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:

Перетворення рядка

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

Хід виконання завдання:

Я вирішив трохи ускладнити собі задачу, і зробив можливість вводу будь якого тексту, але з умовою, що цей текст буде мати 10 символів, якщо ні, то спрацюють обробники помилки і скажуть більше треба, чи менше, якщо ж текст має 10 символів, то за допомогою слайсів, де можна вибрати з якого елементу масива почати, де зупинитись і який крок, в моєму випадку я зазначив лише який крок -1, щоб він йшов з кінця.

Текст програми:

```
text = input("Введіть текст з 10 знаками: ")

if len(text) > 10:

print("Знаків більше ніж 10")

elif len(text) < 10:

print("Знаків менше ніж 10")

else:
```

reserved = text[::-1]

print(reserved)

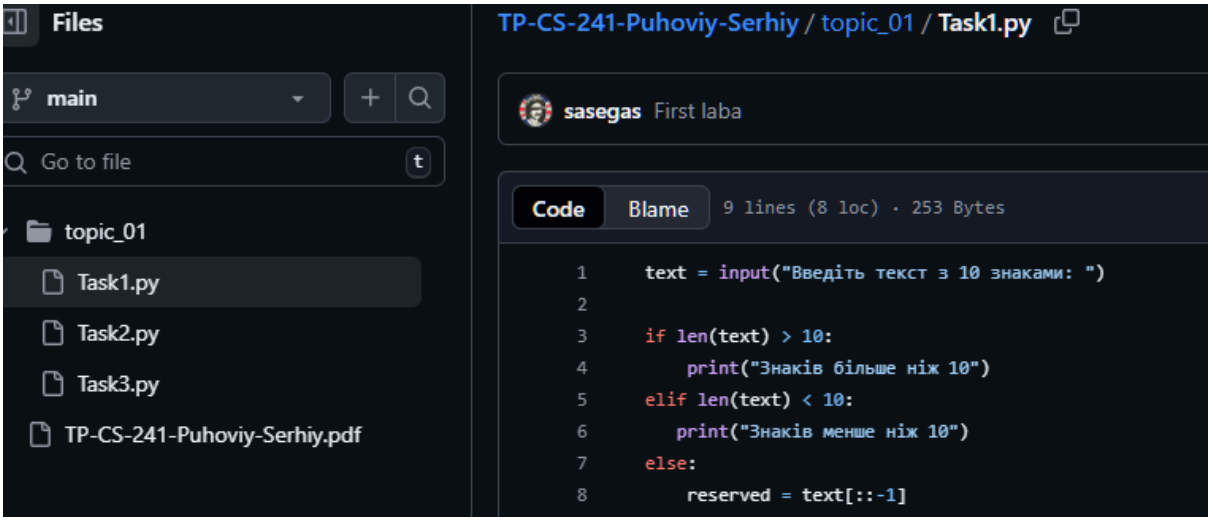

Рис.1 Скріншот першого завдання з GitHub

Посилання:https://github.com/sasegas/TP-CS-241-Puhoviy-Serhiy/blob/main/topic_01/Task1.py

Стилі для тексту

Виконати деякі тести на strip, capitalize, title, upper, lower

Хід виконання завдання:

В цьому завданні я написав текст в змінній string, щоб його можна було перевірити на всі вищезазначені функції,

strip - прибирає зайві пробіли на початку і в кінці,

capitalize - в рядку тільки перша літера, першого слова велика,

title - в кожному слові великі літери, тільки перші, а інші в нижньому регістр,

upper - всі букви у верхньому регістрі,
lower - всі букви у нижньому регістрі

Текст програми:

string = ' hello pytHon! '
print(string)

```
print('strip:',string.strip())
print('capitalize:',string.strip().capitalize())
print('title:',string.strip().title())
print('upper:',string.strip().upper())
print('lower:',string.strip().lower())
```

Рис.2 Скріншот другого завдання з GitHub

Посилання:https://github.com/sasegas/TP-CS-241-Puhoviy-Serhiy/blob/main/topic_01/Task2.py

Квадратне рівняння

Знайти відповідь для квадратного рівняння за допомогою функцій

Хід виконання завдання:

Спочатку я створив функцію quadraticFn в якій задав 3 змінні, a,b,c відповідно як у рівнянні і після цього задав змінну дискримінанта, де його обрахував відповідно до формули і після цього використав умовний оператор if, для того щоб правильно порахувати x, після чого ми маємо три варіанти відповіді, або x не існує, або є тільки один x, або є x1 і x2. Для того щоб порахувати квадратний корінь дискримінанту мені знадобилась бібліотека math.

