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

Звіт до Теми №1

Функції та змінні

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

**1.Перетворення рядка**

Необхідно рядок, що має вигляд "qsdafw12" перетворити наступним чином "21qwfadsq", вважаючи сталою довжину рядку в 10 символів.

Хід виконання завдання:

Цей код виконує операцію перевертання рядка. Ось пояснення крок за кроком:

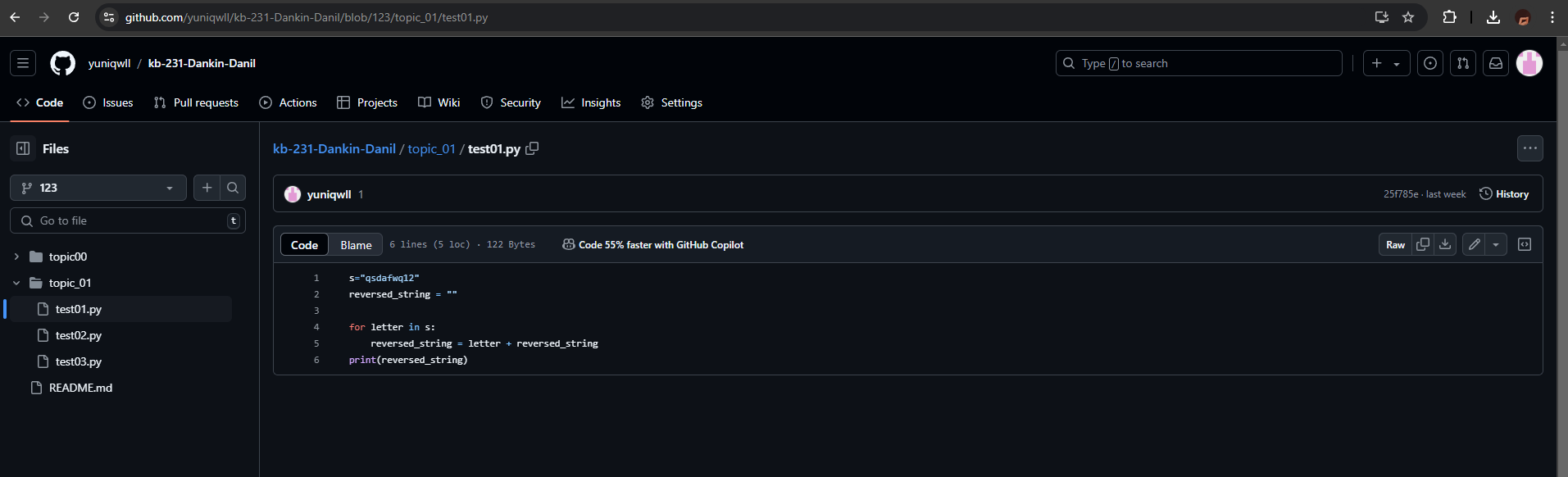
1. Початковий рядок s дорівнює "qsdafwq12"
2. Порожній рядок reversed\_string створюється для збереження результату.
3. Для кожної букви (або символа) в рядку s виконується наступне: Символ додається на початок рядка reversed\_string. Це означає, що кожен новий символ стає перед попередніми, поступово формуючи рядок у зворотному порядку.
4. Робота циклу:На першому кроці циклу змінна letter дорівнює "q", і значення reversed\_string стане "q".

На наступному кроці змінна letter дорівнюватиме "s", і рядок reversed\_string стане "sq". Цей процес продовжується для кожного символу.

Текст програми:

|  |
| --- |
| s="qsdafwq12"  reversed\_string = ""  for letter in s:      reversed\_string = letter + reversed\_string  print(reversed\_string) |

Посилання на github: https://github.com/yuniqwll/kb-231-Dankin-Danil/blob/123/topic\_01/test01.py

Знімок екрану з посилання на github: 

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

Хід виконання завдання:

1. Початковий рядок s дорівнює "Привіт світ ! " (з пробілом на початку та в кінці).
2. Метод strip() видаляє всі зайві пробіли на початку та в кінці рядка.
3. Метод capitalize() робить першу літеру всього рядка великою, а всі інші – малими.
4. Метод title() перетворює кожне слово в рядку так, що перша літера кожного слова стає великою, а інші – малими.
5. Метод upper() переводить всі літери в рядку в верхній регістр (великі літери).
6. Метод lower() переводить всі літери в нижній регістр (малі літери).

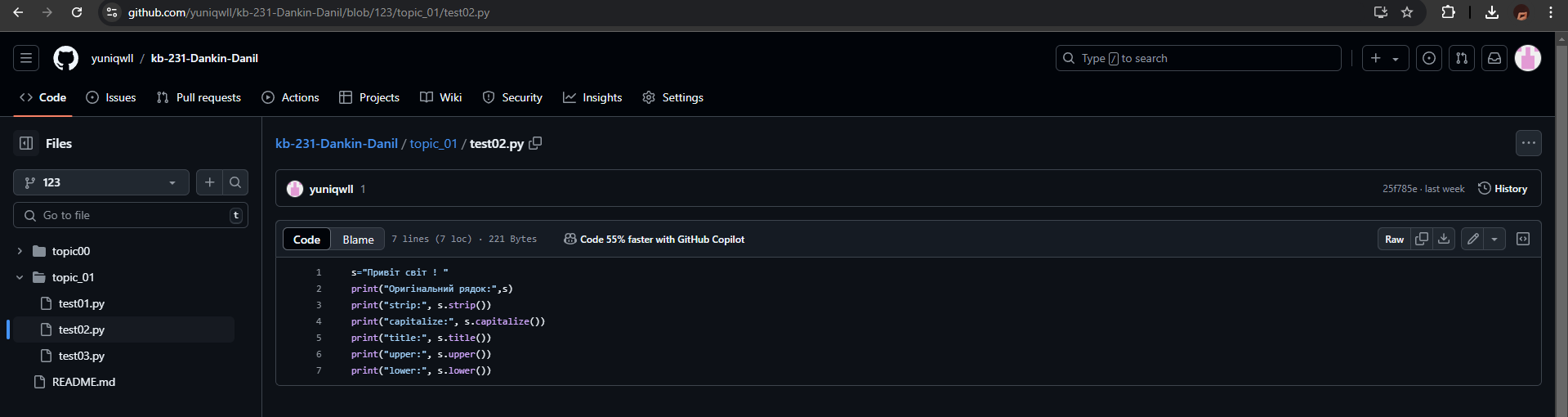
Цей код демонструє роботу з рядками за допомогою кількох вбудованих методів Python для зміни регістру літер і роботи з пробілами.

Текст програми: Текст програми:

|  |
| --- |
| s="Привіт світ ! "  print("Оригінальний рядок:",s)  print("strip:", s.strip())  print("capitalize:", s.capitalize())  print("title:", s.title())  print("upper:", s.upper())  print("lower:", s.lower()) |

Посилання на github: <https://github.com/yuniqwll/kb-231-Dankin-Danil/blob/123/topic_01/test02.py>

Знімок екрану з посилання на github:



**3. Написати функцію пошуку дискримінанту квадратного рівняння.**

Хід виконання завдання:

1) **Оголошення функції findD(a, b, c)**:

Функція приймає три аргументи: a, b та c, які є коефіцієнтами квадратного рівняння.

