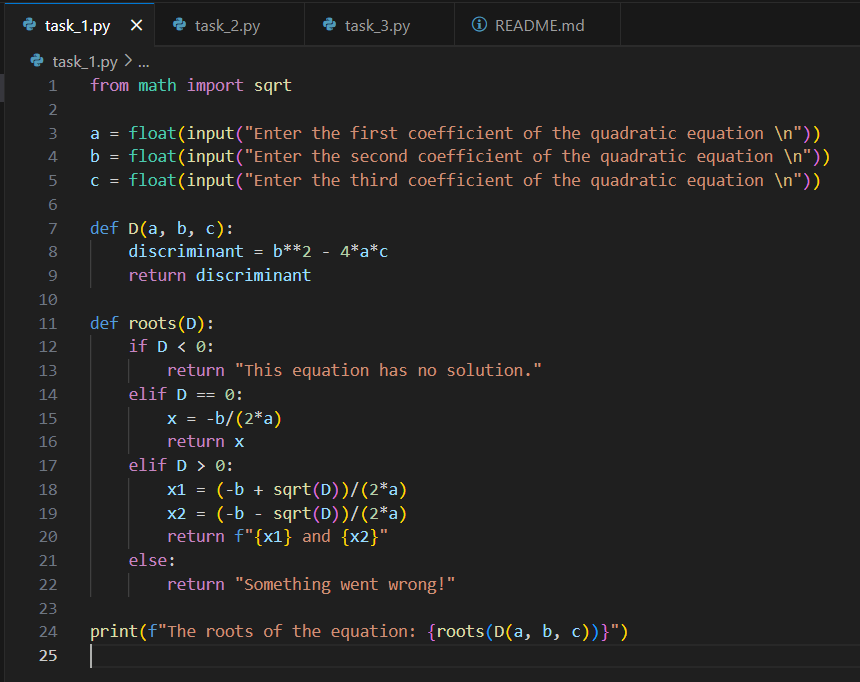
## Написати функцію для знаходження коренів рівняння, використовуючи умови.

Завдання: Необхідно написати функцію для знаходження коренів рівняння, використовуючи умови. Умови ми використовуємо для дискримінанту(далі буде показано як). Функцію знаходження дискримінанту ми візьмемо з розділу 1. завдання 1.3.

Виконання завдання:

Я імпортував з модулі math функцію sqrt. Далі переніс частину коду з розділу 1. завдання 1.3. Дещо переписав: отримання числа не в int, а вже у float для того, щоб можна було не тільки цілі числа використовувати, але й дробові. Потім написав функцію roots, яка отримує параметр – дискримінант. За допомогою умов if, elif прописав умови для різних випадків: якщо дискримінант менше нуля, повертаємо результат - стрічку, що рівняння немає коренів(стрічки 12-13), якщо дискримінант дорівнює нулю, то маємо один корінь(стрічки 14-16), обчислюємо його і виводим, якщо дискримінант більше нуля, то маємо два корені і обчислюємо і виводим. І за допомогою else, якщо піде щось не так, то виводили повідомлення про те, що щось не так. Далі через print виводимо результат, у print викликаємо функцію roots, у якості параметра передаємо функцію D з параметрами a, b, c.

Код завдання:

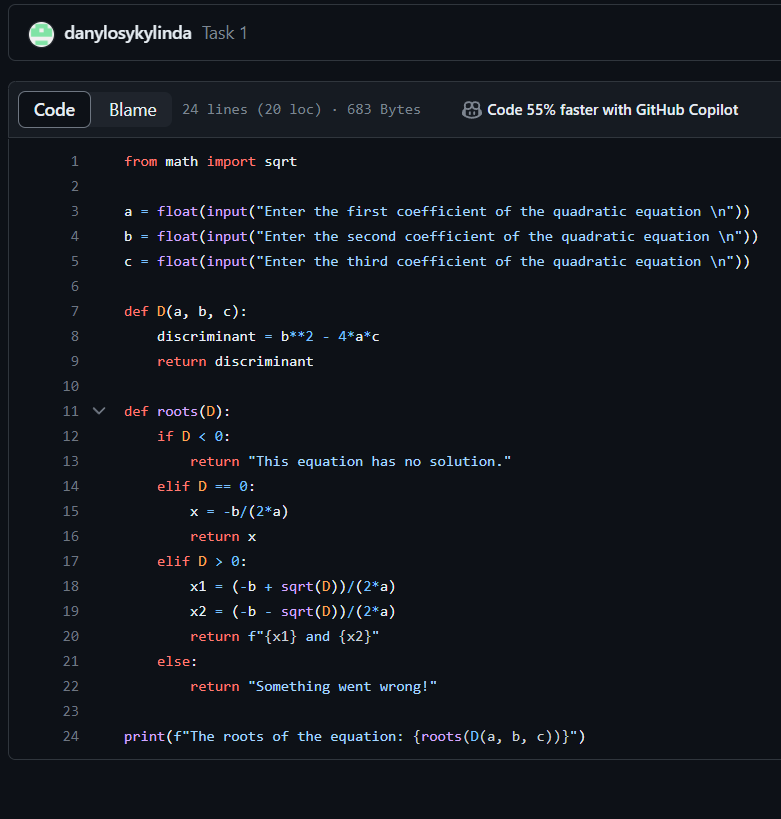


GitHub:

Посилання на GitHub на цей файл:

<https://github.com/danylosykylinda/TP-KB-221-Danylo-Sykylinda/blob/main/topic_02/task_1.py>

Скріншот коду з GitHub:

****

## Написання калькулятора за допомогою умов if, elif. Написання калькулятора за допомогою оператора match.

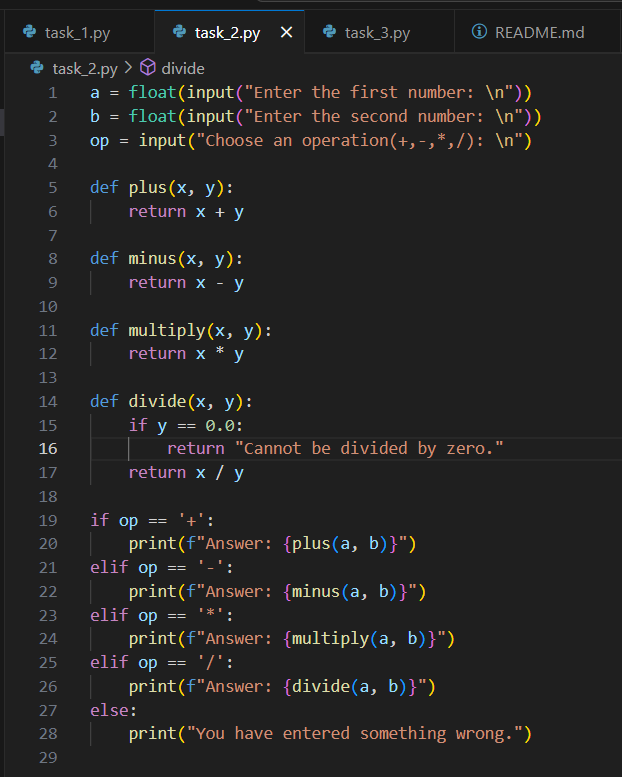
Завдання: Потрібно написати калькулятор, у якому ми запитуємо два числа і дію(+,-,/,\*) від користувача, при цьому коді використовуємо умови if, elif. Потім написати калькулятор, у коді якого ми використаємо оператор match.

Виконання завдання:

Частина 1.

Створив три змінні із запитом до користувача: a та b – числа, op – операція над цими числами. Потім написав функції, які виконують операції(plus – додавання, minus – віднімання, multiply – множення, divide - ділення). У функції divide ми ще умовою перевіряємо, чи другий аргумент є нулем, бо якщо другий дорівнює нулю, то не можна поділити, на нуль не ділиться. Потім написав умову, якщо у змінну користувач написав ‘+’, то ми виводимо результат в print, у ньому викликаємо функцію plus і передаємо два числа їй – a та b. Аналогічні дії робимо для інших операцій. У кінці дописуємо else з повідомленням, якщо користувач написав щось неправильно.

Код завдання:

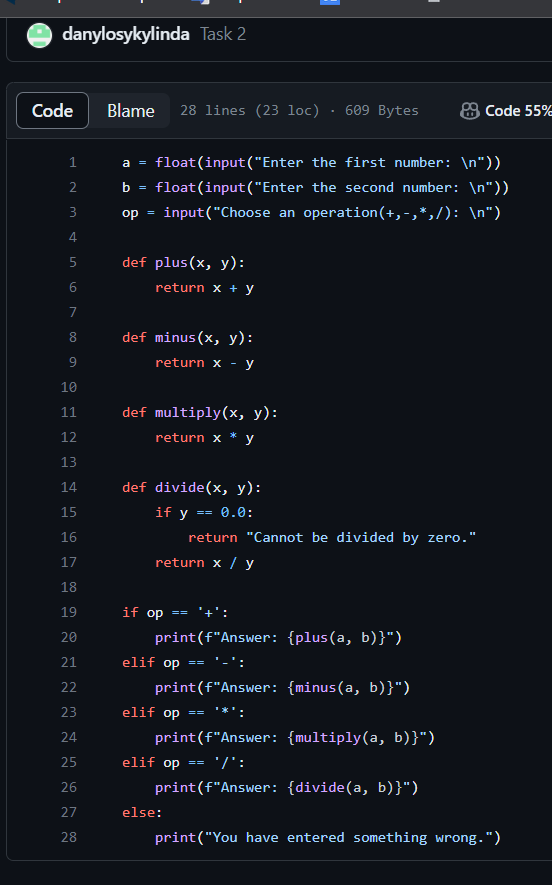


GitHub:

Посилання на GitHub на цей файл:

<https://github.com/danylosykylinda/TP-KB-221-Danylo-Sykylinda/blob/main/topic_02/task_2.py>

Скріншот коду з GitHub:

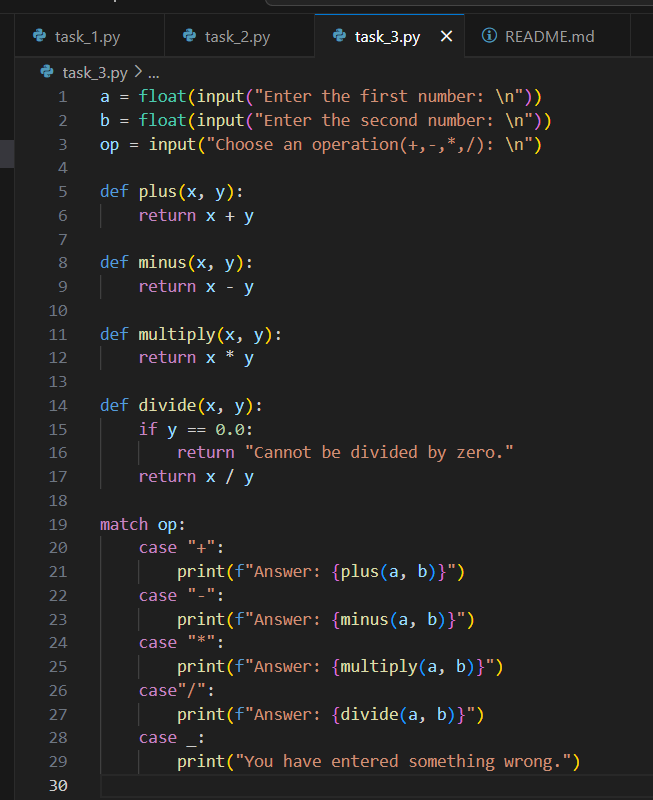
****

Виконання завдання:

Частина 2.

Стрічки 1-17 скопійовані з частини 1 цього завдання. Далі ми за допомогою оператора match прописуємо випадки(case) для (+, -, \*, /). І аналогічно, як частині 1 цього завдання, ми викликаємо функцію print, у ній викликаємо функцію plus і передаємо два аргументи a та b. Аналогічно з іншими математичними діями. У кінці прописуємо case \_: - це означає, що якщо будуть уведені інші символи відмінні, що зазначені в інших case, то реагувати певним чином, у даному case ми виводимо повідомлення, що щось пішло не так.