Текст програми:

```
import math


def quadraticFn():
   a = int(input('Введіть перший коефіцієнт:'))
   b = int(input('Введіть другий коефіцієнт:'))
   c = int(input('Введіть вільний член:'))
   discriminator = b**2 - 4*a*c
   if discriminator == 0:
      print ("Рівняння має один корінь")
      x = -b/(2*a)
      print(x)
   elif discriminator > 0:
      print ("Рівняння має два корінь")
      x1 = (-b - math.sqrt(discriminator))/(2*a)
      x2 = (-b + math.sqrt(discriminator))/(2*a)
      print('x1=',x1)
      print('x2=',x2)
   else:
      print("Рівняння не має дійсних коренів")


quadraticFn()
```

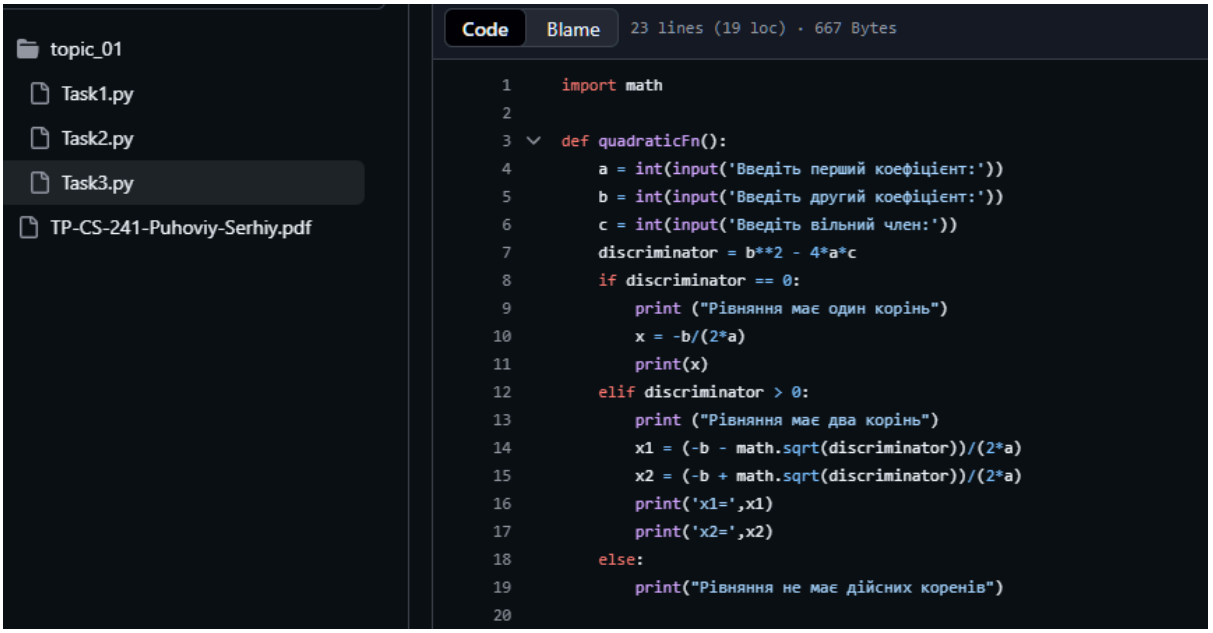

Рис.3 Скріншот третього завдання з GitHub

Посилання:https://github.com/sasegas/TP-CS-241-Puhoviy-Serhiy/blob/main/topic_01/Task2.py

Звіт до Теми №2
Умовний перехід

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

Функція пошуку коренів

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

Хід виконання завдання:

Використав код для знаходження дискримінанта з попередньої теми, який я додав, після нього я написав функцію, для розрахунку дискримінанта, за допомогою умовних операторів if, elif, else, я додав три умови по яким і знаходяться корені.

Текст програми:

```
import math


def discriminant(a,b,c):
   return b**2 - 4*a*c




def quadratic_roots(a, b, c):
    d = discriminant(a, b, c)


    if d > 0:
        print("Рівняння має два корені")
        x1 = (-b - math.sqrt(d)) / (2*a)
        x2 = (-b + math.sqrt(d)) / (2*a)
        print("x1 =", x1)
```

```
            print("x2 =", x2)
            return x1, x2
        elif d == 0:
            print("Рівняння має один корінь")
            x = -b / (2*a)
            print("x =", x)
            return x,
        else:
            print("Рівняння не має дійсних коренів")
            return None



a, b, c = 3, -18, 27
roots = quadratic_roots(a, b, c)
```