Усередині функції виконується обчислення дискримінанта за формулою D=b2−4acD = b^2 - 4acD=b2−4ac.

Оператор return повертає значення дискримінанта

2) **Ведення значень**:

За допомогою функції input() програма запитує у користувача значення коефіцієнтів:

a — для коефіцієнта при x^2,

b — для коефіцієнта при x,

c — вільного члена.

Введені значення перетворюються на цілі числа за допомогою функції int()

3) **Обчислення дискримінанта**:

Функція findD(a, b, c) викликається із переданими значеннями коефіцієнтів, і обчислюється дискримінант

Обчислене значення дискримінанта виводиться на екран за допомогою функції print().

Текст програми:

def findD(a,b,c):

    return b\*\*2-4\*a\*c

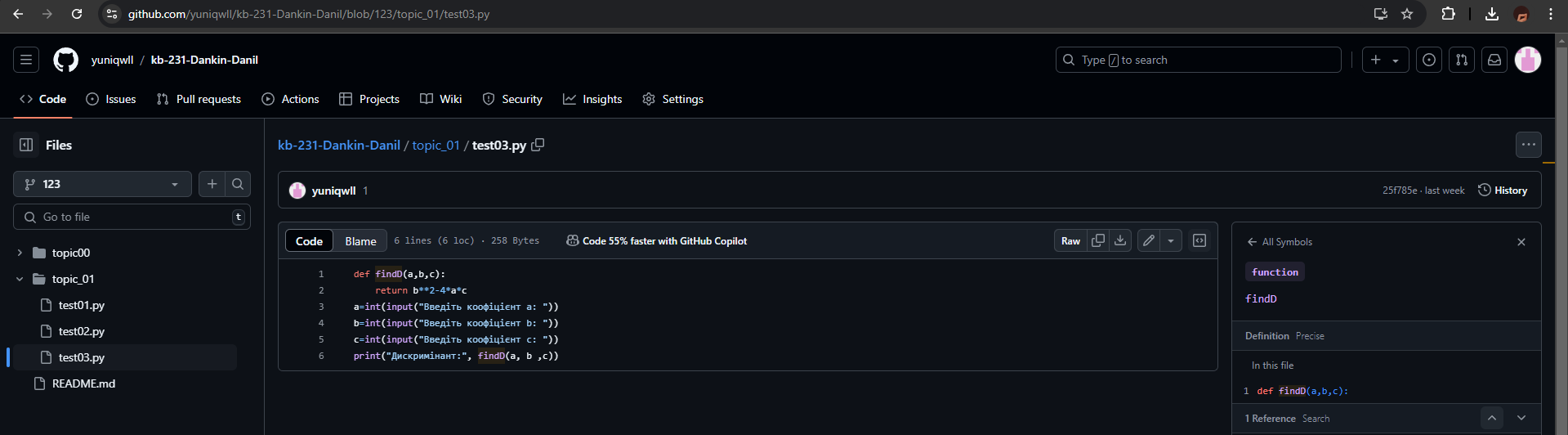
a=int(input("Введіть коофіцієнт a: "))

b=int(input("Введіть коофіцієнт b: "))

c=int(input("Введіть коофіцієнт c: "))

print("Дискримінант:", findD(a, b ,c))

Посилання на github: <https://github.com/yuniqwll/kb-231-Dankin-Danil/blob/123/topic_01/test03.py>

Знімок екрану з посилання на github: 

Звіт до Теми №2

Умовний перехід

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

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

Хід виконання завдання:

1. Імпорт бібліотеки:

Імпортується бібліотека math, щоб використовувати функцію sqrt() для обчислення квадратного кореня дискримінанта.

1. Додаємо функцію для обчислення дискримінанту

Ця функція приймає на вхід коефіцієнти a, b та c і повертає дискримінант, обчислений за формулою:

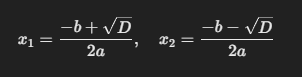
D=b\*\*2−4\*a\*c

1. Додаємо функцію для знаходження коренів

Спочатку у цій функції обчислюється дискримінант D за допомогою попередньо визначеної функції discriminant.

Далі код перевіряє значення дискримінанта:

**Якщо D>0**, у рівняння два різні дійсні корені. Ці корені обчислюються за формулами:



**Якщо** D=0, рівняння має один дійсний корінь. Він обчислюється за формулою:



**Якщо** D<0, дійсних коренів немає, і виводиться відповідне повідомлення.

1. Введення коофіцієнтів і запуск функції

Користувач вводить значення коефіцієнтів a, b і c через консоль. Після цього викликається функція find\_roots, яка розв'язує рівняння.

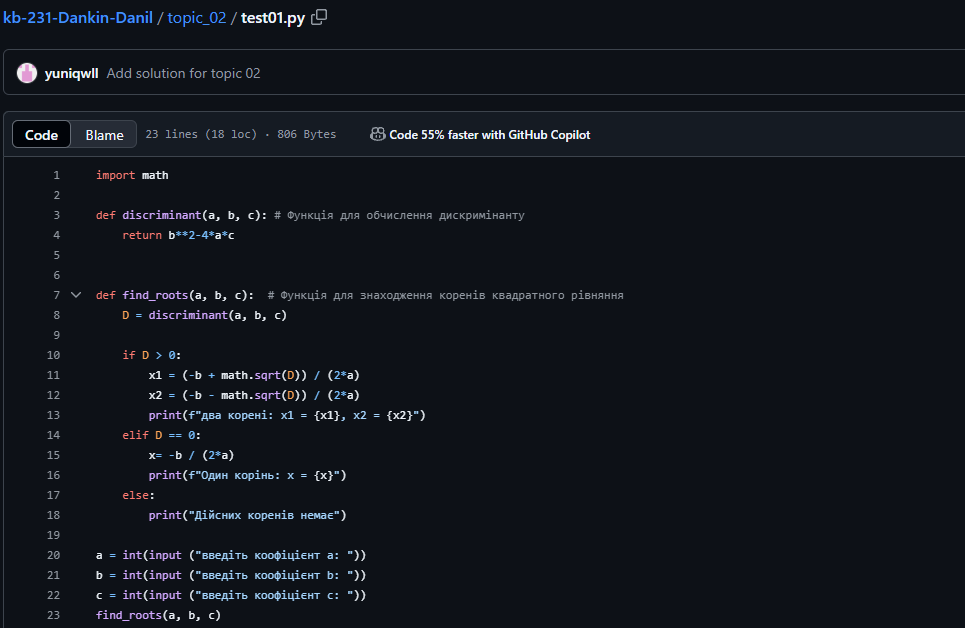
Текст програми:

|  |
| --- |
| import math  def discriminant(a, b, c): # Функція для обчислення дискримінанту      return b\*\*2-4\*a\*c  def find\_roots(a, b, c):  # Функція для знаходження коренів квадратного рівняння      D = discriminant(a, b, c)      if D > 0:          x1 = (-b + math.sqrt(D)) / (2\*a)          x2 = (-b - math.sqrt(D)) / (2\*a)          print(f"два корені: x1 = {x1}, x2 = {x2}")      elif D == 0:          x= -b / (2\*a)          print(f"Один корінь: x = {x}")      else:          print("Дійсних коренів немає")  a = int(input ("введіть коофіцієнт a: "))  b = int(input ("введіть коофіцієнт b: "))  c = int(input ("введіть коофіцієнт с: "))  find\_roots(a, b, c) |