Код завдання:

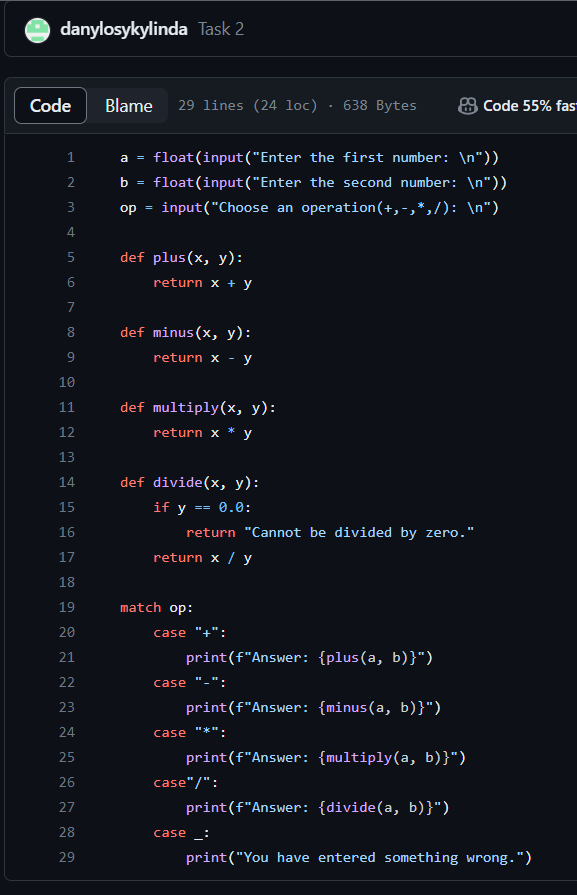


GitHub:

Посилання на GitHub на цей файл:

<https://github.com/danylosykylinda/TP-KB-221-Danylo-Sykylinda/blob/main/topic_02/task_3.py>

Скріншот коду з GitHub:

****

# Звіт до теми №3

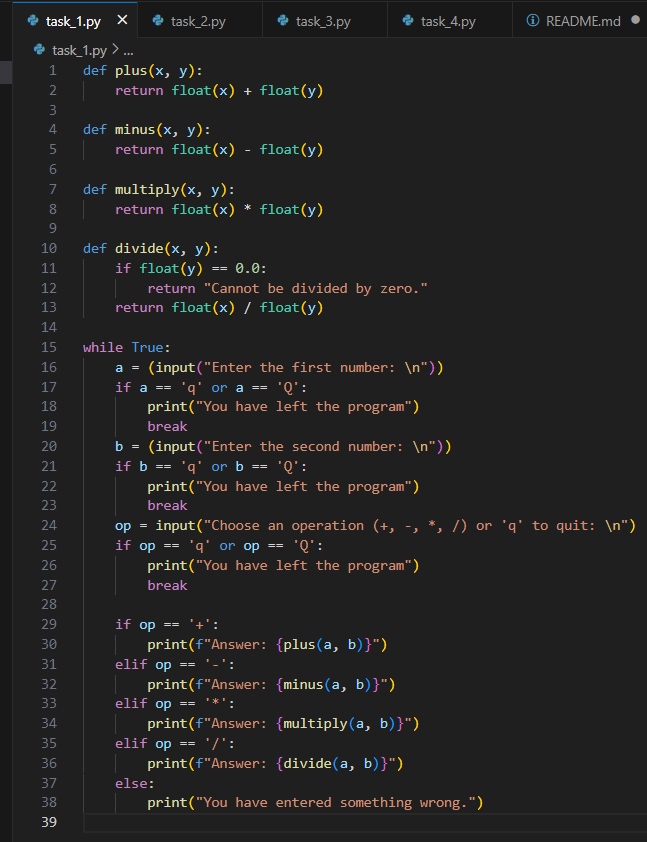
## Постійно працююча програма за допомогою циклу while.

Завдання: Необхідно переписати калькулятор із з розділу 2 пункту 2 так, щоб він безперервно працював, тобто, після введення даних користувачем він знову запитував користувача про числа та дії, також потрібно реалізувати, щоб користувач міг завжди вийти із циклу.

Виконання завдання:

Для початку беремо код з попереднього розділу пункту 2 (тут візьмемо калькулятор на if/elif), для більшої читальбельності винесемо функції окремо, а все інше обернемо в цикл while. while True: означає, що поки не буде break програма буде працювати. Після кожного вводу користувача перевіряємо, чи користувач не увів “Q” або “q”(ці букви у нас будуть позначати вихід). Якщо “q” чи “Q” було уведено, то значить виводимо повідомлення про вихід, далі break і цикл зупинено, програма завершила роботу, якщо ж не уведено, то програма починає знову запитувати користувача про числа та операцію над ними.

Код завдання:

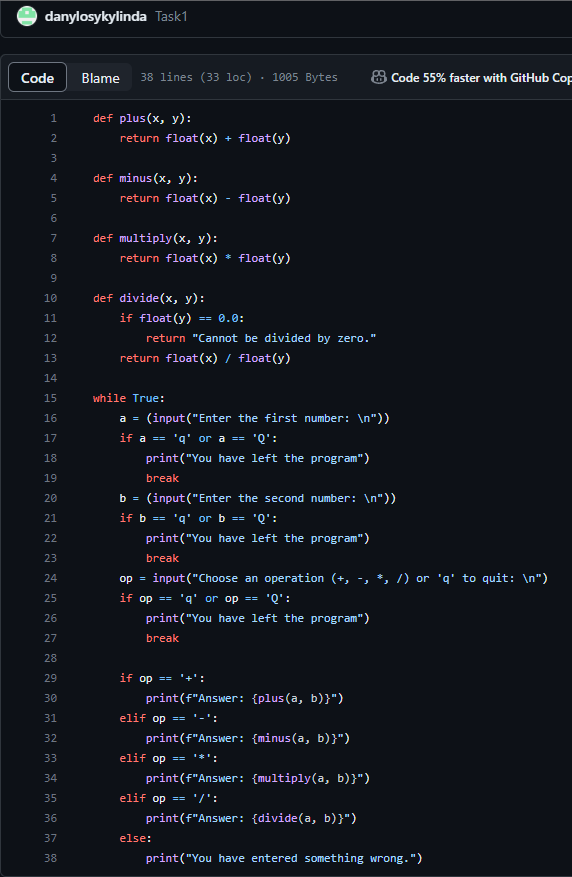


GitHub:

Посилання на GitHub на цей файл:

<https://github.com/danylosykylinda/TP-KB-221-Danylo-Sykylinda/blob/main/topic_03/task_1.py>

Скріншот коду з GitHub:



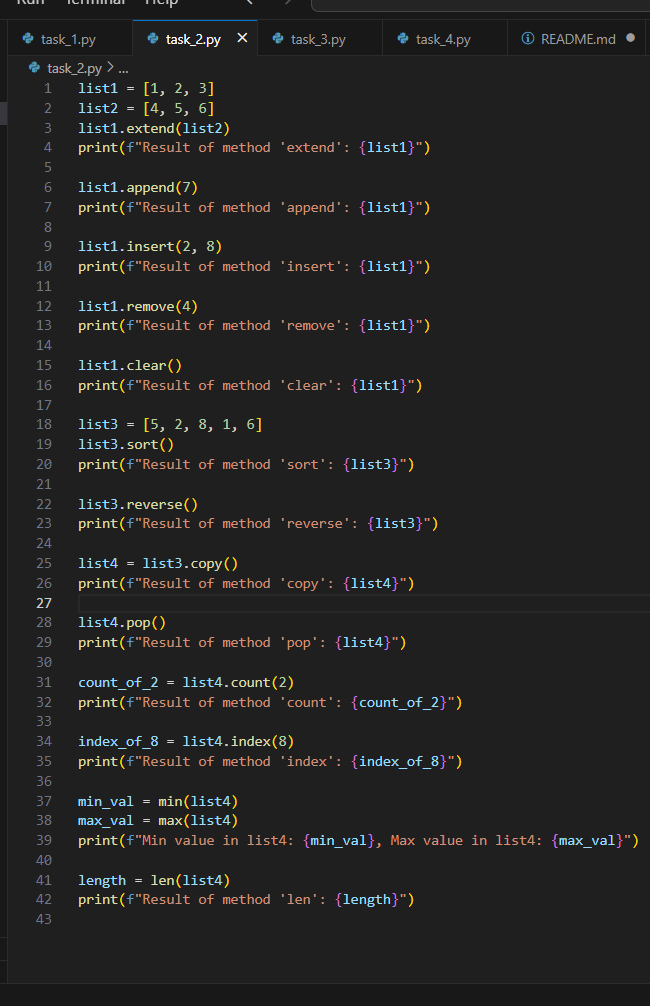
## Експерименти з методами масиву.

Завдання: Поекспериментувати з методами масиву такими як: extend(), append(), insert(id, val), remove(val), clear(), sort(), reverse(), copy() та з іншими методами масиву.

Виконання завдання:

Я створив два масиви list1 та list2 із різними числами. Застосувавши метод extend() з аргументом масив list2 для list1, я доповнив масив list1 масивом list2. Далі методом append() я додав число 7 в кінець масиву list1. Потім за допомогою методу insert() я поставив число 8 в масив list1 на позицію 2. Далі я видалив число 4 з масиву list1 за допомогою методу remove(). Потім за допомогою методу clear() я очистив масив list1 – тепер він пустий. Далі я створив масив list3 з різними числами, використавши метод sort(), я відсортував масив з числами в порядку зростання. Потім за допомогою методу reverse() я повернув всі значення масиву у зворотньому порядку. Далі я створив змінну, в яку копіював list3 за допомогою методу copy(). Потім для list4 я використав метод pop(), цей метод видалив останній елемент в масиві. Далі я використав метод count() з аргументом число 2 в масиві list4, цей метод допомагає знайти нам, скільки чисел 2 знаходиться в цьому масиві. Потім я використав метод index() з аргументом число 8 в масиві list4, цей метод знаходить індекс цього значення у масиві, тобто в цьому випадку він шукає на якій позиції в масиві знаходиться число 8. Далі я використав вбудовані функції min() і max() та передав в якості аргументу масив list4, відповідно ці функції знайшли найменше та найбільше значення в масиві. Далі використав метод len() для масиву list4, щоб дізнатися, яка кількість елементів знаходиться в масиві.

Код завдання:

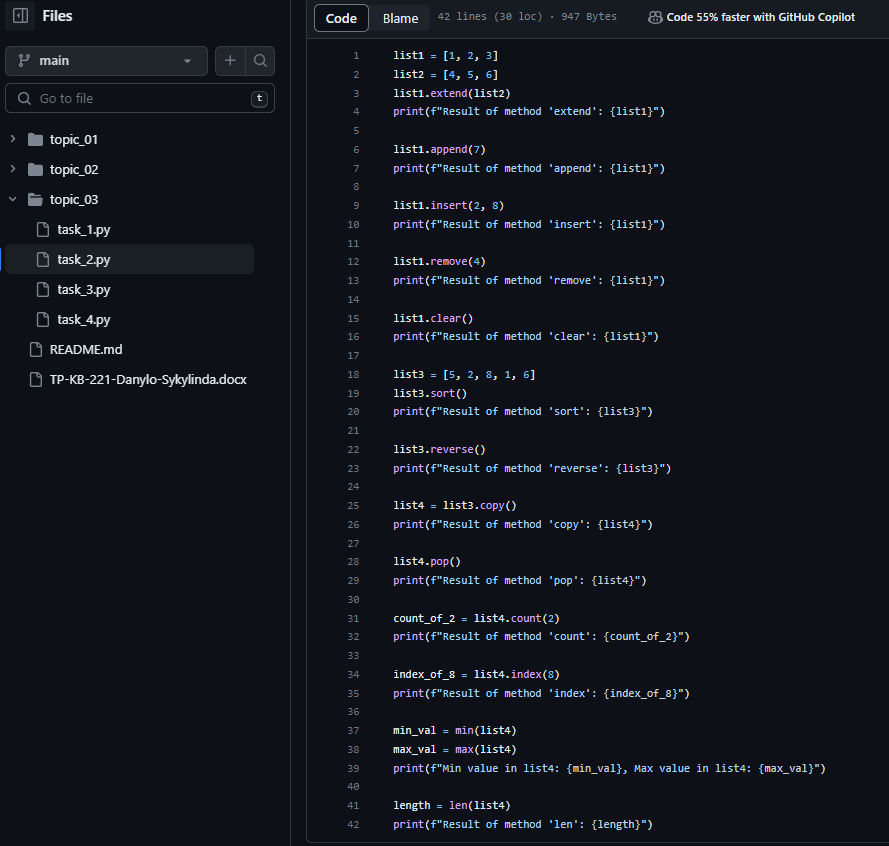


GitHub:

Посилання на GitHub на цей файл:

<https://github.com/danylosykylinda/TP-KB-221-Danylo-Sykylinda/blob/main/topic_03/task_2.py>

Скріншот коду з GitHub:



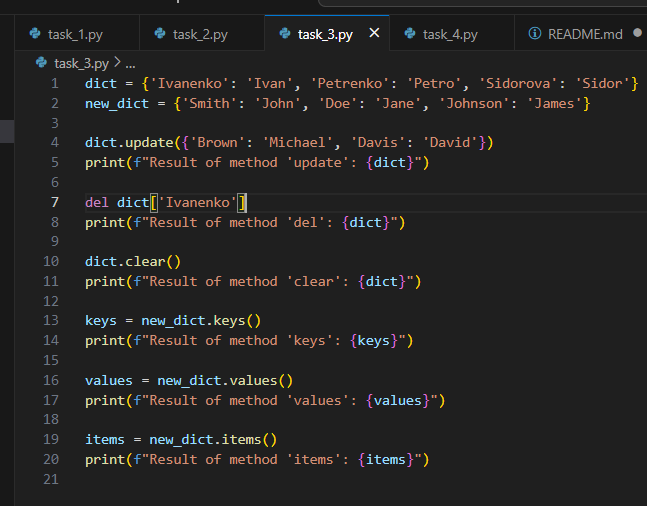
## Експерименти з методами для словарів.

Завдання: Поекспериментувати з методами для словарів такі як: update(), del, clear, keys(), values(), items().

Виконання завдання:

Я створив два словарі з іменами та прізвищами. Далі за допомогою методу update() я додав в масив dict два ім’я-прізвища (ім’я як ключ, а прізвище як значення). Потім я видалив з словара dict одне ім’я-прізвище за допомогою del, звертаючись по ключу. Потім я за допомогою методу clear() я очистив словар dict. Далі за допомогою методу keys() я дізнався про всі наявні ключі в словарі new\_dict. За допомогою методу values() я дізнався про всі значення ключів в словарі new\_dict. Потім за допомогою методу items() я дізнався про кожну пару ключ-значення в словарі new\_dict, тобто, відразу дізнався про кожен ключ і його значення, що є словарі new\_dict.