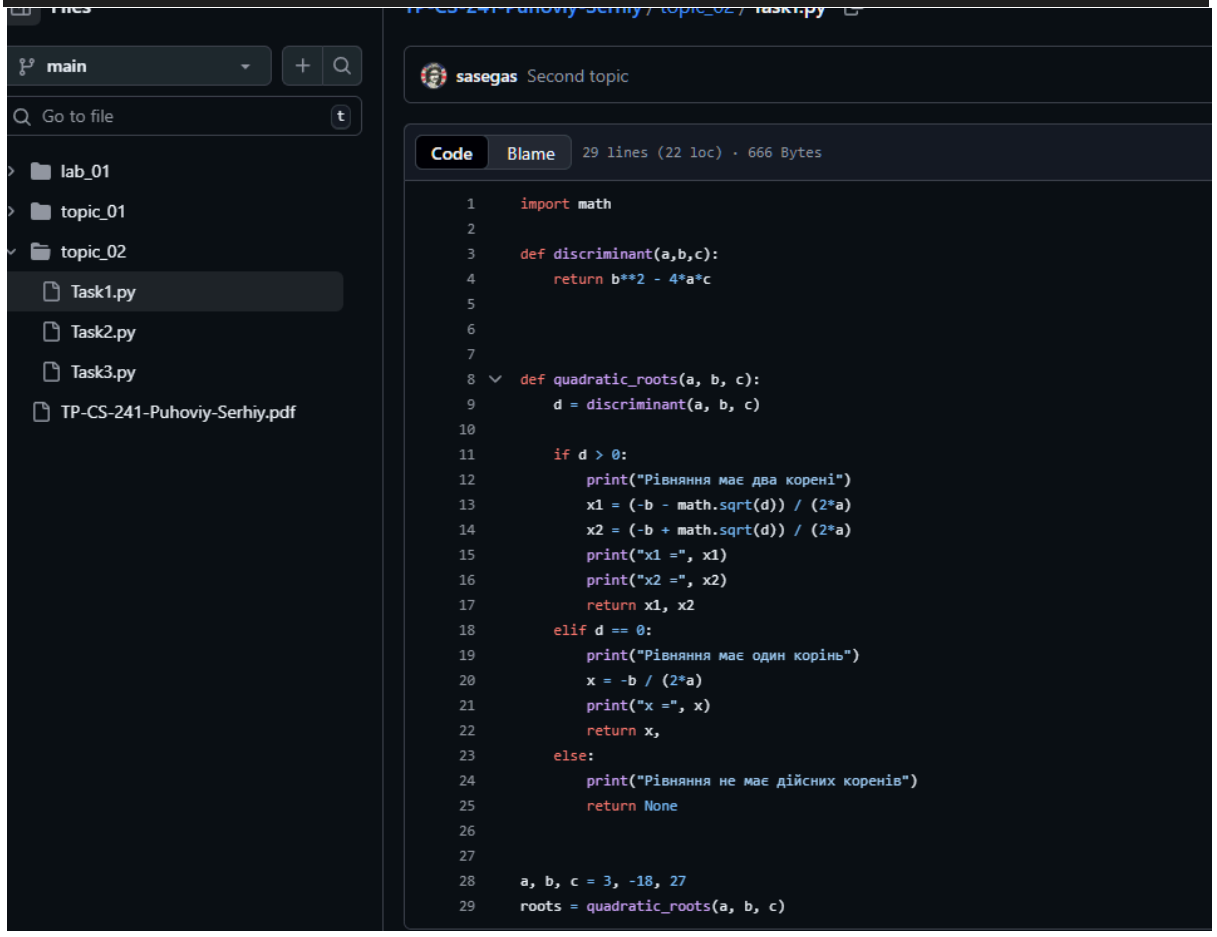

Рис.1 Скріншот першого завдання з GitHub

Калькулятор використовуючи if

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

Хід виконання завдання:

В цьому завданні я спочатку додав input для кожного значення, а саме першого числа, другого і знаку, потім створив функцію калькулятор, яки за допомогою умовних операторів робить певну дію, чи то множення, ділення, чи інші, а потім значення цієї функції поміщається в змінну result, яку, потім показую за допомогою print

Текст програми:

```
a = float(input("введіть перше число: "))
b = float(input("введіть перше друге число: "))
sign = input("введіть дію(+ - * /): ")


def calculator (a, b, sign):
   if sign == "+":
      return  a + b
   elif sign == "-":
      return a - b
   elif sign == "*":
      return  a * b
   elif sign == "/":
      if b != 0:
         return  a / b
      else:
         return "Ділити на 0 неможна"
   else:
      return "Була задана некоректна дія"
```

```
result = calculator(a,b,sign)

print("Відповідь: ", result)
```