Посилання на github:

https://github.com/yuniqwll/kb-231-Dankin-Danil/blob/123/topic\_02/test01.py

Знімок екрану з посилання на github:



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

Хід виконання завдання:

**1)Функції для арифметичних операцій**:

Кожна функція виконує одну з арифметичних операцій над двома числами: додає, віднмає, перемножує та ділить

**2)Основна програма калькулятора**: Функція colculator() отримує від користувача два числа і обрану арифметичну операцію. Далі вона виконує відповідну арифметичну операцію, використовуючи функції, що були описані раніше. Цей блок отримує два числа від користувача та операцію, яку потрібно виконати.

Далі йде блок умов if-elif-else, який перевіряє, яку саме операцію обрав користувач

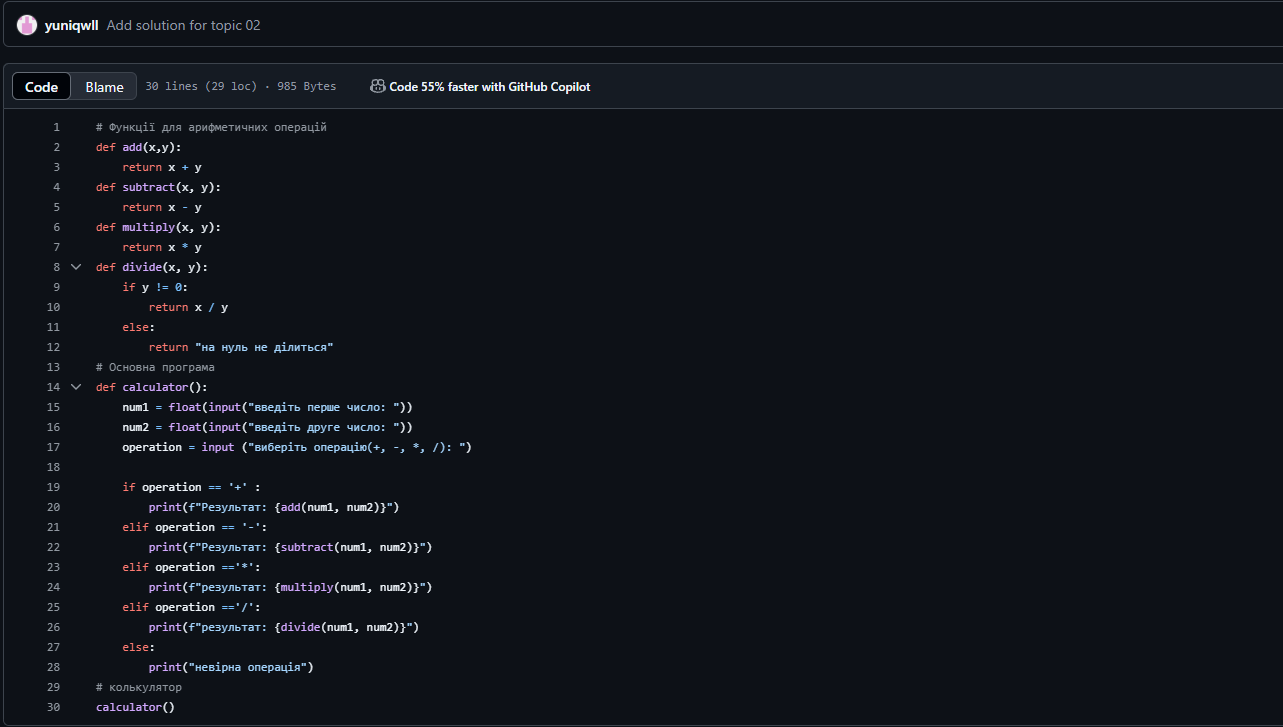
Текст програми:

|  |
| --- |
| # Функції для арифметичних операцій  def add(x,y):      return x + y  def subtract(x, y):      return x - y  def multiply(x, y):      return x \* y  def divide(x, y):      if y != 0:          return x / y      else:          return "на нуль не ділиться"  # Основна програма  def calculator():      num1 = float(input("введіть перше число: "))      num2 = float(input("введіть друге число: "))      operation = input ("виберіть операцію(+, -, \*, /): ")      if operation == '+' :          print(f"Результат: {add(num1, num2)}")      elif operation == '-':          print(f"Результат: {subtract(num1, num2)}")      elif operation =='\*':          print(f"результат: {multiply(num1, num2)}")      elif operation =='/':          print(f"результат: {divide(num1, num2)}")      else:          print("невірна операція")  # колькулятор  calculator() |

Посилання на github:

https://github.com/yuniqwll/kb-231-Dankin-Danil/blob/123/topic\_02/test02.py

Знімок екрану з посилання на github:



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

**Функції для арифметичних операцій**: У коді є чотири функції, кожна з яких відповідає за виконання певної арифметичної операції.

1)Додавання:

:def add(x,y):

    return x + y

Функція приймає два аргументи x і y та повертає їх суму.

2)Віднімання:

def subtract(x, y):

    return x - y

Функція приймає xxx і yyy і повертає результат віднімання y від x.

3)Множення

def multiply(x, y):

    return x \* y

Функція повертає результат множення x на y

4)Ділення

def divide(x, y):

    if y != 0:

        return x / y

    else:

        return "на нуль не ділиться"

Функція перевіряє, чи не є yyy нулем. Якщо так, вона повертає повідомлення про помилку ("на нуль не ділиться"). Якщо yyy не дорівнює нулю, виконується ділення xxx на yyy.

**Основна програма**: У функції calculator() реалізовано основну логіку програми.

Введення чисел та вибір операції:

def calculator():

    num1 = float(input("Введіть перше число: "))

    num2 = float(input("Введіть друге число: "))

    operation = input("Виберіть операцію (+, -, \*, /): ")

Користувач вводить символ операції (+, -, \*, /).

**Використання конструкції match-case**: Це ключовий елемент коду:

**match operation:** — ця конструкція виконує порівняння змінної operation із можливими значеннями, як це робиться в умовах if-elif-else. В залежності від того, яке значення ввів користувач, виконується відповідна арифметична операція.

**case '+', case '-', case '\*', case '/'** — кожен блок відповідає певній операції. Якщо, наприклад, користувач вводить "+", викликається функція add(num1, num2).