Код завдання:

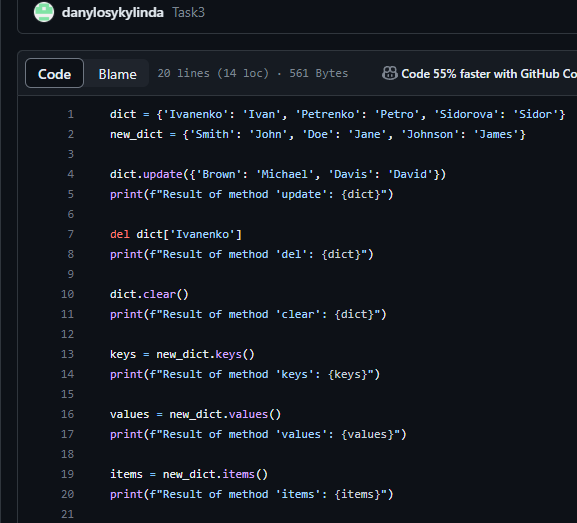


GitHub:

Посилання на GitHub на цей файл:

<https://github.com/danylosykylinda/TP-KB-221-Danylo-Sykylinda/blob/main/topic_03/task_3.py>

Скріншот коду з GitHub:



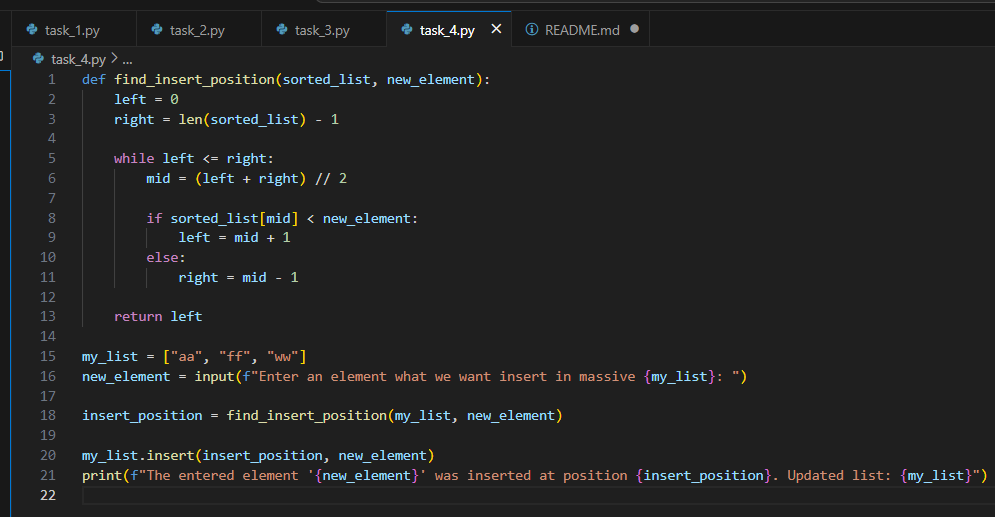
## Написання функції для вставки нових елементів у відсортований масив.

Завдання: Написати функцію, яка буде шукати позицію для вставки в масив нового елемента, позиція для нового елементу має бути така, щоб масив лишився сортованим. При цьому не можна використовувати вбудовані функції.

Виконання завдання:

Щоб реалізувати правильну вставку вже у відсортований масив my\_list, потрібно використати алгоритм бінарного пошуку. Наша функція find\_insert\_position прийматиме два аргументи: перший – відсортований масив, другий – значення, яке потрібно правильно поставити в масив так, щоби він залишився відсортованим. Як це буде працювати? Масив ділять на дві частини. Далі створюємо дві змінні left і right, в left поміщаємо 0(перший індекс), а в right останній індекс масиву. Потім пишемо цикл допоки ліва частина менша або рівна правій, ми шукаємо середину так, щоб вона була цілим числом(тому що це індекс), робимо перевірку, якщо елемент з індексом mid менший за елемент, який ми хочемо вставити, то ми до left присвоюємо середину плюс один, оскільки ліва частина нас вже не цікавить, бо там числа ще менші за наш новий елемент. Якщо ситуація навпаки, що у нас середній елемент більший за елемент, що вставляють, то відкидаємо праву частину і призначаємо змінній right середину мінус один. Коли left зрівняється з right, то ми повертаємо left, який і буде позицією для вставки нового елемента. І далі вже вставляємо елемент за допомогою методу insert() у відсортований масив і виводимо масив, щоб переконатися, що все вийшло правильно.

Код завдання:

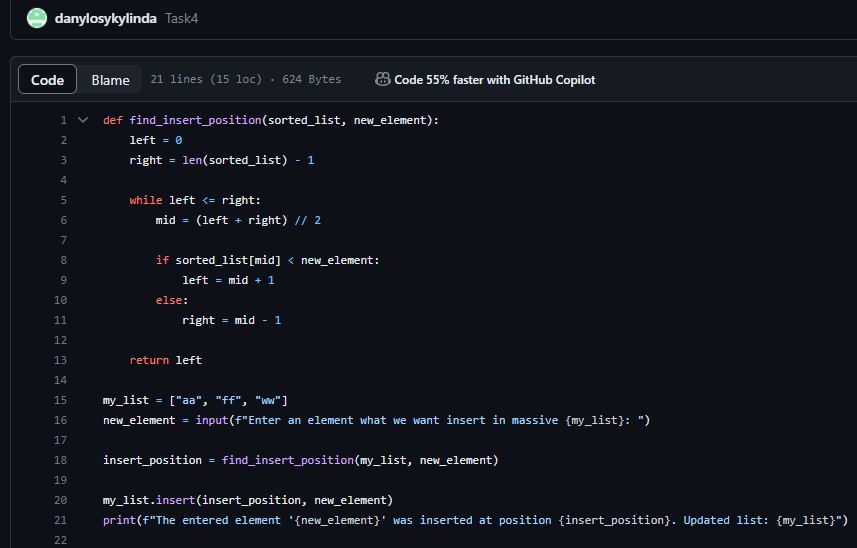


GitHub:

Посилання на GitHub на цей файл:

<https://github.com/danylosykylinda/TP-KB-221-Danylo-Sykylinda/blob/main/topic_03/task_4.py>

Скріншот коду з GitHub:



# Звіт до теми №4

## Обробка виняткових ситуацій.

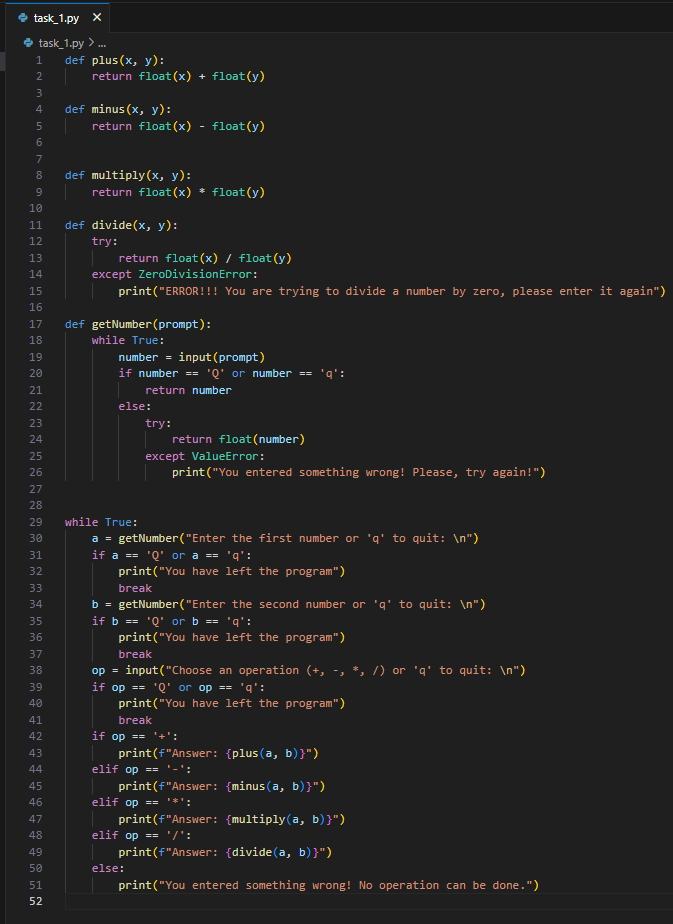
Завдання: Потрібно написати функцію, яка прийматиме запит у користувача у програмі калькулятора (із попередньої теми завдання 3.1) та оброблюватиме виняткові ситуації (неправильний ввід користувача – введення букв чи інших знаків замість тільки чисел), а також у функції ділення обробити виняткову ситуацію ділення ділення на нуль.

Виконання завдання:

Скопіюємо код із попередньої теми завдання 3.1 в наш файл. Для початку треба зрозуміти, яка помилка видається при неправильному вводі(введення букв та інших знаків замість тільки чисел). І якщо глянути на код, то стає зрозуміло, що якщо буде введено, наприклад, букви, то буде помилка **ValueError** через те, що вбудована функція float() не зможе перетворити рядок в цифри, бо там букви, а не числа. Отже нам потрібно спіймати помилку ValueError. Це ми можемо зробити за допомогою конструкції **try:… except: …** Тож створюємо функцію getNumber(), яка прийматиме запит і буде перевіряти, чи число, чи щось інше на виході. Спочатку в циклі ми перевіримо введене користувачем, чи не є воно(q або Q) виходом з програми, якщо є, то програма завершиться. Якщо ж користувач такого не уводив, тоді ми в try… поміщаємо return float(number), якщо це буде щось відмінне від числа, то ми отримаємо помилку ValueError, отже в except пишемо цю помилку, щоб «спіймати» і, коли помилка буде «спіймана», то вивести повідомлення. Не забудемо поміняти input() на getNumber(), оскільки запит є в самій функції(точніше ми передаємо питання для запиту до користувача, а функція цей запит виконує).

У функції divide() при діленні на нуль ми отримаємо помилку **ZeroDivisionError**. Отже її потрібно обробити і вивести повідомлення про те, що сталася помилка через ділення на нуль. Для цього пишемо конструкцію try:…. except…, у блок try поміщаємо return float(x) / float(y), а в except пишемо помилку, яку ми хочемо «спіймати», а саме: ZeroDivisionError, і також в цьому блоку пишемо повідомлення, на випадок якщо ця помилка станеться.

Код завдання:



GitHub:

Посилання на GitHub на цей файл: <https://github.com/danylosykylinda/TP-KB-221-Danylo-Sykylinda/blob/main/topic_04/task_1.py>

Скріншот коду з GitHub:



## Ознайомлення зі списком виняткових ситуацій.

Завдання: Ознайомитись зі списком виняткових ситуацій за посиланням <https://docs.python.org/3/library/exceptions.html> .

Виконання завдання:

Я поглибив свої знання щодо виняткових ситуацій і дізнався про нові такі як: RecursionError(рекурсія вичерпала свій ліміт), OverflowError(наприклад, перевищення максимального допустимого числа для певного типу даних), ImportError(наприклад, відсутність модуля, що мав імпортуватися) та інші.

# Звіт до теми №5

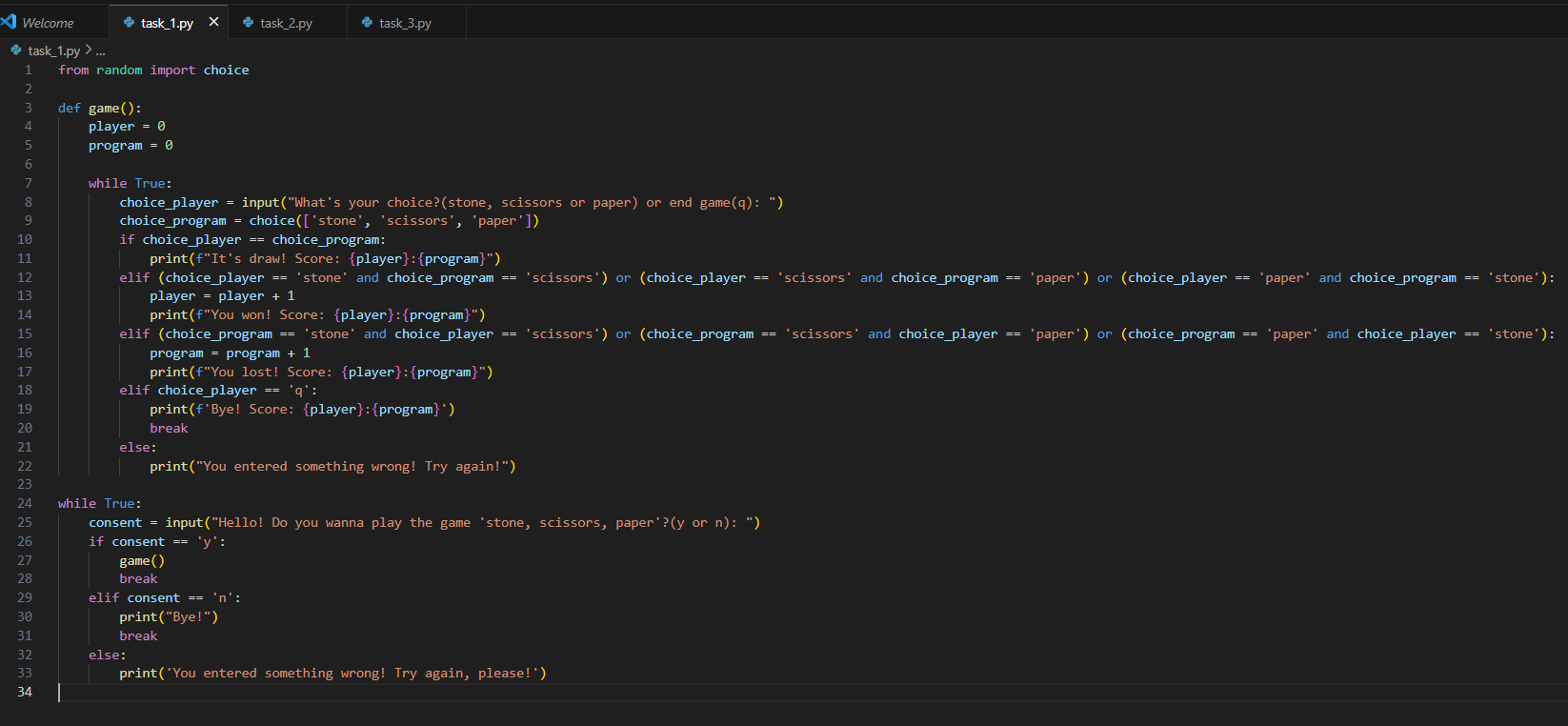
## Гра «Камінь, Ножиці, Папір».