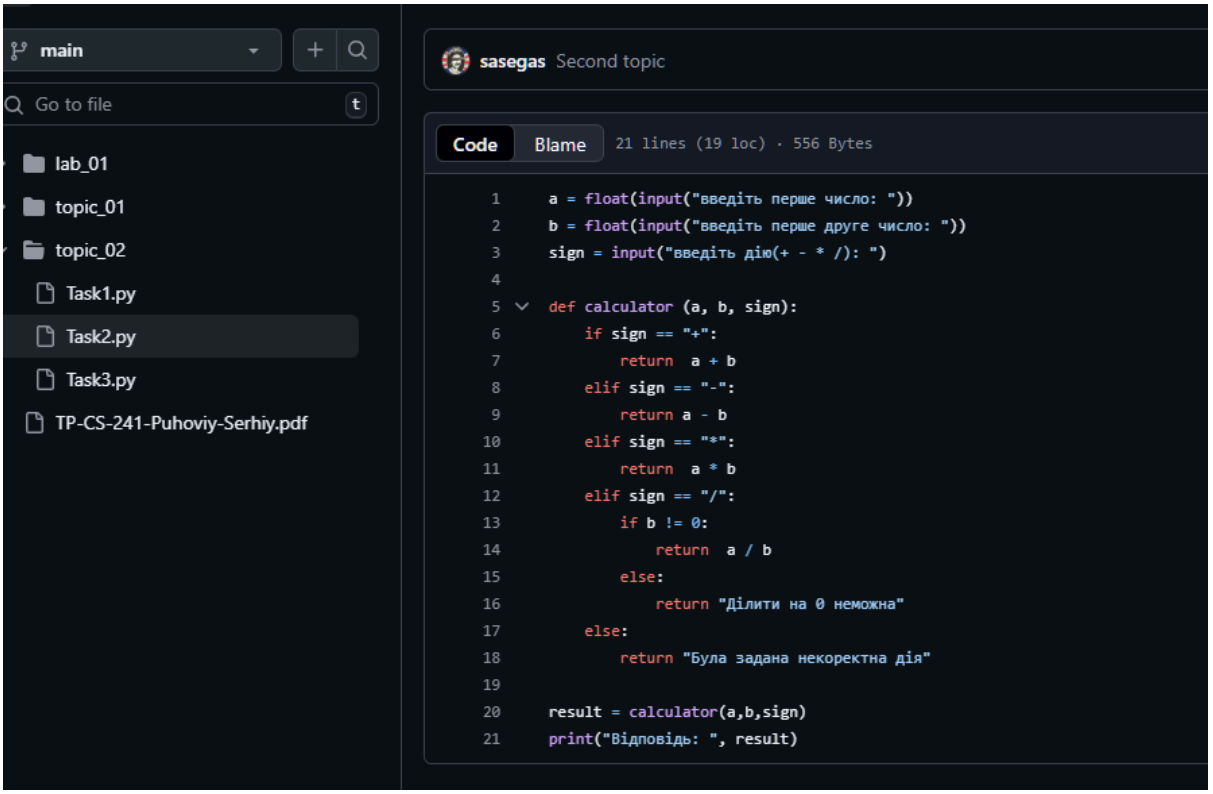

Рис.2 Скріншот другого завдання з GitHub

Калькулятор використовуючи match

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

Хід виконання завдання:

В цьому завданні я зробив 4 окремі функції, які відповідають за конкретну дію і за допомогою match запускав певну дію в залежності від вибору користувача, а потім виводив

Текст програми:

```
def plus(a,b):
   return a+b
def minus(a,b):
   return a-b
def multiply(a,b):
```

```
    return a*b
def divide(a,b):
    if b!= 0:
        return a/b
    else:
        return "Ділити на 0 неможна"

a = float(input("введіть перше число: "))
b = float(input("введіть перше друге число: "))
sign = input("введіть дію(+ - * /): ")



match(sign):
    case "+":
        result = plus(a,b)

    case "-":
        result = minus(a,b)

    case "*":
        result = multiply(a,b)

    case "/":
        result = divide(a,b)
    case _:
        result = "Була задана некоректна дія"

print("Відповідь: ", result)
```