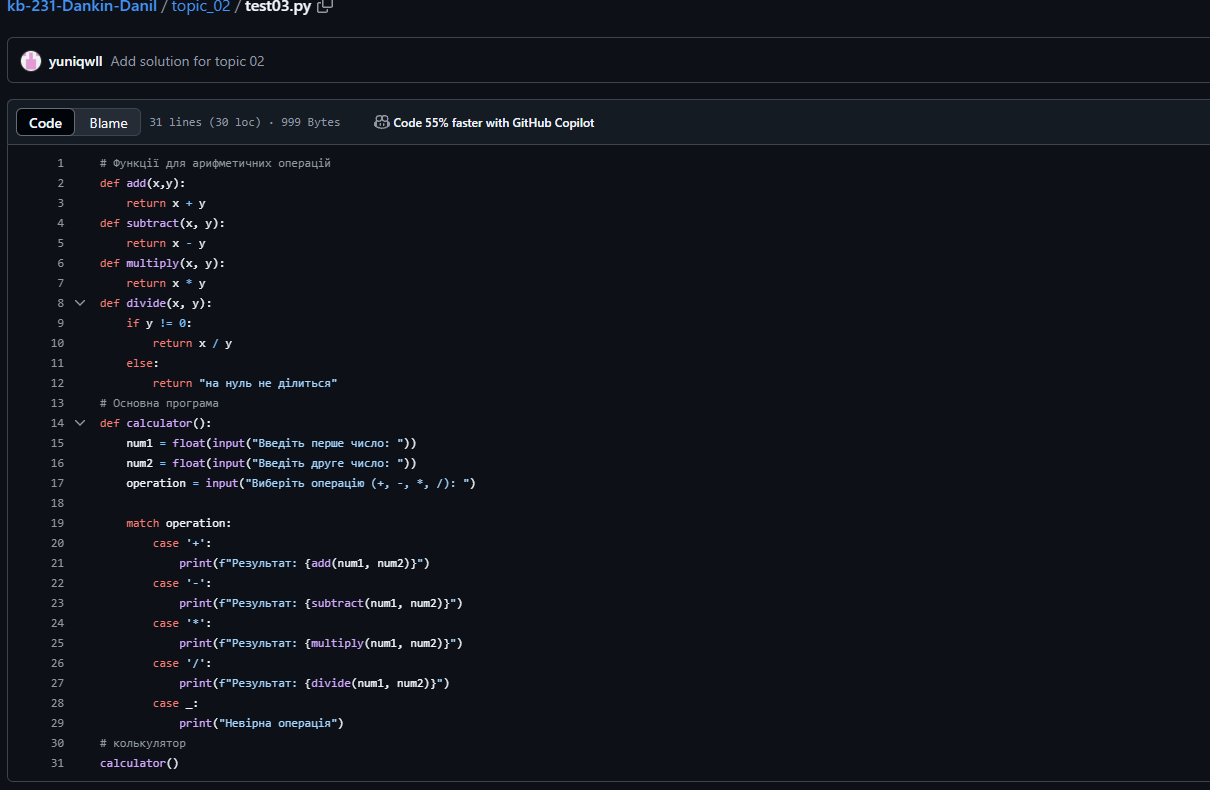
**case \_:** — цей блок використовується як "умовчання", тобто якщо введена операція не збігається з жодною з попередніх, виводиться повідомлення "Невірна операція".

**Запуск калькулятора**: Функція calculator() викликається наприкінці програми:

Текст програми:

|  |
| --- |
| # Функції для арифметичних операцій  def add(x,y):      return x + y  def subtract(x, y):      return x - y  def multiply(x, y):      return x \* y  def divide(x, y):      if y != 0:          return x / y      else:          return "на нуль не ділиться"  # Основна програма  def calculator():      num1 = float(input("Введіть перше число: "))      num2 = float(input("Введіть друге число: "))      operation = input("Виберіть операцію (+, -, \*, /): ")      match operation:          case '+':              print(f"Результат: {add(num1, num2)}")          case '-':              print(f"Результат: {subtract(num1, num2)}")          case '\*':              print(f"Результат: {multiply(num1, num2)}")          case '/':              print(f"Результат: {divide(num1, num2)}")          case \_:              print("Невірна операція")  # колькулятор  calculator() |

Посилання на git hub: <https://github.com/yuniqwll/kb-231-Dankin-Danil/blob/123/topic_02/test03.py>

Скрін: 

Звіт до Теми №3

Цикли

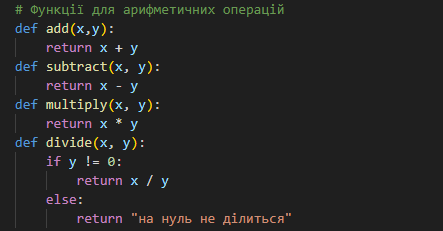
Під час виконання практичного завдання до Теми №3 було надано варіанти рішення до наступних задач:

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

1.

Код містить чотири функції для виконання базових арифметичних операцій:

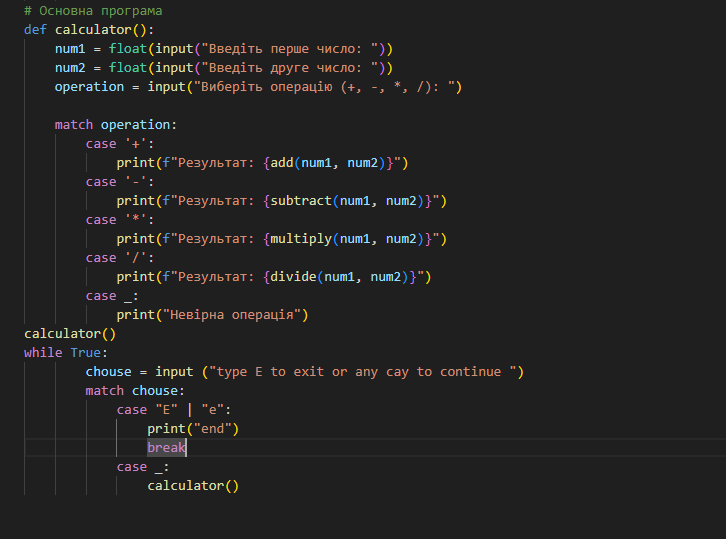
* **add(x, y)**: додає два числа.
* **subtract(x, y)**: віднімає друге число від першого.
* **multiply(x, y)**: перемножує два числа.
* **divide(x, y)**: ділить перше число на друге. Якщо друге число дорівнює нулю, функція повертає повідомлення "на нуль не ділиться".



**Основна програма calculator**

**Кроки виконання:**

* 1. Запит у користувача першого та другого чисел.
  2. Введення бажаної операції (+, -, \*, /).
  3. Виконання відповідної функції залежно від обраної операції за допомогою конструкції match.
  4. Виведення результату операції на екран.

****

2)

Цей код демонструє основні методи роботи зі списками в Python

**1. Список**

Створений список list = [14, 4, 7, 4, 7] містить числові елементи.

Цей список використовується як базовий для тестування різних методів роботи зі списками.

### ****2. Функція**** test\_list\_methods

Ця функція тестує кілька методів списків Python, показуючи їх дію на прикладах.

**Методи, що тестуються:**

1. **extend()**

Додає кілька елементів із іншого списку до кінця існуючого.