Завдання: Написати реалізацію гри «Камінь, Ножиці, Папір».

Виконання завдання:

Імпортуємо метод choice() з модуля random. Створимо функцію game() у якій ми пропишемо логіку гри. Внизу створимо цикл, у якому помістимо запит до користувача, чи хоче він зіграти в гру, якщо так, то ми цей цикл припиняємо і викликає game(), щоб запустити гру, якщо ні – то припиняємо цикл і виводимо «Бувай», якщо щось інше введено, крім ні або так, то виводиться, що щось введено неправильно і тоді знову запитуємо користувача. Щодо логіки гри. У функції game() пишемо цикл в якому запитуємо про вибір гравця і формуємо випадковий вибір з масиву виборів – це буде вибір комп’ютера. І далі перевіряємо, якщо одинаковий – то нічия, якщо ж у гравця був камінь проти ножиць чи ножиці проти паперу, чи папер проти каменю, то гравець виграв, і аналогічну перевірку, коли вже навпаки комп’ютер виграв. Для цікавості гри я додав змінні, які відображають кількість перемог гравців. Далі я додав перевірку на вихід, якщо користувач захотів вийти. І якщо користувач увів щось зовсім не те, запит починається спочатку.

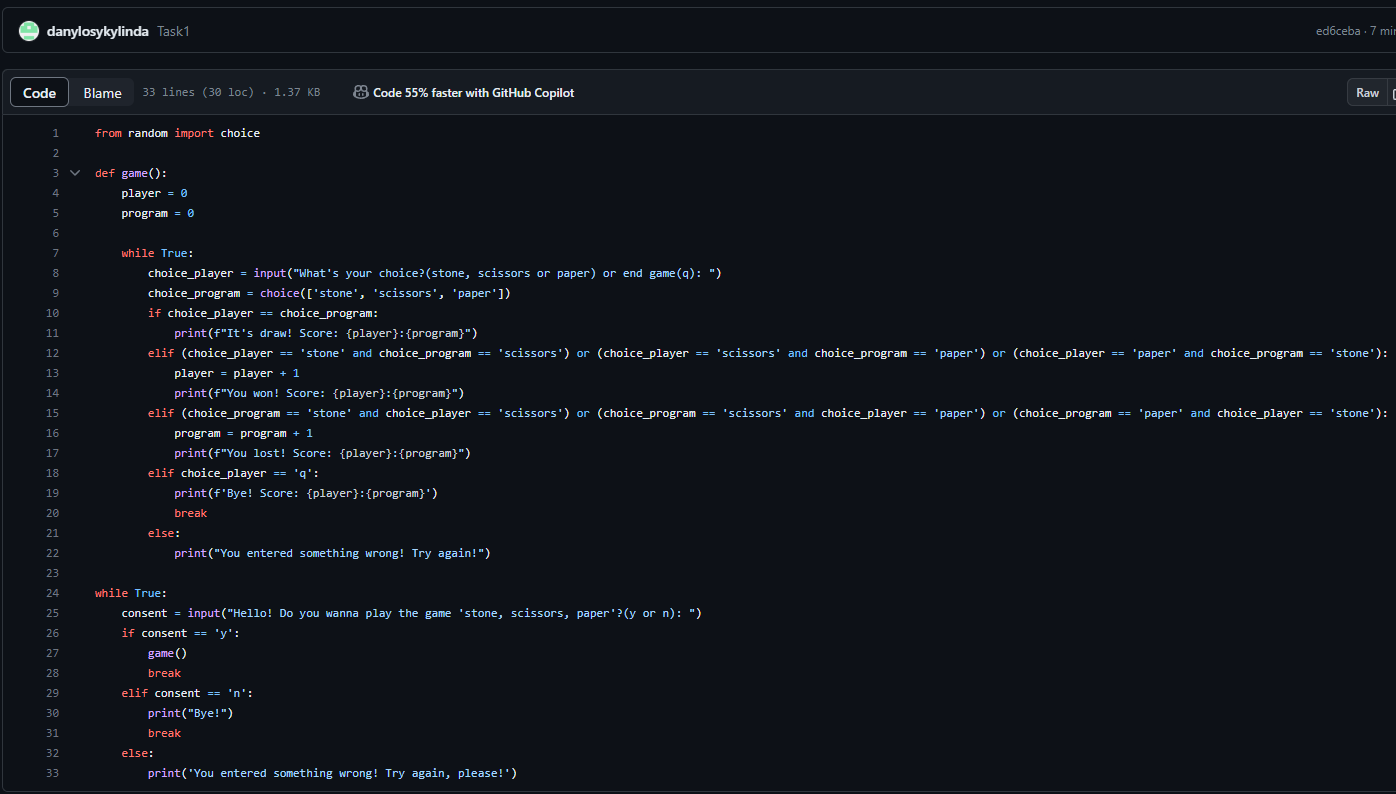
Код завдання:



GitHub:

Посилання на GitHub на цей файл: <https://github.com/danylosykylinda/TP-KB-221-Danylo-Sykylinda/blob/main/topic_05/task_1.py>

Скріншот коду з GitHub:



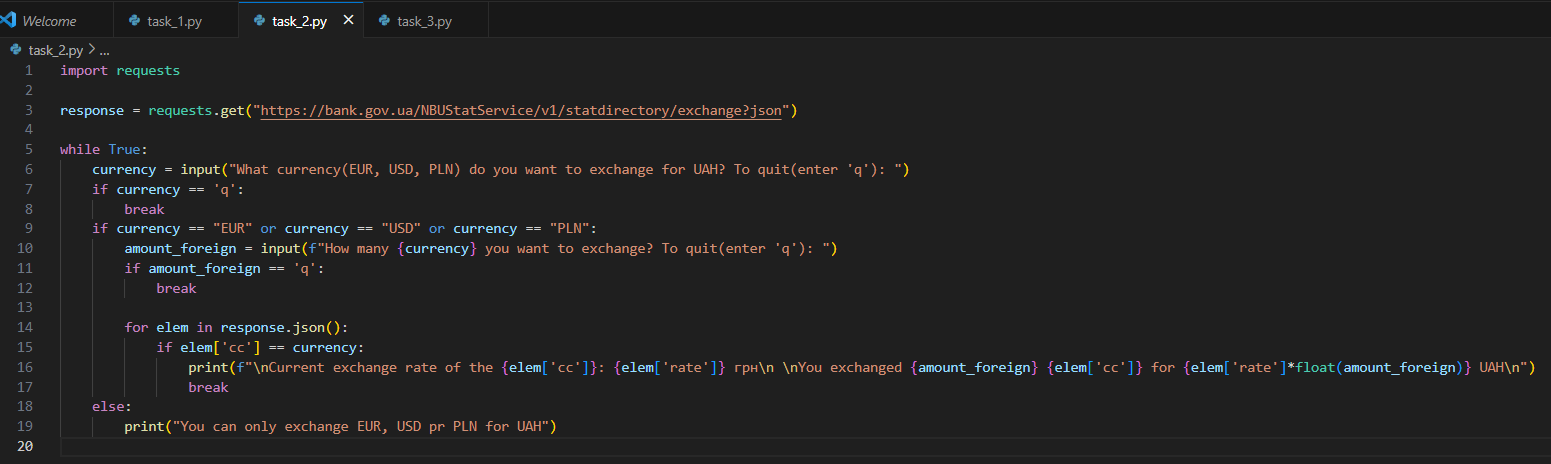
## Конвертер валют.

Завдання: Написати реалізацію для конвертера валют(долара, євро, злотих на гривні).

Виконання завдання:

Імпортуємо модуль requests, він нам знадобиться, щоб звертатися до серверу Нацбанку України для отримання даних про курс валют. Далі пишемо цикл, в якому пишемо два запити до користувача(яку валюту хоче на гривні поміняти користувач, яка кількість валюти). Далі перевіряємо, чи не хоче вийти користувач з програми, далі перевіряємо чи користувач обрав щось серед трьох валют, якщо так, питаємо скільки хочеш обміняти, далі через цикл for оброблюємо отримані дані від серверу, що зберігаються в змінній response, якщо знаходимо потрібну нам валюту, то обраховуємо і виводимо результат, цикл зупиняємо(оскільки в подальшому переборі немає необхідності). Якщо ж користувач увів щось інше чи іншу валют, окрім трьох, то виводиться повідомлення і у користувача знову запитуємо, яку валюту він хоче обміняти.

Код завдання:



GitHub:

Посилання на GitHub на цей файл: <https://github.com/danylosykylinda/TP-KB-221-Danylo-Sykylinda/blob/main/topic_05/task_2.py>

Скріншот коду з GitHub:



## Модулі для програми «Калькулятор».

Завдання: Розділити код програми «Калькулятор», що був створений в попередній темі на модулі operations.py(перевірку числа, додамо ще функцію на перевірку операцій), functions.py(функції дій(додавання, віднімання, множення, ділення), calc.py(цикл та імпорт модулів).

Виконання завдання:

Для виконання завдання створимо папку task\_3, щоб окремо виділити, що це третє завдання. Функцію, що виконують дії можна відразу перенести до файлу functions.py. Тут без змін і все очевидно. У файл operations.py переносимо код функції getNumber(), далі пишемо аналогічну до функції getNumber() функцію getOperations() для операцій. Тут все аналогічно, отримуємо запит, перевіряємо чи дійсно користувач увів операцію або q(вихід) і якщо так, то функція повертає операцію, якщо ж ні, то повідомлення про помилку і далі знову буде запит про операцію в користувача. У файл calc.py все, що залишилося від файлу з попередньої теми переносимо сюди. Далі у calc.py імпортуємо наші модулі operations, functions. Не забудемо внести виправлення, оскільки функції тепер викликаються з модулів, особливо потрібно в змінній для операції треба змінити код з input на виклик функції getOperations() з модуля operations, щоб передавати питання до функції, щоб робити запит.

Код завдання:

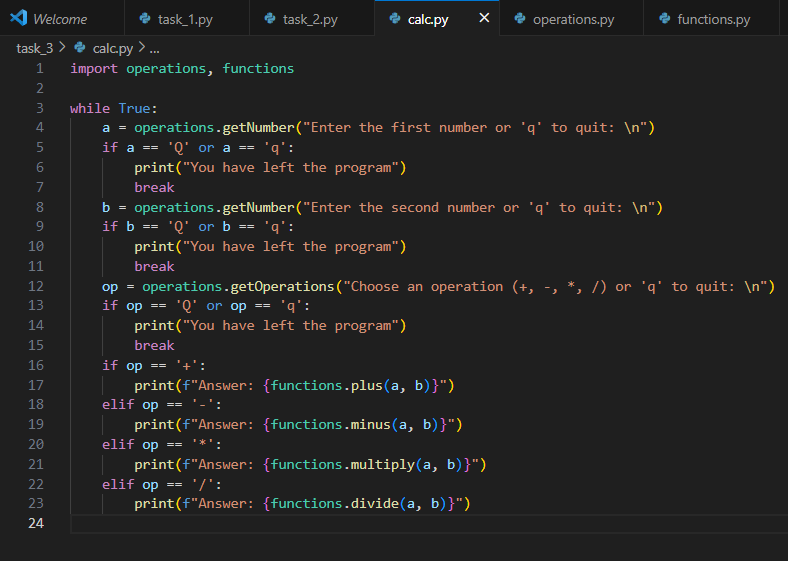


Рис. 1. – Файл calc.py.

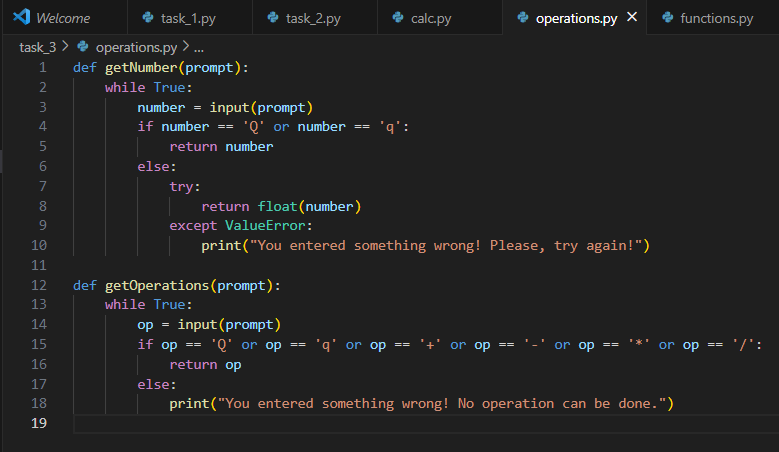


Рис. 2. – Файл operations.py.