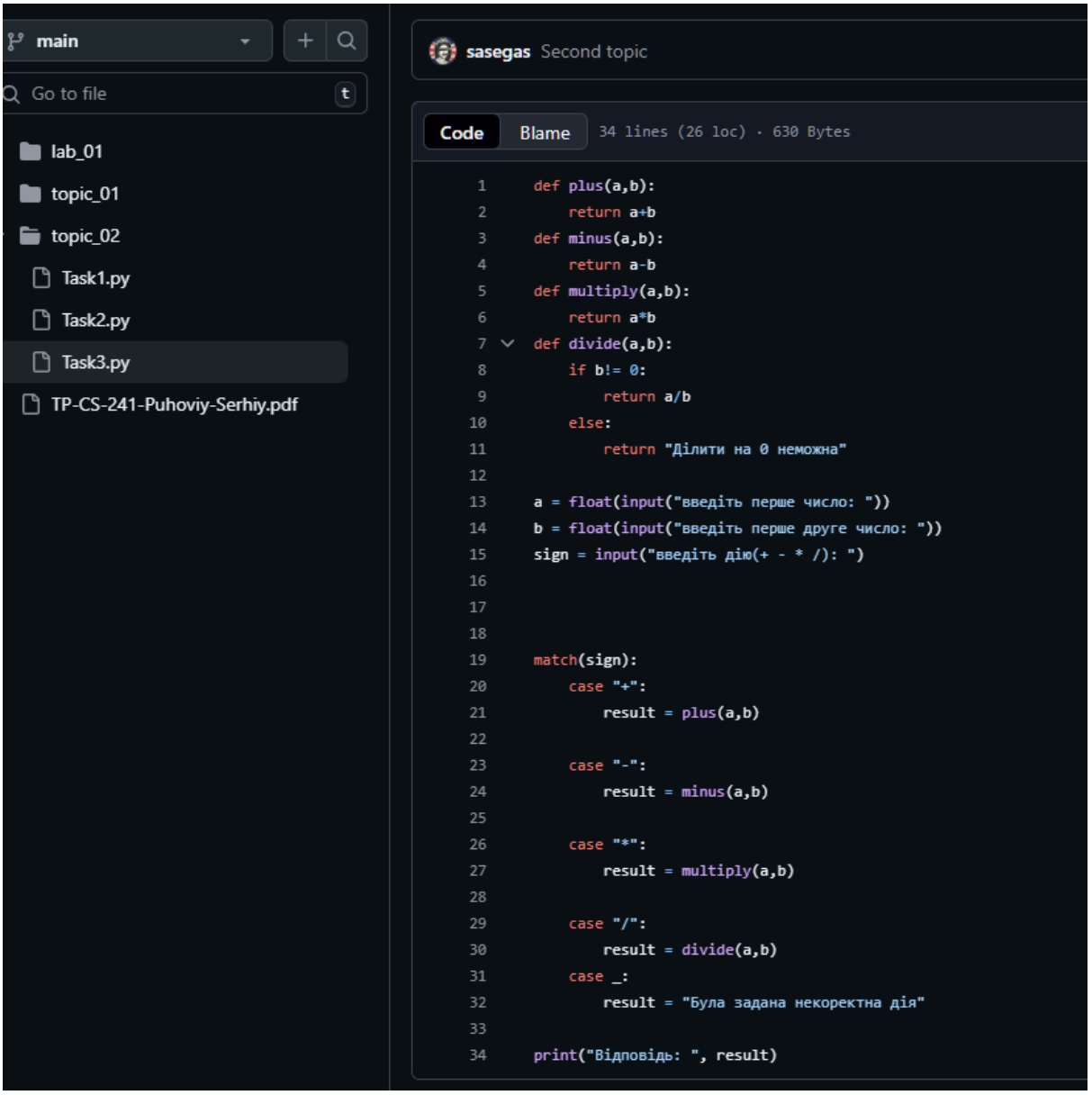

Рис.3 Скріншот третього завдання з GitHub

Звіт до Теми №3

Цикли

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

Програма калькулятор з використанням циклів

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:

В цьому завданні я використав минулий код з калькулятором, тільки додав ще цикл while який дає змогу працювати калькулятору, поки його не вимкнуть.

Текст програми:

```
def plus(a, b):
    return a + b


def minus(a, b):
    return a - b


def multiply(a, b):
    return a * b


def divide(a, b):
    if b != 0:
        return a / b
    else:
        return "Ділити на 0 не можна"


print("Щоб вийти, введіть 'exit' замість знака дії.")
```

```python
while True:
    try:
        a = float(input("Введіть перше число: "))
        b = float(input("Введіть друге число: "))
    except ValueError:
        print("Потрібно вводити числа!")
        continue

    sign = input("Введіть дію (+ - * /) або 'exit' для виходу: ")

    if sign.lower() == "exit":
        print("Роботу завершено.")
        break

    match sign:
        case "+":
            result = plus(a, b)
        case "-":
            result = minus(a, b)
        case "*":
            result = multiply(a, b)
        case "/":
            result = divide(a, b)
        case _:
            result = "Була задана некоректна дія"

    print("Відповідь:", result, "\n")
```