Приклад:

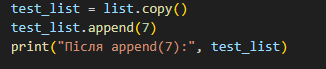


Результат: [14, 4, 7, 4, 7, 9, 2, 6].

2 **append()**

Додає один елемент в кінець списку.

Приклад:

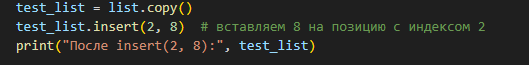


Результат: [14, 4, 7, 4, 7, 7].

3 **insert()**

Вставляє елемент на задану позицію.

Приклад:

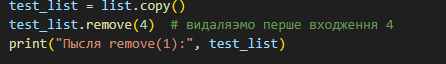


Результат: [14, 4, 8, 7, 4, 7] (8 вставлено на індекс 2).

**4 emove()**

Видаляє перше входження заданого елемента.

Приклад:

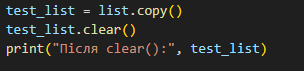


Результат: [14, 7, 4, 7] (перше входження 4 видалено).

**5 clear()**

Очищає список, видаляючи всі елементи.

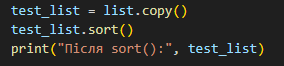
Приклад:



Результат: [] (порожній список).

6 **sort()**

* Сортує список у порядку зростання.
* Приклад:



* Результат: [4, 4, 7, 7, 14].

**7 reverse()**

* Розгортає список (змінює порядок елементів на зворотний).

Приклад:



* Результат: [7, 4, 7, 4, 14].

**8 copy()**

* Створює копію списку.
* Приклад:



* Результат: [14, 4, 7, 4, 7].

**3. Виведення результатів**

Кожен метод демонструє результат у консолі за допомогою функції print():

* Спочатку виводиться вихідний список.
* Потім показується список після застосування кожного методу.

### ****4. Виклик функції****

Функція test\_list\_methods запускається наприкінці, що забезпечує тестування всіх методів списків.

Звіт до Теми №4

Виняткові ситуації

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

Завдання для самостійного виконання

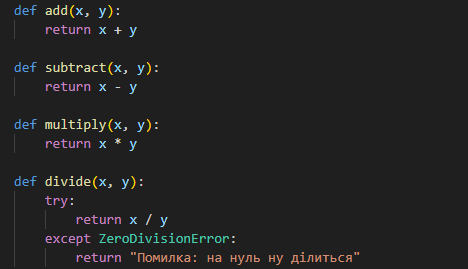
**Попередні умови**: реалізована програма калькулятор, що використовує метод нескінченного введення даних для обробки. Всі дії (додавання, віднімання, множення, ділення) реалізовані як окремі функції та використовуються у відповідних місцях.

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

Цей код реалізує калькулятор, який обробляє арифметичні операції та враховує можливі помилки під час введення даних і виконання операцій

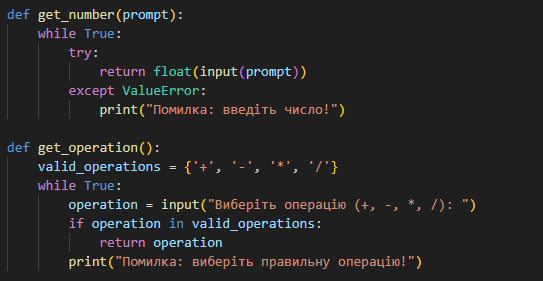
**Арифметичні функції**

1. **add(x, y)**
   * Повертає суму чисел x і y.
2. **subtract(x, y)**
   * Повертає різницю чисел x і y.
3. **multiply(x, y)**
   * Повертає добуток чисел x і y.
4. **divide(x, y)**
   * Виконує ділення числа x на y.
   * Якщо y = 0, обробляє виключення ZeroDivisionError і повертає повідомлення: "Помилка: на нуль не ділиться".



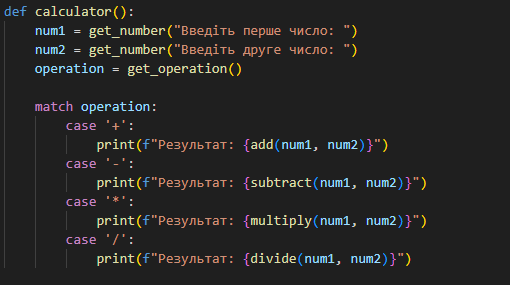
**Функції для введення даних**

1. **get\_number(prompt)**
   * Запитує у користувача число з текстом, переданим через prompt.
   * Якщо користувач вводить нечислове значення, обробляє виключення ValueError і вимагає повторного введення.
2. **get\_operation()**
   * Запитує у користувача операцію (одну з: +, -, \*, /).
   * Якщо операція недопустима, виводить повідомлення про помилку і повторно запитує.



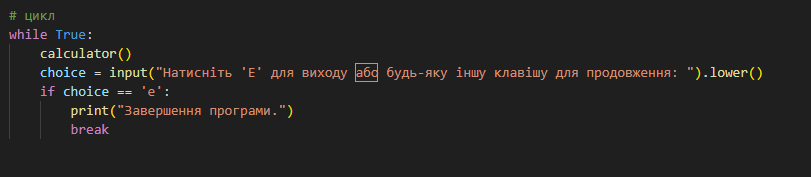
**Головна функція калькулятора**

1. **calculator()**
   * Викликає функції:
     + get\_number() для отримання двох чисел.
     + get\_operation() для отримання операції.
   * Виконує вибрану арифметичну операцію, використовуючи match-case (або if-elif для версій Python <3.10).
   * Виводить результат.



**Цикл виконання**

1. **Основний цикл while True**
   * Викликає функцію calculator() для виконання розрахунку.
   * Після кожної операції запитує, чи хоче користувач продовжити:
     + Якщо введено 'e' або 'E', програма завершується.
     + Інакше цикл повторюється.



Звіт до Теми №5

Бібліотеки

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

1. Гра з комп’ютером: камінь, ножиці, папір. Програма виконує запит від користувача на введення одного із значень ["stone", "scissor", "paper"]. Наступним кроком, використовуючи модуль random, програма у випадковому порядку вибирає одне із значень ["stone", "scissor", "paper"]. В залежності від умови, що камінь перемагає ножиці, ножиці перемагають папір, а папір перемагає камінь визначити переможця.
2. 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.
3. 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.

1. Цей код реалізує гру "Камінь-Ножиці-Папір" між користувачем та комп'ютером

1)**Імпорт модуля**

* **random** — стандартний модуль Python, який використовується для роботи з випадковими числами або вибором випадкових елементів зі списків.
* У цьому коді метод random.choice() використовується для вибору випадкового ходу комп'ютера.



### 2) ****Функція**** play\_rps

Ця функція реалізує гру. Її структура:

**a) Варіанти вибору**



Створюється список можливих ходів: "stone" (камінь), "scissor" (ножиці), "paper" (папір).

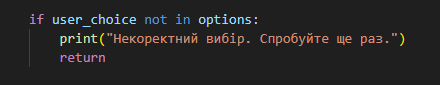
**b) Вибір користувача**



Користувач вводить свій вибір (одне зі слів: stone, scissor, paper).