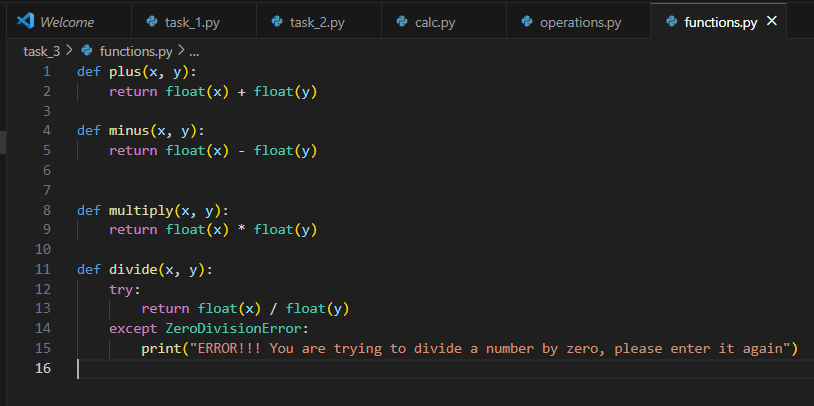


Рис. 3. – Файл functions.py.

GitHub:

Посилання на GitHub на папку цього завдання: <https://github.com/danylosykylinda/TP-KB-221-Danylo-Sykylinda/blob/main/topic_05/task_3>

Скріншот коду з GitHub:

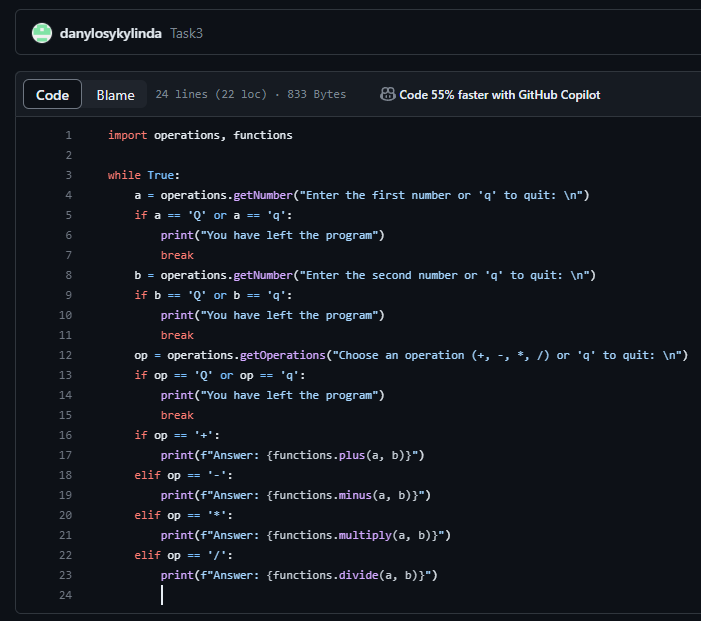


Рис. 4. – Файл calc.py.



Рис. 5. – Файл functions.py

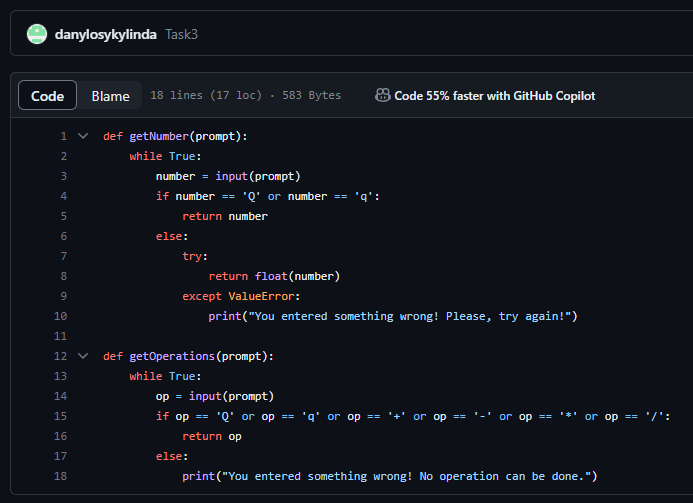


Рис. 6. – Файл operations.py.

# Звіт до теми №6

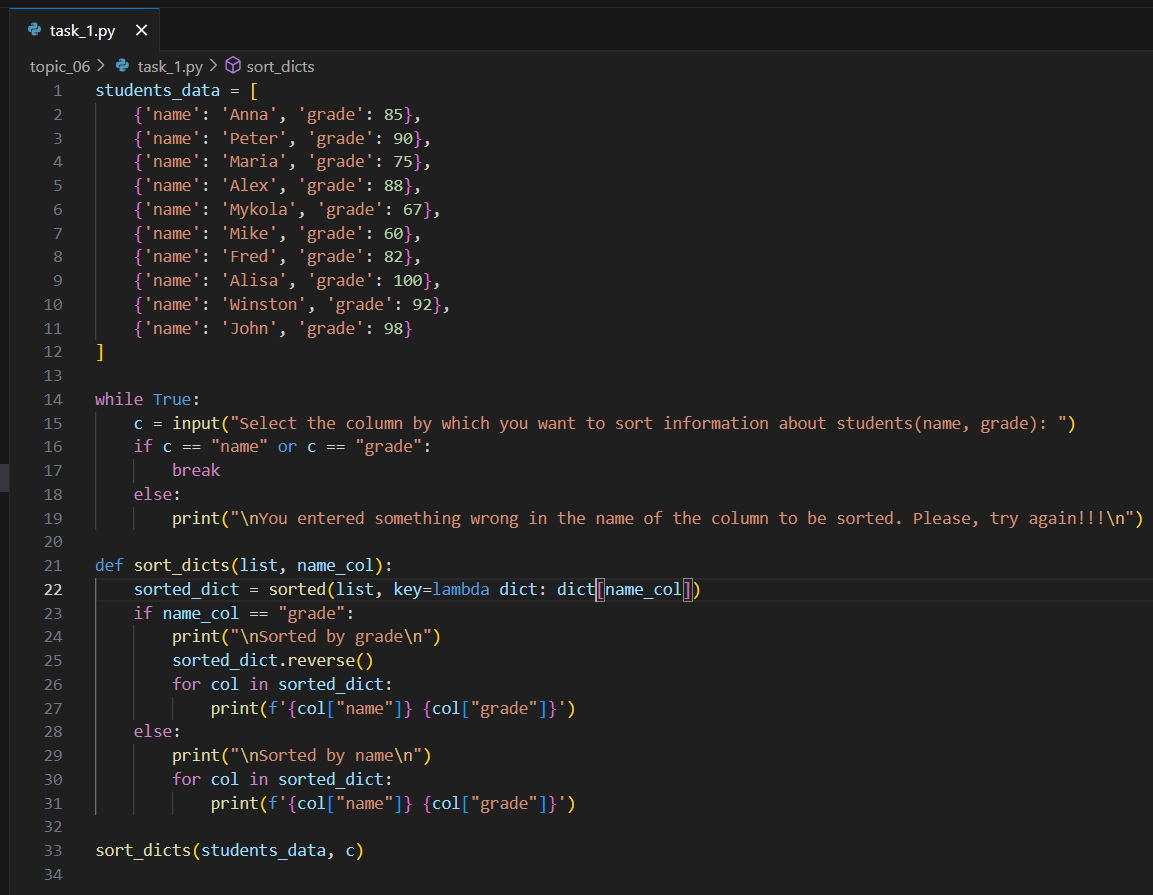
## Використання lamda-функції.

Завдання: Маючи невідсортований список, елементами якого є словники з двома параметрами (ім’я та оцінка) виконати сортування списку, використовуючи стандартну функцію sorted(). Другим параметром для функції sorted() має бути lambda функція, що повертає ім’я або оцінку із елемента словника.

Виконання завдання:

Маючи несортований список словників, в яких є імена та оцінки студентів, ми створимо функцію sort\_dicts() для сортування по певним полям, яка приймає список і назву колонки, по якій ми маємо сортувати. Список відсортовуємо вбудованою функцією sorted(), в якій ми вказуємо, який список сортувати, а другим аргументом вказуємо лямбду-функцію, яка приймає словник і повертає колонку по якій треба сортувати. І далі дивлячись, яка була обрана колонка по ній іде сортування, потім виводимо це через перебір циклом for і отримуємо результат. Ще також додав можливість вибору користувачем, яку колонку він хоче сортувати і відповідно його вибір передається в функцію sort\_dicts().

Код завдання:

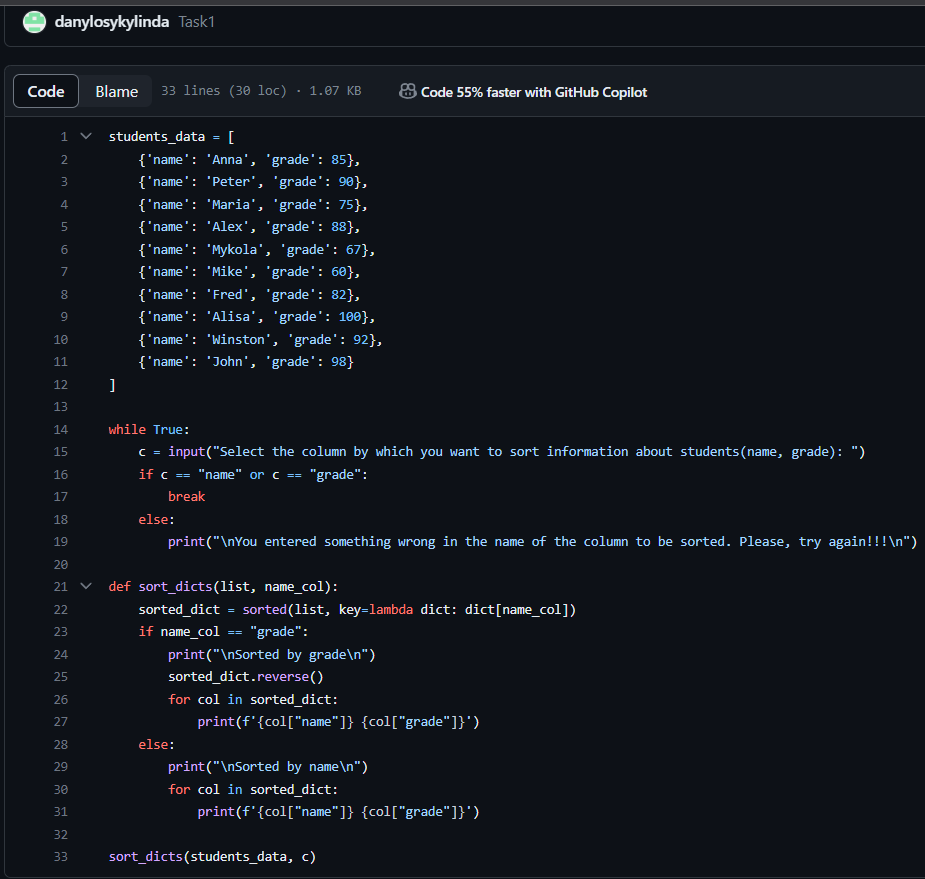


GitHub:

Посилання на GitHub на цей файл:

<https://github.com/danylosykylinda/TP-KB-221-Danylo-Sykylinda/blob/main/topic_06/task_1.py>

Скріншот коду з GitHub:



## Логування всіх дій у застосунку «Калькулятор».

Завдання: Необхідно розробити механізм логування всіх дій(у файл calc.log), що виконує програма «Калькулятор», також потрібно забезпечити зберігання інформацію про введені данні, виконану операцію та результат виконання операції над даними.

Виконання завдання:

Взянемо файли цієї програми з попередньої теми №5 завдання 5.3, помістимо їх в папку task\_2. Писати функцію для логування будемо в файлі operations.py. Для початку підвантажимо модуль datetime, щоб отримувати поточний час для запису логу. Функцію назвемо log, яка прийматиме чотири параметри: перше число, друге число, операція, результат. Я призначив значення None параметрам за замовчуванням, оскільки може бути ситуація, коли користувач вийшов з програми, тоді не отримаємо всіх параметрів. Для початку відслідкуємо функції (getNumber та getOperations), де відбуваються перевірки на помилки і викличемо функцію log і передамо одне значення, яке означатиме певну помилку(err\_num – неправильний ввід чисел і err\_op – неправильний ввід під час введення операції). Повернемося до нашої функції і напишемо, що будемо робити, коли та чи інша помилка станеться. Для початку у змінну time буде записуватися поточний час і також використаємо метод strftime() для того, щоб виправити положення чисел в даті з американського формату в український формат часу. Далі, якщо в перший аргумент дорівнює err\_num, то відкриваємо файл calc.log з флагом «а»(для запису на кінець файлу) і передаємо змінну із повідомленням про помилку з часом, коли це сталося на запис в файл. Потім якщо сталася помилка err\_op, робимо аналогічне як з попереднім випадком, повідомлення інше. Якщо користувач увів q або Q, то ми робимо аналогічно з попереднім випадком, записуємо в файл calc.log з відповідним повідомленням, що користувач покинув програму. А якщо ні те і ні інше(користувач щось обрахував і отримав результат), то ми використовуємо всі отримані параметри і формуємо повідомлення записуємо в файл. До того ж, якщо користувач якесь число поділить на нуль, то це відобразиться тут, але в лог файлу буде записано, наприклад: користувач увів а=3 і b=0, і операцію «/», отримав результат помилка. Щоб так виводилося у файлі functions.py я зробив зміну коду в останній стрічці коду в функції divide() з print() на return, тобто функція повертає повідомлення про помилку. Це логічніше, оскільки це помилка не через введення чогось неправильного, а математична дія - ділення на нуль, яка немає результату. Тепер не забудемо в основному файлі програми calc.py додати виклик функції log() і передати необхідні в неї параметри, де це нам потрібно в коді.