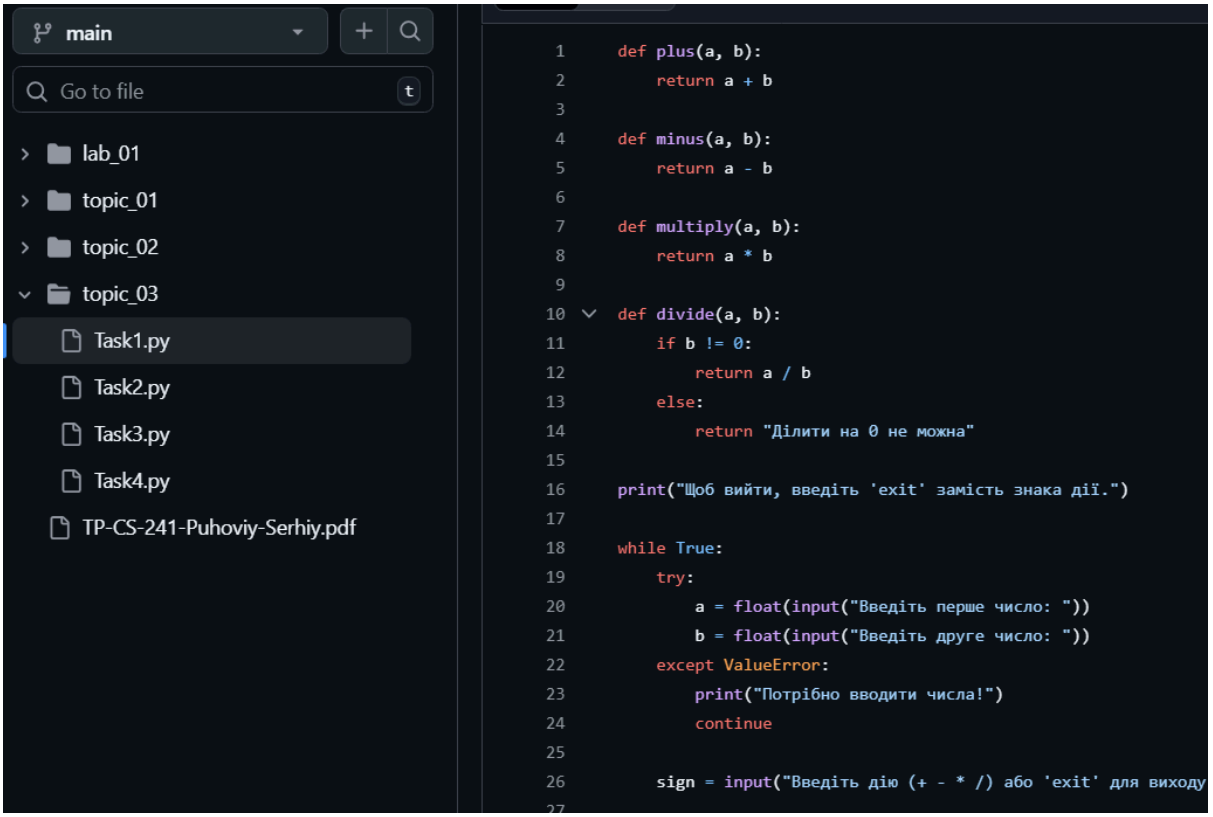

Рис.1 Скріншот першого завдання з GitHub

Програма тестування функцій списків

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

Хід виконання завдання:

У процесі виконання завдання я ознайомився з основними функціями роботи зі списками в Python. Зокрема, використав метод append() для додавання одного елемента в кінець списку, extend() – для об’єднання списків, insert() – для вставки елемента за вказаним індексом, remove() – для видалення першого входження заданого елемента, clear() – для повного очищення списку, sort() – для сортування елементів у списку, reverse() – для зміни порядку елементів на зворотній, а також copy() – для створення копії списку.

Текст програми:

```
print("=== Тестування методів списків ===")

list = [1, 5, 7]

print('Початковий список: ', list)
```

```
list.extend([4, 2])

print('extend([4, 2]):    ', list)

list.append(3)

print('append(3):         ', list)

list.insert(1,1)

print('insert(1,1):       ', list)

list.remove(1)

print('remove(1):         ', list)

list.sort()

print('sort():            ', list)

list.reverse()

print('reverse():         ', list)

list_copy = list.copy()

print("copy():            ", list_copy)

list.clear()

print('clear():           ', list)
```