**.lower()** переводить введений текст у нижній регістр, щоб уникнути помилок через регістр літер (наприклад, Stone також буде прийнято).

**c) Перевірка коректності вводу**



Якщо вибір користувача не входить у список options, програма виводить повідомлення про помилку і завершує функцію.

**d) Вибір комп'ютера**



Комп'ютер випадковим чином обирає свій хід зі списку options.

Програма повідомляє користувача, який хід зробив комп'ютер.

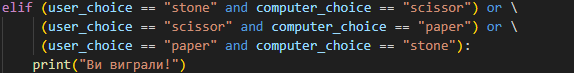
### ****Логіка визначення переможця****

### ****a) Нічия****



* Якщо вибір користувача та комп'ютера збігаються, оголошується нічия.

**b) Виграш користувача**

* Перевіряється, чи виграє користувач згідно з правилами:
  + Камінь (stone) перемагає ножиці (scissor).
  + Ножиці (scissor) перемагають папір (paper).
  + Папір (paper) перемагає камінь (stone)

**c) Виграш комп'ютера**

* 
* У всіх інших випадках оголошується перемога комп'ютера.

2). Код складається з двох основних функцій і є прикладом простої програми для конвертації валют на основі даних Національного банку України (НБУ)

1. **Імпорт бібліотеки** requests

Ця бібліотека використовується для виконання HTTP-запитів, що дозволяє програмі звертатися до API НБУ для отримання курсів валют.



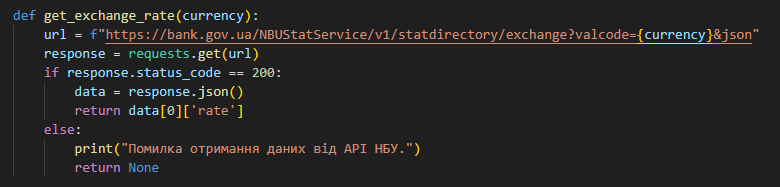
2.**Функція get\_exchange\_rate(currency)**

Ця функція приймає на вхід код валюти (наприклад, "USD", "EUR", "PLN") і виконує такі дії:

1. Формує URL-адресу API НБУ для отримання поточного курсу валюти.
2. Надсилає запит до API за допомогою requests.get(url).
3. Перевіряє, чи запит був успішним (код відповіді 200).

Якщо так, обробляє JSON-відповідь і повертає курс валюти (data[0]['rate']).

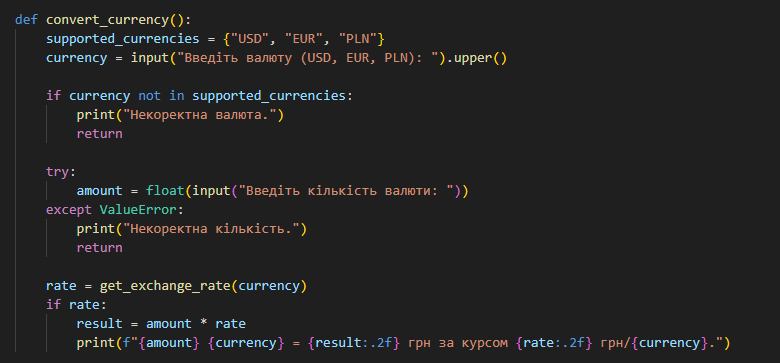
Якщо запит неуспішний, виводить повідомлення про помилку і повертає None



**3. Функція convert\_currency()**

Ця функція відповідає за взаємодію з користувачем і виконує такі завдання:

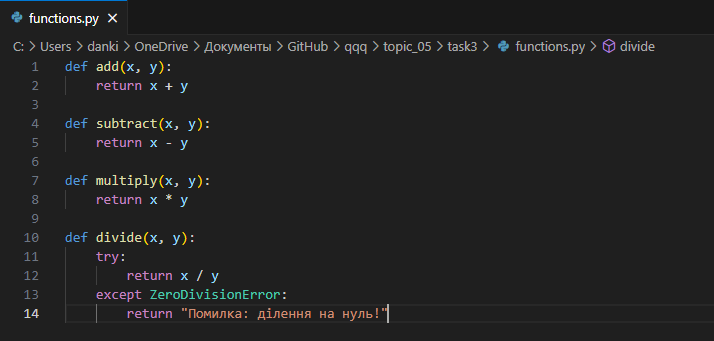
1. **Список підтримуваних валют**: Створює набір підтримуваних валют: {"USD", "EUR", "PLN"}.
2. **Запит валюти**: Запитує у користувача, яку валюту потрібно конвертувати, і переводить введення у верхній регістр (upper()).
3. **Перевірка валюти**: Якщо введена валюта не підтримується, виводиться повідомлення про помилку і функція завершується.
4. **Запит кількості валюти**: Просить у користувача ввести кількість валюти. Якщо введення некоректне (наприклад, текст замість числа), виводить помилку і завершує роботу.
5. **Отримання курсу**: Викликає функцію get\_exchange\_rate() для отримання курсу валюти.
6. **Обчислення та виведення результату**: Якщо курс валюти отримано успішно, обчислює суму в гривнях (amount \* rate) і виводить результат.

4**. Виклик програми**

У кінці коду функція convert\_currency() викликається одразу після її визначення, щоб програма почала працювати після запуску.

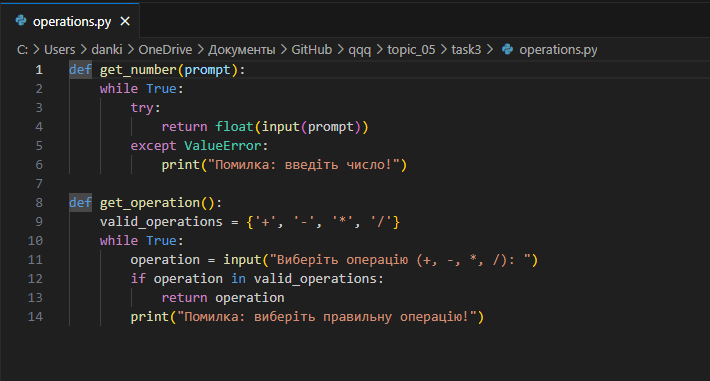
  
  
  
  
3) Ці три файли разом утворюють простий калькулятор, реалізований у модульній формі. Кожен файл виконує свою роль у структурі програми:  
  
1. functions.py: Функції для арифметичних операцій

Цей модуль містить функції для виконання основних математичних операцій:

* **add(x, y)**: Додає два числа.
* **subtract(x, y)**: Віднімає друге число від першого.
* **multiply(x, y)**: Множить два числа.
* **divide(x, y)**: Ділить перше число на друге. Якщо друге число дорівнює нулю, обробляє помилку і повертає відповідне повідомлення.  
    
  2. operations.py: Функції для отримання даних від користувача

Цей модуль відповідає за взаємодію з користувачем і обробку його введення:

print("Помилка: введіть число!")