Код завдання:

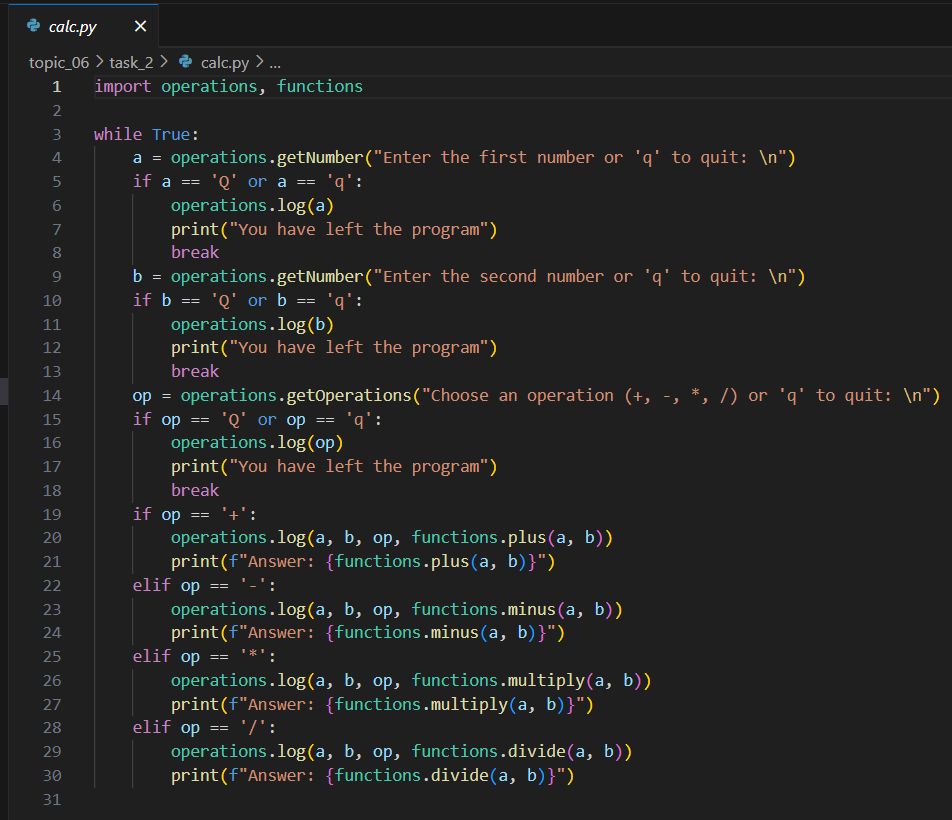


Рис. 1. – Файл calc.py.

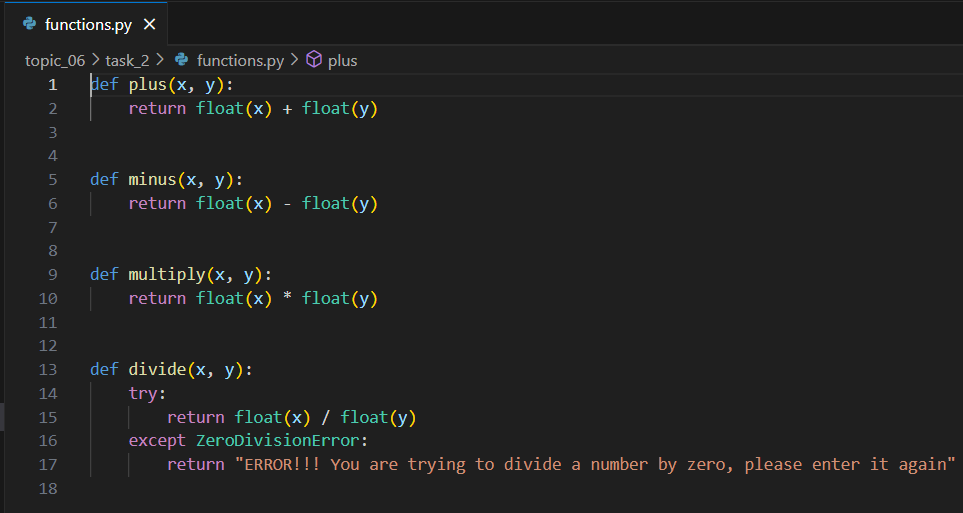


Рис. 2. – Файл functions.py

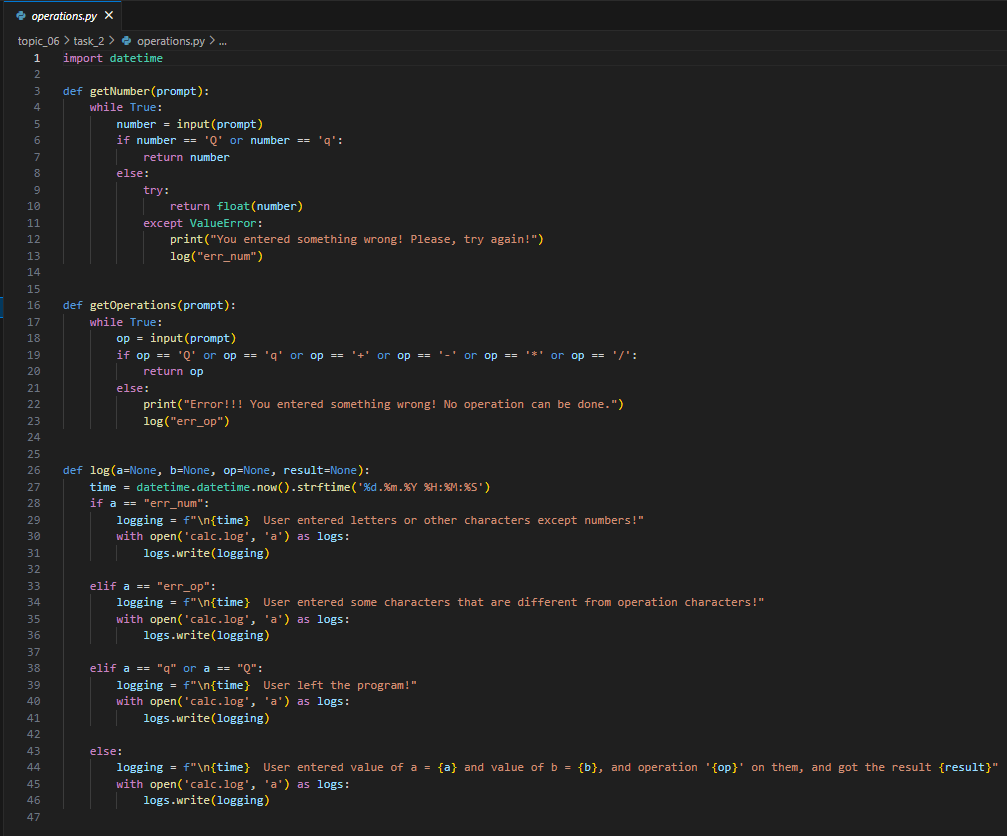


Рис. 3. – Файл operations.py.

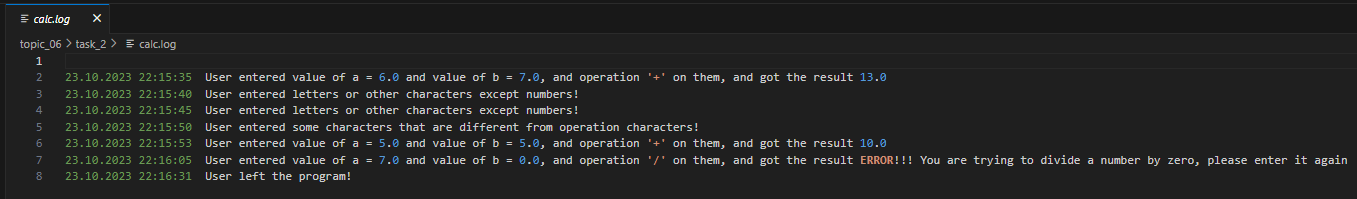


Рис. 4. – Файл calc.log (логи програми).

GitHub:

Посилання на GitHub на папку цього завдання: <https://github.com/danylosykylinda/TP-KB-221-Danylo-Sykylinda/blob/main/topic_06/task_2>

Скріншот коду з GitHub:

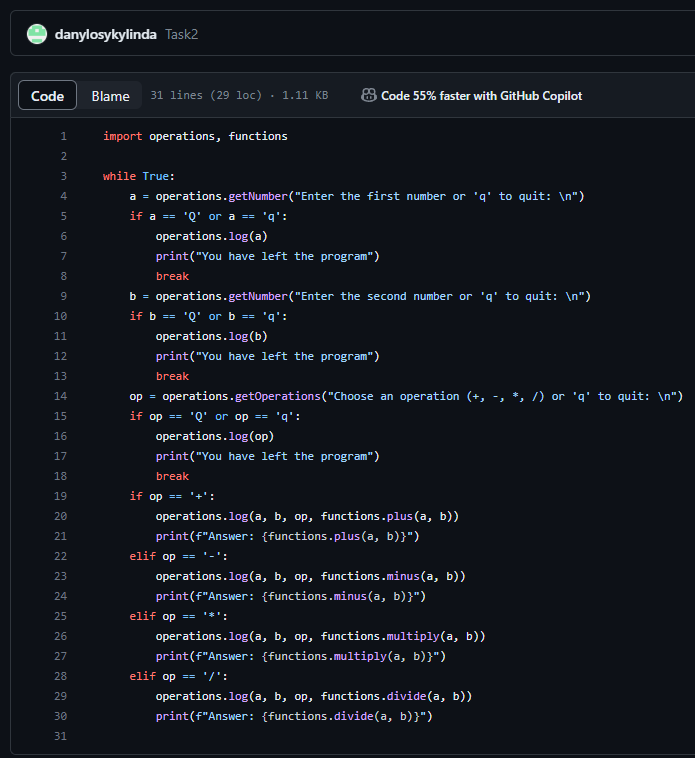


Рис. 5. – Файл calc.py.



Рис. 6. – Файл functions.py

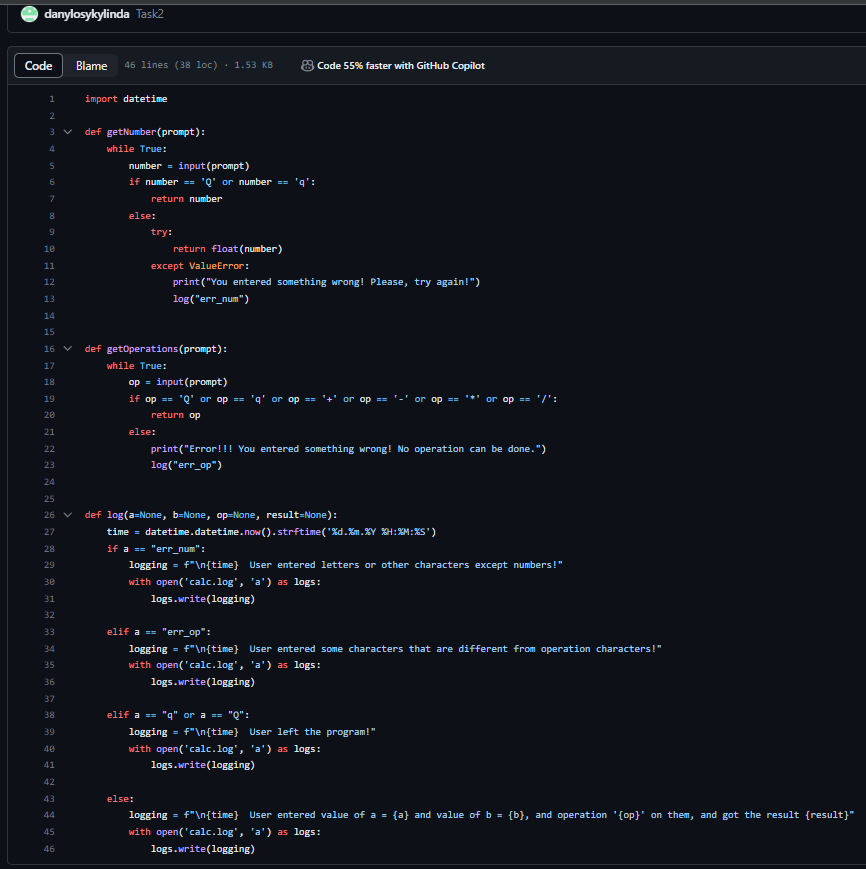


Рис. 7. – Файл operations.py.