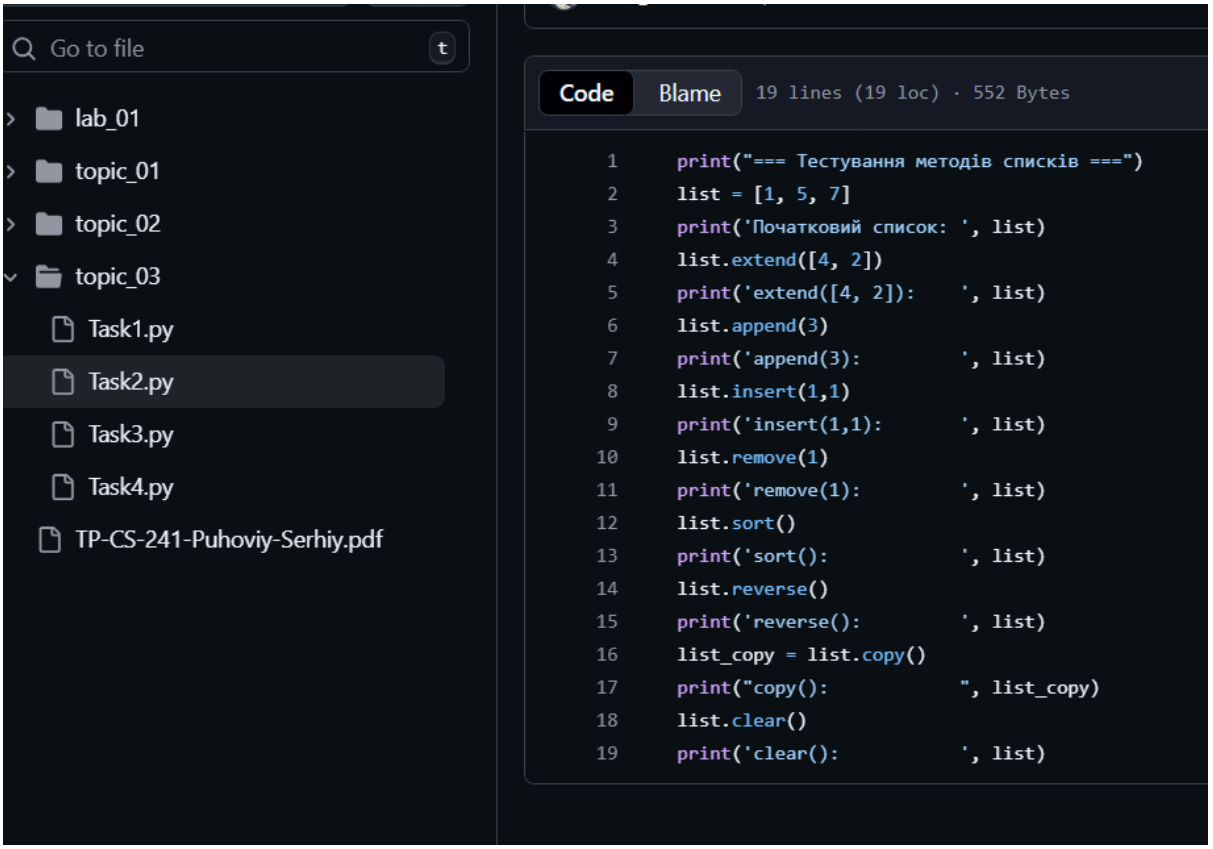

Рис.2 Скріншот другого завдання з GitHub

Програма тестування функцій словника

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

Хід виконання завдання:

У процесі виконання завдання я ознайомився з основними функціями роботи зі словниками в Python. Зокрема, використав метод update() для додавання або оновлення пар ключ-значення, оператор del – для видалення ключів зі словника, clear() – для повного очищення словника, keys() – для отримання всіх ключів, values() – для отримання всіх значень, а також items() – для отримання всіх пар ключ-значення.

Текст програми:

```
print("=== Тестування методів словників ===")

dict = {'a':1,'b':2,'c':3}

print("Початковий словник:     ",dict)
```

```
dict.update({'d':4, 'b':20})

print("update({'d':4, 'b':20}):",dict)

del dict['a']

print("del dict['a']:          ",dict)

print("keys():                ", list(dict.keys()))

print("values():              ", list(dict.values()))

print("items():               ", list(dict.items()))

dict.clear()

print("clear():               ",dict)
```

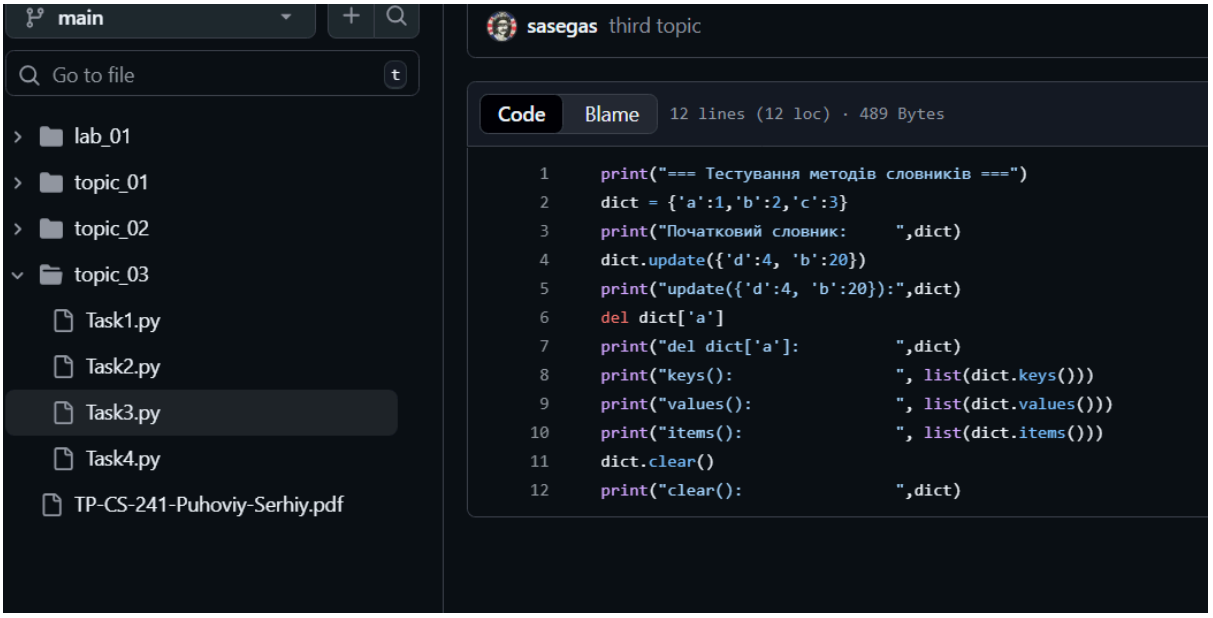

Рис.3 Скріншот третього завдання з GitHub

Пошук позиції для вставки

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

Хід виконання завдання:

В цьому завданні я зробив цикл, який вираховує куди треба вставити елемент у відсортований список, суть циклу в перебиранні списку з середини і в залежності від значення рухається назад, або вперед, після

сортування за допомогою метода include вставляється новий елемент в список.