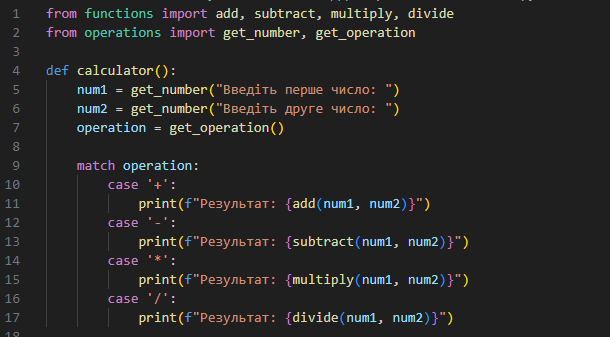
* **get\_number(prompt)**: Запитує у користувача число.
  + Повертає введене число, якщо воно коректне.
  + Якщо користувач вводить некоректні дані (наприклад, текст), повторює запит.
* **get\_operation()**: Запитує у користувача одну з підтримуваних операцій (+, -, \*, /).
  + Повторює запит, доки користувач не введе коректну операцію.  
      
      
    

### 3. calc.py: Основна логіка калькулятора

Цей файл об’єднує всі модулі і реалізує основну логіку калькулятора.

* Імпортує функції для обчислень (add, subtract, multiply, divide) і для взаємодії з користувачем (get\_number, get\_operation).

Основна функція:

1. **num1 і num2**: Отримує перше і друге число від користувача через get\_number.
2. **operation**: Отримує вибрану операцію від користувача через get\_operation.
3. **match-case**: Вибирає відповідну функцію залежно від операції і виконує обчислення.  
     
     
   Цикл для повторення:

* Програма повторює роботу калькулятора, доки користувач не введе 'E' для виходу.

### ****Як це працює разом:****

1. Користувач запускає файл calc.py.
2. Програма запитує два числа і математичну операцію.
3. Викликається відповідна функція з functions.py для виконання обчислення.
4. Результат виводиться на екран.
5. Користувач може вибрати, чи продовжувати роботу, чи завершити програму.

Звіт до Теми №6

Робота з файлами

Під час виконання практичного завдання до Теми №6 було надано варіанти рішення до наступних задач:

Завдання для самостійного виконання

Попередні умови: Реалізована програма калькулятор в файлі calc.py, до якого підключають файл functions.py та operations.py.

1)Розробити механізм логування всіх дій, що виконує програма. Забезпечити зберігання інформації про введені данні, виконану операцію та результат виконання операції над даними.

2)Маючи не відсортований 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.

1. **У новій версії коду з'явилося логування, яке додає механізм запису дій калькулятора до лог-файлу. Це дозволяє зберігати історію виконаних операцій, включно з введеними числами, вибраною операцією та отриманим результатом. Нижче описано всі нововведення і їхнє призначення.  
1)Імпорт модуля** logging

* **logging**: Вбудований модуль Python для ведення логів.

### Логи можна записувати у файл, консоль або інші місця, що дозволяє аналізувати виконання програми навіть після її завершення. 2) ****Налаштування логування****

* **filename="calculator.log"**: Указує, що логи будуть записуватися у файл з назвою calculator.log.
* **level=logging.INFO**: Указує, що будуть записані повідомлення рівня INFO і вище.
  + Рівень INFO використовується для загальної інформації про події.
* **format="%(asctime)s - %(message)s"**: Указує формат запису:
  + **%(asctime)s**: Дата і час запису логів.
  + **%(message)s**: Основне повідомлення.

### 3) ****Логування в основній функції**** calculator()

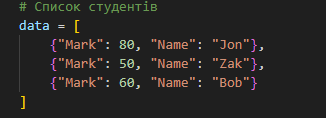
У функцію calculator додано запис до логів після виконання операції:

* **Що записується:**
  + Операція (наприклад, +, -, \*, /).
  + Введені числа num1 і num2.
  + Результат обчислення result.
* **Призначення:**
  + Збереження інформації про всі виконані операції.
  + Допомога у відстеженні роботи програми (наприклад, для налагодження або аналізу помилок).

### ****Як це працює:****

1. Коли користувач вводить числа і операцію, програма виконує обчислення, як і раніше.
2. Після виконання операції результат виводиться на екран.
3. Інформація про виконану операцію додається у лог-файл.

2)**Цей код демонструє, як сортувати список словників за різними критеріями за допомогою функції sorted() і лямбда-функції. Нижче пояснення кожного етапу.  
  
1.Вихідні дані**

]

* Це список, що містить словники. Кожен словник представляє інформацію про студента:
  + "Mark" — оцінка студента.
  + "Name" — ім'я студента.

### 2. ****Сортування за ім'ям****

* **sorted()**:
  + Функція для сортування колекцій (списків, кортежів тощо).
  + Параметр key визначає, за яким критерієм виконувати сортування.
* **key=lambda x: x["Name"]**:
  + Лямбда-функція використовується для отримання значення, за яким виконується сортування. У цьому випадку це значення ключа "Name".
* **Результат сортування**: Список сортується за алфавітом імен:

[

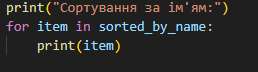
{"Mark": 60, "Name": "Bob"}, # Bob (перший за алфавітом)

{"Mark": 80, "Name": "Jon"}, # Jon

{"Mark": 50, "Name": "Zak"} # Zak (останній за алфавітом)

]

* **Виведення**:



### ****3)Сортування за оцінкою****

* **key=lambda x: x["Mark"]**:
  + Лямбда-функція повертає значення ключа "Mark" — оцінки студента. Сортування виконується за оцінками у порядку зростання.
* **Результат сортування**: Список сортується за оцінками:

[

{"Mark": 50, "Name": "Zak"}, # Zak (оцінка 50, найменша)

{"Mark": 60, "Name": "Bob"}, # Bob (оцінка 60)

{"Mark": 80, "Name": "Jon"} # Jon (оцінка 80, найбільша)

]

* **Виведення**:

### ****Пояснення роботи**** sorted()

* Функція sorted() створює новий відсортований список, не змінюючи початковий  
    
    
   Звіт до Теми №7

Об'єктно-орієнтоване програмування

Під час виконання практичного завдання до Теми №7 було надано варіанти рішення до наступних задач:

Завдання для самостійного виконання

1. Розробити клас **Student** атрибутами якого э два параметра **name** та **age**. Створити список елементами якого є об'єкти класу **Student**. Написати цикл який виводить на екран елементи списку у відсортованому порядку. Для сортування використати стандартну функцію **sorted**. Функція **sorted** має використовувати **lambda** функцію для визначення ключа сортування.
2. Використовуючи принципи ООП переписати програму Калькулятор. Завдання має бути виконано використовуючи модульний підхід

1. Цей код демонструє, як сортувати список об'єктів класу Student за допомогою функції sorted() і лямбда-функції. Нижче пояснення кожного компонента коду.

### 1) ****Клас**** Student

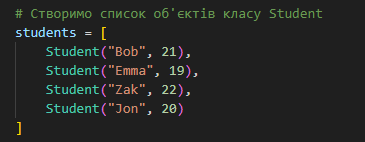
### 

* Конструктор класу Student, який ініціалізує об'єкт з двома атрибутами:
  + **name**: Ім'я студента.
  + **age**: Вік студента.