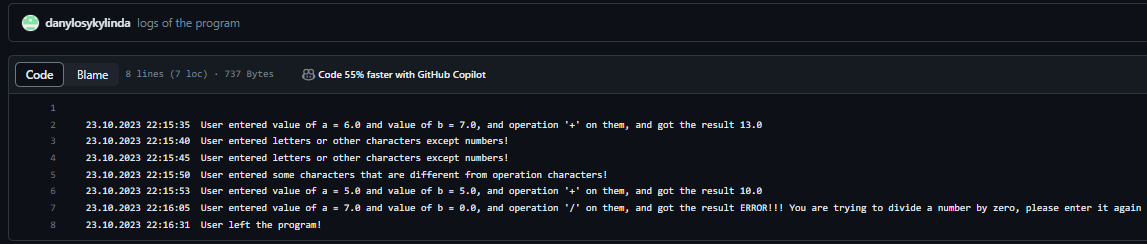


Рис. 8. – Файл calc.log (логи програми).

# Звіт до теми №7

## Перегляд документації про Class.

Завдання: Ознайомитися більше з конструкцією Class в мові Python.

Виконання завдання:

Я детальніше ознайомився з Class та його методами: \_\_init\_\_, \_\_str\_\_, \_\_repr\_\_, а також для чого потрібен self.

## Перегляд документації про \_\_func\_\_.

Завдання: Ознайомитися більше з \_\_func\_\_ в мові Python.

Виконання завдання:

Я дізнався, що атрибут \_\_func\_\_ є посиланням на оригінальну функцію в класі.

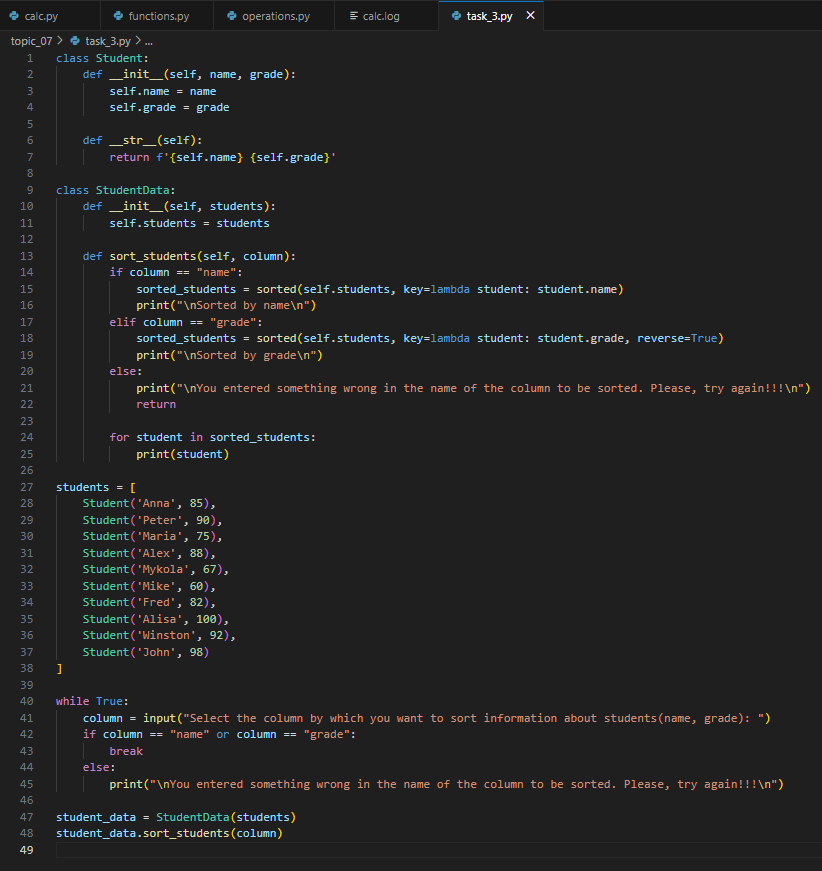
## Перепис коду програми із завдання 6.1. (про сортування списку і ламбду-функцію) із використанням ООП.

Завдання: Переписати код із завдання 6.1. з використанням ООП.

Виконання завдання:

Я написав клас Student, далі у конструкторі створив два атрибути name і grade. Потім прописав метод \_\_str\_\_ для того, що потім готові дані представляти в строку. Далі створив клас StudentData, додав атрибут students, у цьому класі сюди переніс функцію sort\_students() і дещо оптимізував її. Тоді далі переписав список студентів, кожного ініціалізував. Після цього цикл із питанням, яку колонку сортувати. І тоді вже екземпляр класу StudentData, передаю йому список студентів, і далі викликаю функцію sort\_students із класу і передаю їй колонку, по якій треба зробити сортування.

Код завдання:

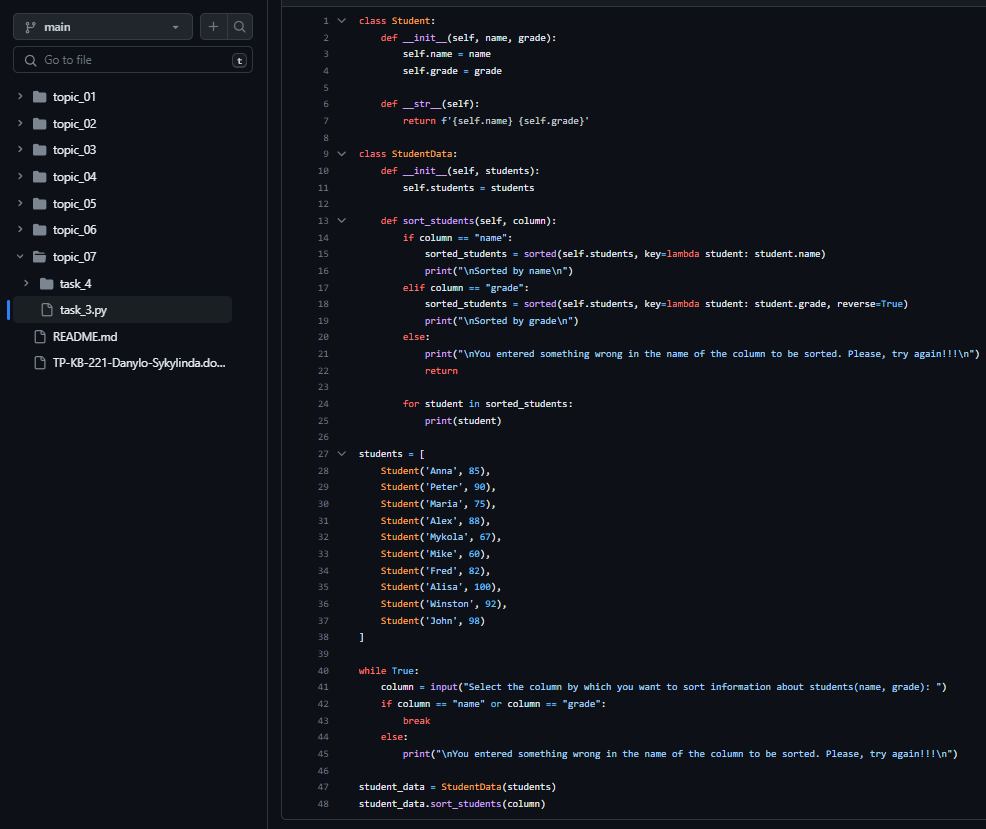


GitHub:

Посилання на GitHub на цей файл:

<https://github.com/danylosykylinda/TP-KB-221-Danylo-Sykylinda/blob/main/topic_07/task_3.py>

Скріншот коду з GitHub:



## Переписати код програми «Калькулятор» за допомогою ООП.

Завдання: Переписати код програми «Калькулятор» з попередніх тем з використанням ООП.

Виконання завдання:

Спочатку за допомогою ООП я переписав файл functions.py. Я створив клас Calculator і переніс в нього попередній код та адаптував його під клас. Далі в файлі operations.py зробив аналогічне: створив клас Operations і переніс в нього попередній код, що був в файлі, адаптував код під клас. Потім в основному файлі calc.py я ініціалізував класи Calculator і Operations, і далі викликав їхні функції в коді, де це було потрібно. Після цього перевірив, чи все працює.

Код завдання:

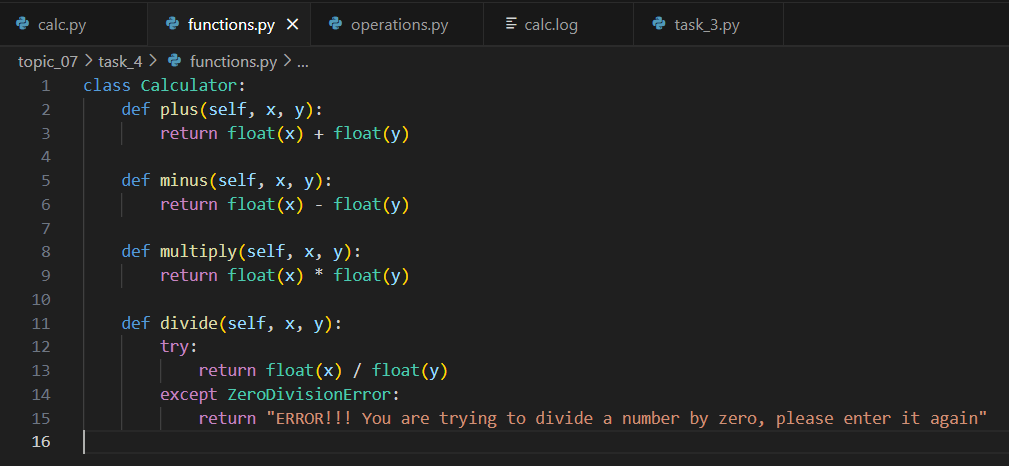


Рис. 1. – Код файлу functions.py.

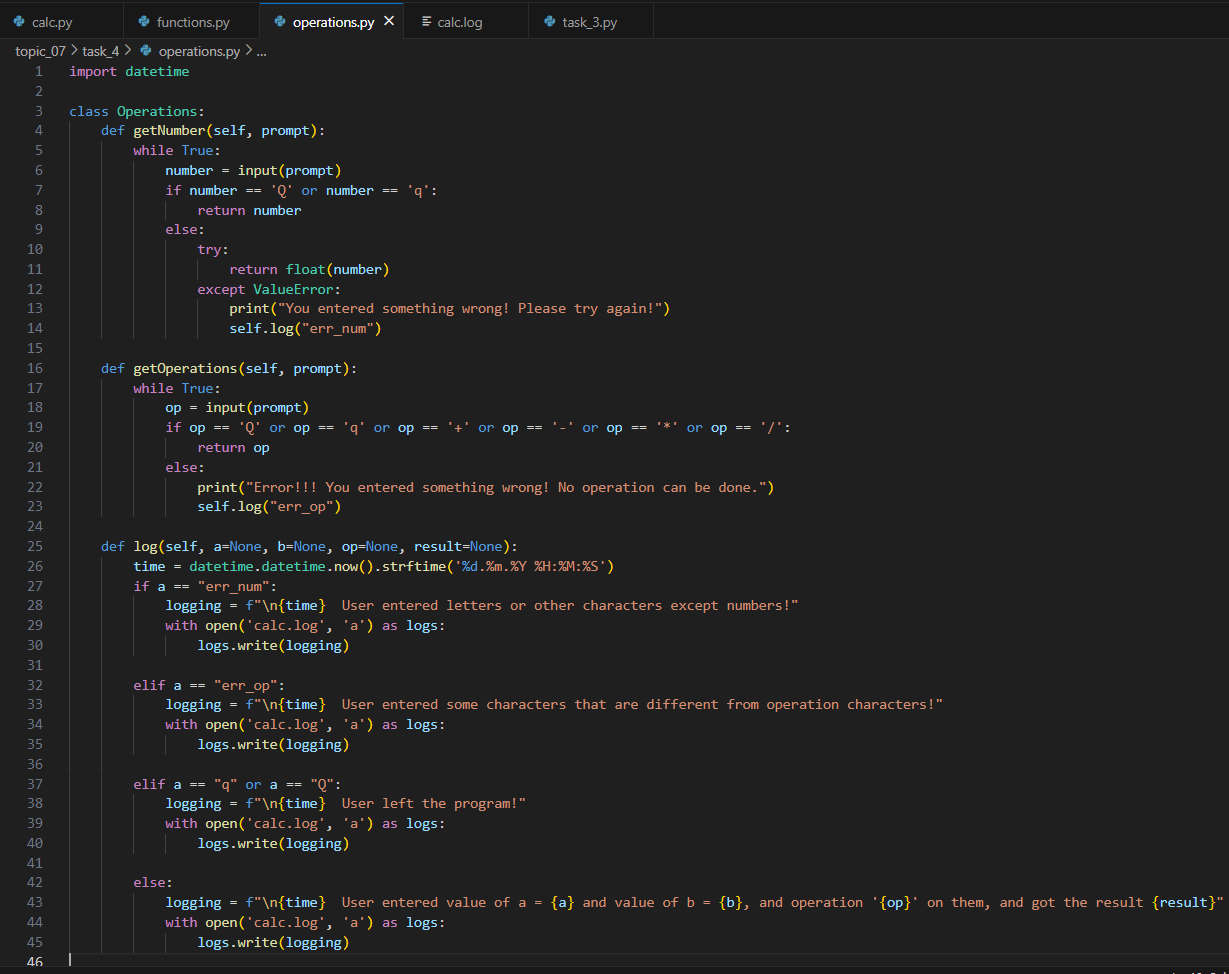


Рис. 2. – Код файлу operations.py.

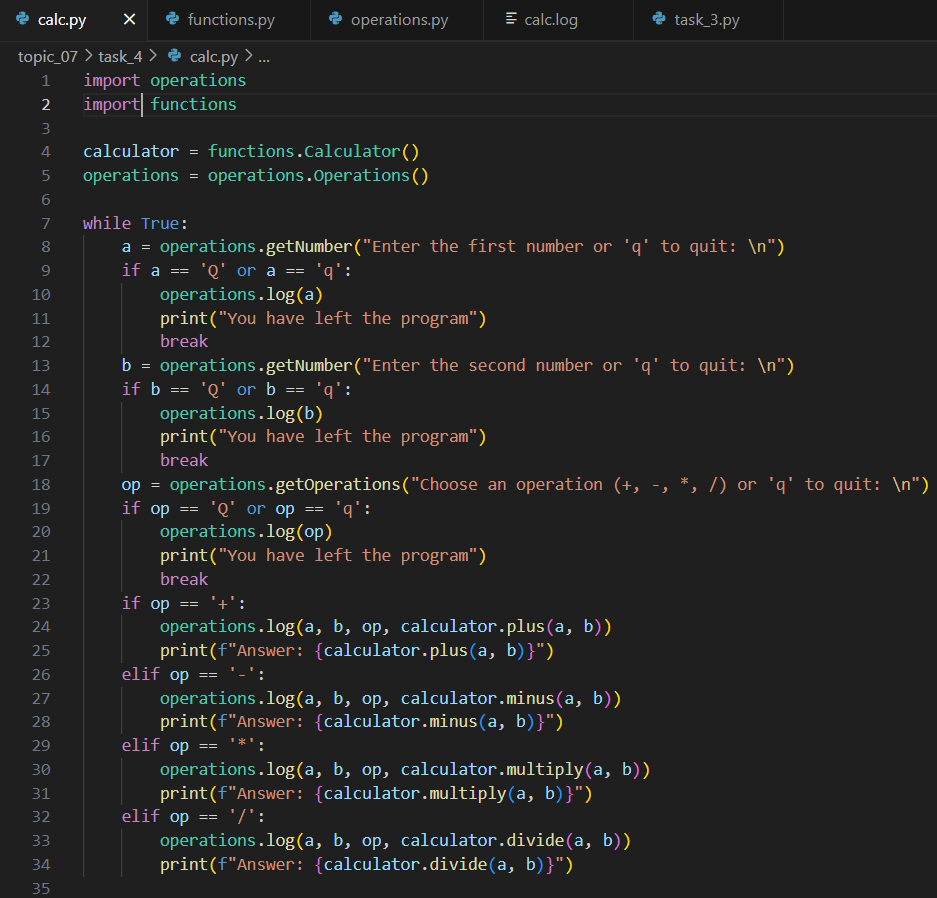


Рис. 3. – Код файлу calc.py.

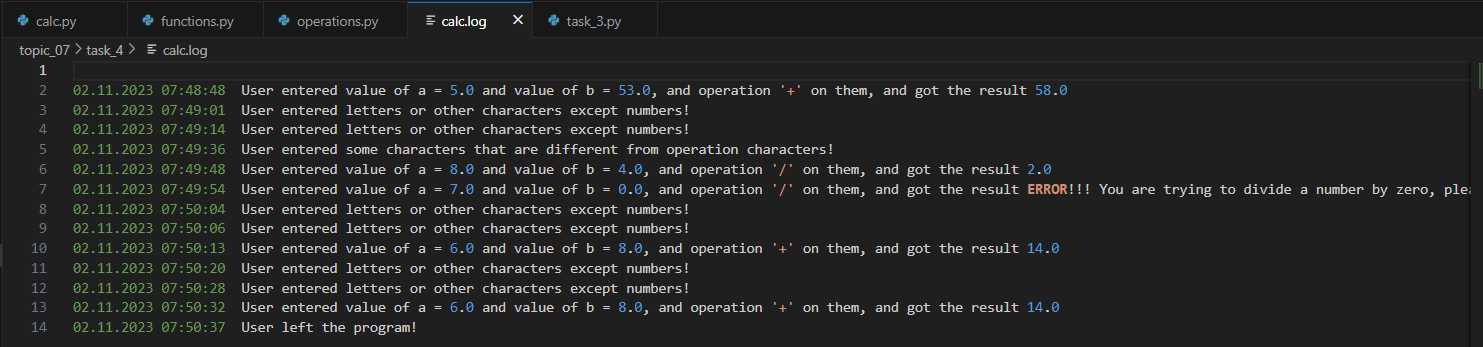


Рис. 4. – Вміст файлу calc.log (логи програми).

GitHub:

Посилання на GitHub на папку цього завдання: <https://github.com/danylosykylinda/TP-KB-221-Danylo-Sykylinda/blob/main/topic_07/task_4>.

Скріншот коду з GitHub:

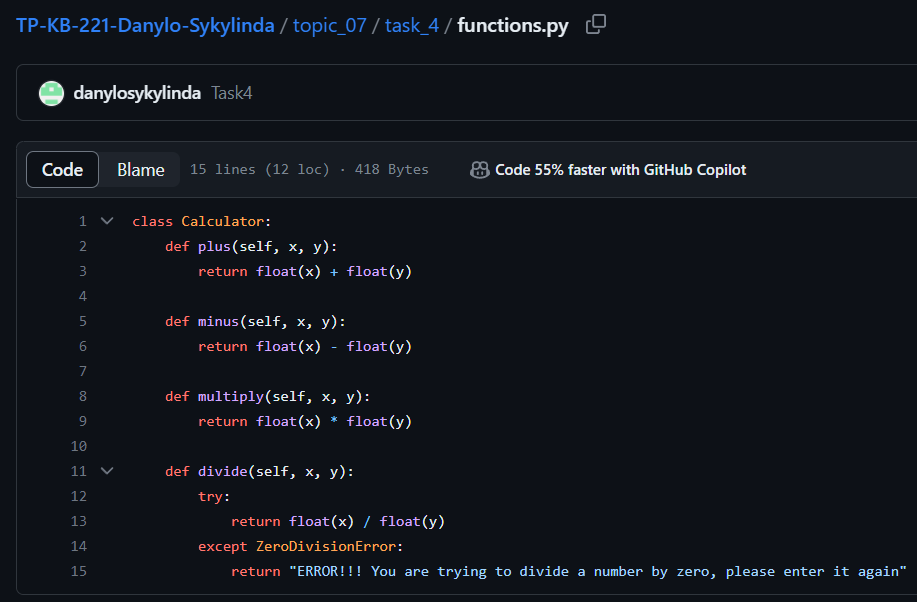


Рис. 5. Код файлу functions.py.

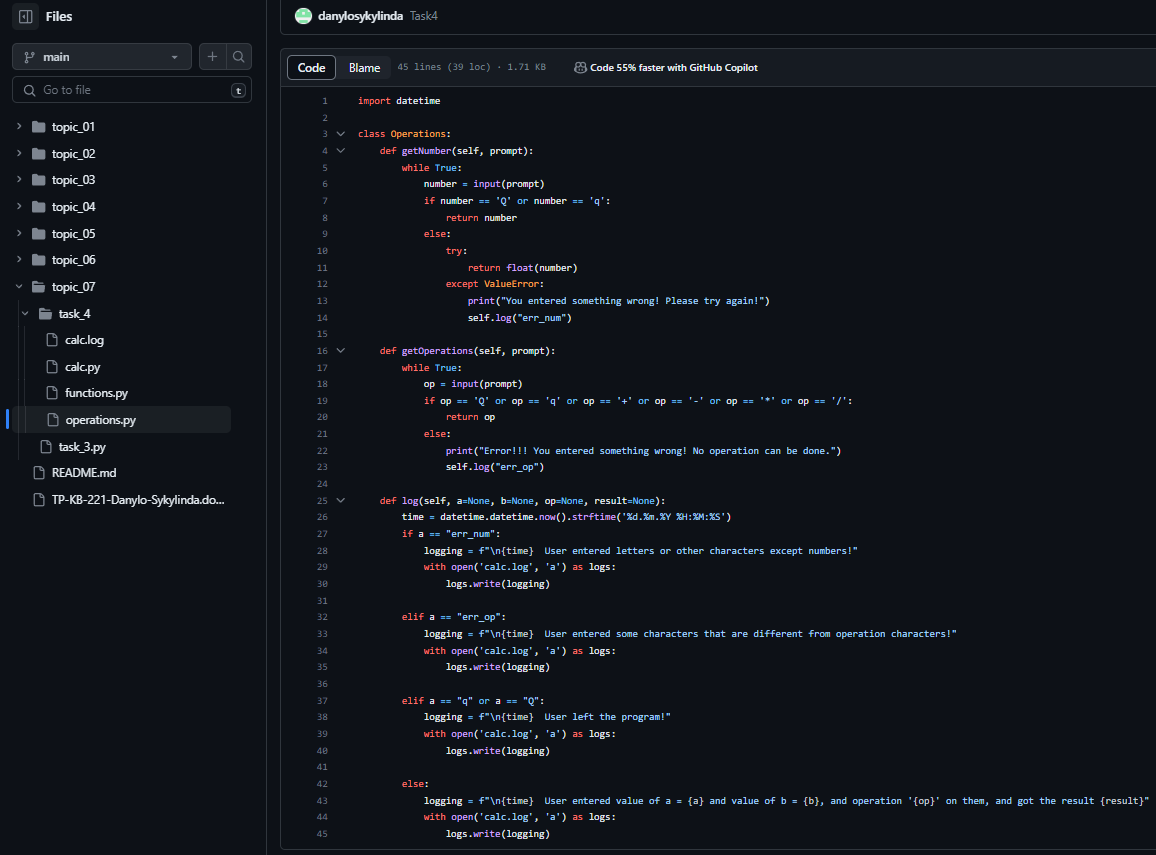


Рис. 6. Код файлу operations.py.

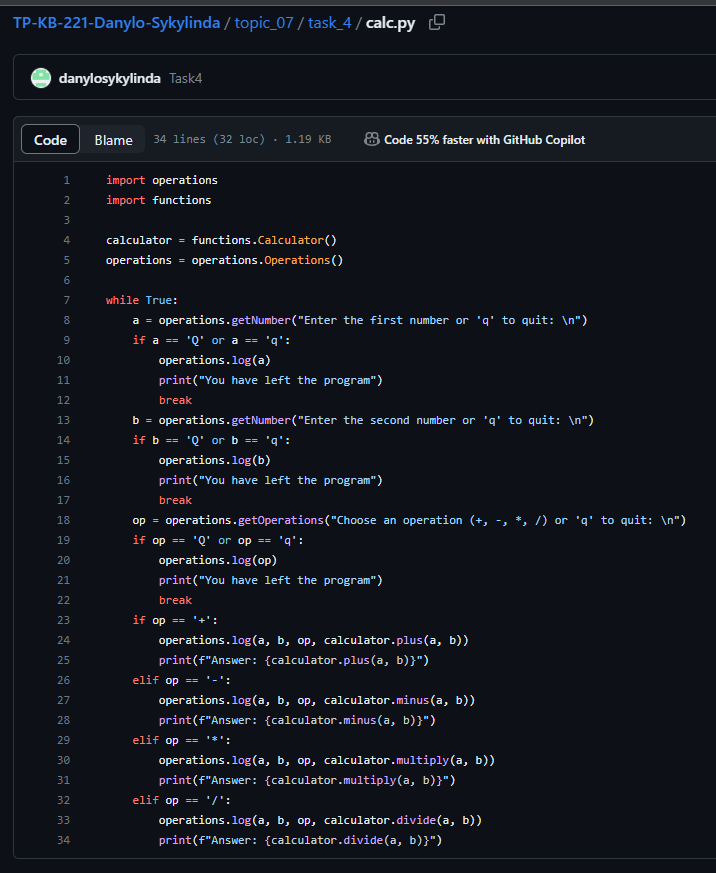


Рис. 7. Код файлу calc.py.

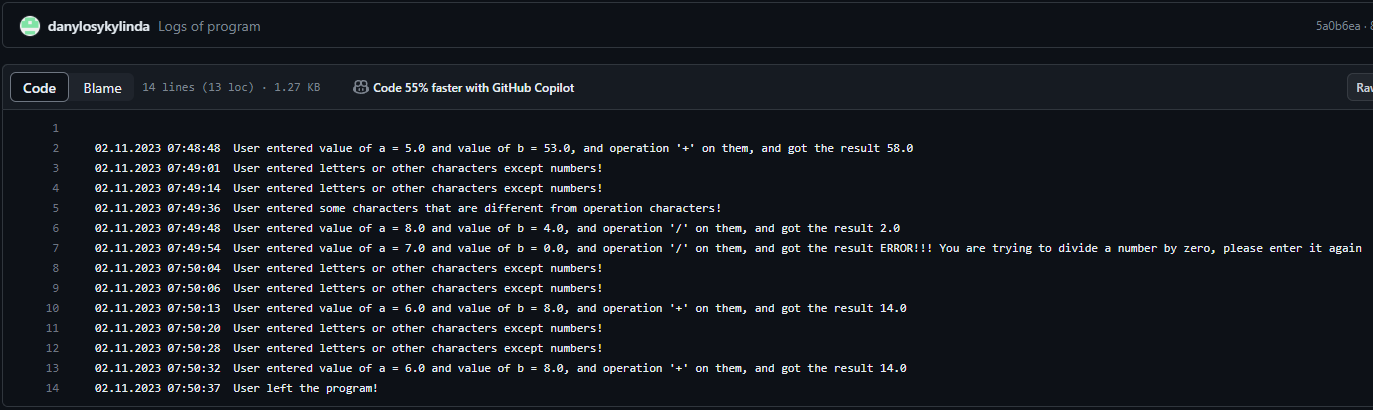


Рис. 8. Вміст файлу calc.log (логи програми).