**Метод** \_\_init\_\_

* Конструктор класу Student, який ініціалізує об'єкт з двома атрибутами:
  + **name**: Ім'я студента.
  + **age**: Вік студента.

### ****2)Створення списку студентів****



* Це список об'єктів класу Student, кожен з яких має ім'я (name) і вік (age).

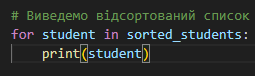
### 3) ****Сортування за віком (****age****)****



**sorted()**

* Створює новий відсортований список на основі переданого критерію сортування (key).
* У цьому випадку ключ сортування — це значення атрибута age кожного об'єкта Student.  
    
  key=lambda student: student.age
* Лямбда-функція повертає значення age кожного студента, яке використовується для сортування.
* Наприклад:
  + Для об'єкта Student("Bob", 21) лямбда-функція поверне 21.

### ****4)Виведення відсортованого списку****



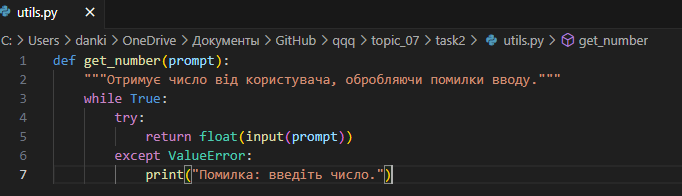
Цикл for проходить по відсортованому списку sorted\_students і друкує кожен об'єкт.

Метод \_\_repr\_\_ визначає вигляд кожного об'єкта при друку.

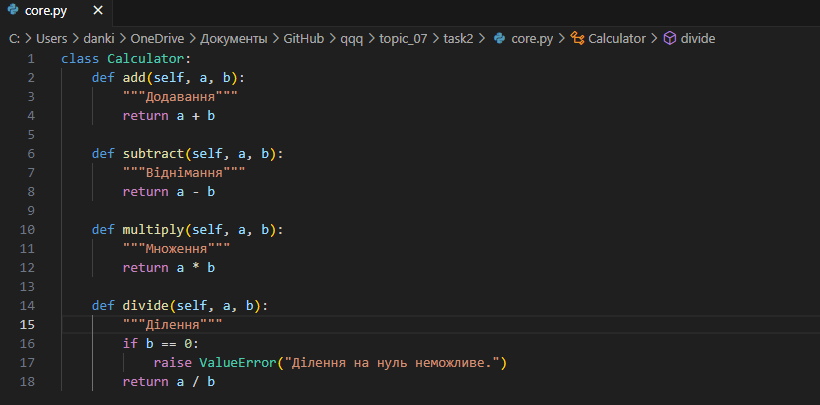
### ****Як працює код у цілому****

1. Створюється клас Student з полями name і age.
2. Ініціалізується список об'єктів students.
3. Список сортується за віком за допомогою функції sorted() і лямбда-функції.
4. Відсортований список виводиться на екран.

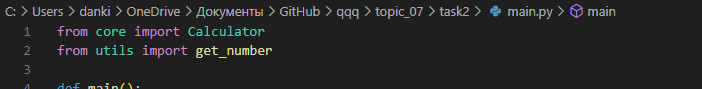
### 2. Цей код реалізує простий консольний калькулятор у модульному вигляді. Кожен файл виконує окрему функцію, що робить програму структурованою, зручною для підтримки та розширення. Нижче пояснення всіх трьох файлів. 1) utils.py****: Утилітарний модуль****

* **Функція get\_number(prompt)**:
  + Використовується для отримання числового вводу від користувача.
  + **Цикл while True**: Повторює запит, доки користувач не введе коректне число.
  + **Обробка помилок ValueError**: Якщо користувач вводить текст або некоректне значення, виводиться повідомлення і запит повторюється.
* **Призначення**:
  + Зробити процес введення чисел надійним і зручним для користувача.

### 2)core.py****: Клас**** Calculator

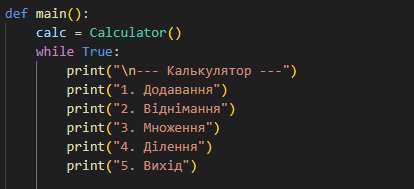
* **Клас Calculator**:
  + Інкапсулює основні математичні операції: додавання, віднімання, множення і ділення.
* **Методи класу**:
  + add(a, b): Повертає суму двох чисел.
  + subtract(a, b): Повертає різницю між двома числами.
  + multiply(a, b): Повертає добуток двох чисел.
  + divide(a, b): Повертає результат ділення. Якщо друге число b дорівнює нулю, піднімає помилку ValueError.
* **Призначення**:
  + Централізація математичної логіки.
  + Полегшує повторне використання цих операцій у будь-яких інших проектах.

### ****3.**** main.py****: Головний файл****



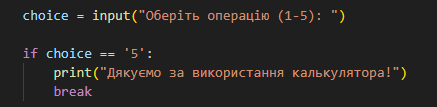
* Імпортує клас Calculator з модуля core.py і функцію get\_number з модуля utils.py.

**Функція** main()



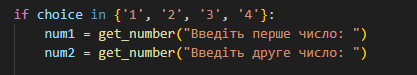
**Меню**:

* + Виводить список доступних операцій.
  + Пропонує користувачу вибрати операцію або вийти з програми.



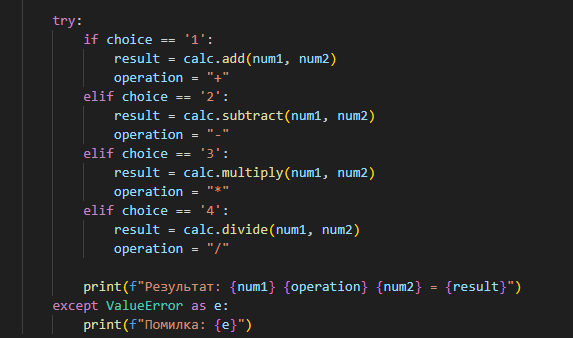
**Вихід**:

* + Якщо користувач вибрав 5, програма завершує роботу.



**Обробка вибору**:

* + Якщо користувач вибрав операцію (1-4), запитує два числа, використовуючи функцію get\_number().

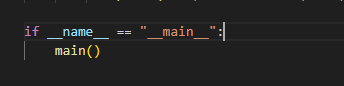
 **Виконання операції**:

* + Викликає відповідний метод класу Calculator, залежно від вибору користувача.
  + Якщо користувач вибрав ділення на нуль, обробляє помилку ValueError і виводить повідомлення.

 **Неправильний вибір**:

* + Якщо введено некоректний номер операції, програма повідомляє про помилку.

**Виконання програмиЖ**

*   
  Запускає функцію main() лише тоді, коли файл запускається безпосередньо (а не імпортується як модуль).

### ****Як працює програма****

1. Користувач вибирає операцію з меню.
2. Вводить два числа, які обробляються функцією get\_number().
3. Програма виконує вибрану операцію, використовуючи методи класу Calculator.
4. Результат виводиться на екран.
5. Користувач може вибрати нову операцію або вийти з програми.

https://github.com/yuniqwll/kb-231-Dankin-Danil