Текст програми:

```
def find_insert_position(sorted_list, value):
    left = 0
    right = len(sorted_list)

    while left < right:
        mid = (left + right) // 2
        if sorted_list[mid] < value:
            left = mid + 1
        else:
            right = mid
    return left


my_list = [1, 3, 5, 7, 9]
new_value = 6
position = find_insert_position(my_list, new_value)
print(f"Новий елемент {new_value} слід вставити на позицію {position + 1}")


my_list.insert(position, new_value)
print("Список після вставки:", my_list)
```

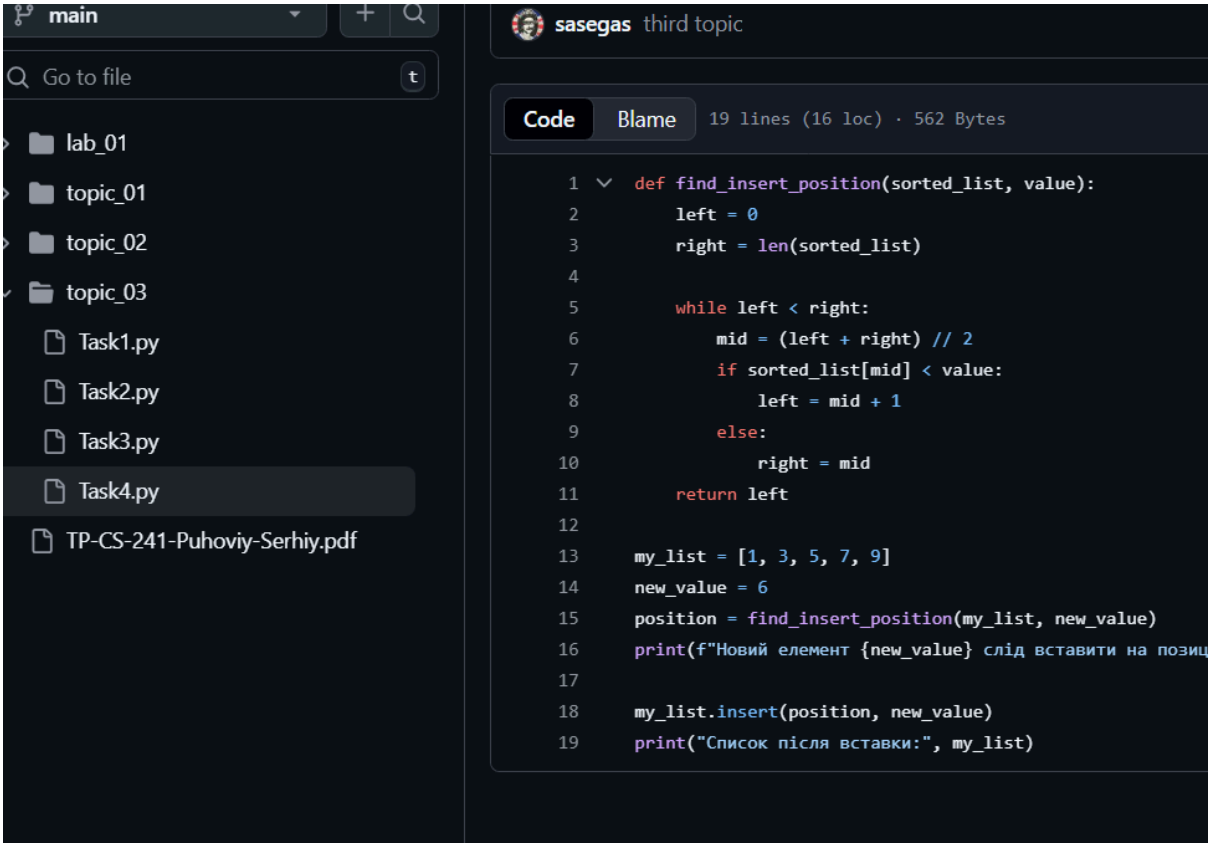

Рис.4 Скріншот четвертого завдання з GitHub

Звіт до Теми № 4

Виняткові ситуації

Обробка виняткових ситуацій в калькуляторі

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

Хід виконання завдання:

Для виконання цього завдання я додав обробку помилки ValueError, щоб програма не зупинялась, якщо був введений неправильний тип даних

Текст програми:

```
def plus(a, b):

    return a + b
```

```python
def minus(a, b):
    return a - b


def multiply(a, b):
    return a * b


def divide(a, b):
    return a / b




print("Щоб вийти, введіть 'exit' замість знака дії.")


while True:
    try:
        a = float(input("Введіть перше число: "))
        b = float(input("Введіть друге число: "))
    except ValueError:
        print("Потрібно вводити числа!")
        continue


    sign = input("Введіть дію (+ - * /) або 'exit' для виходу: ")


    if sign.lower() == "exit":
        print("Роботу завершено.")
        break


    match sign:
        case "+":
            result = plus(a, b)
        case "-":
            result = minus(a, b)
        case "*":
            result = multiply(a, b)
        case "/":
```

```
                result = divide(a, b)

            case _:

                result = "Була задана некоректна дія"


        print("Відповідь:", result, "\n")
```

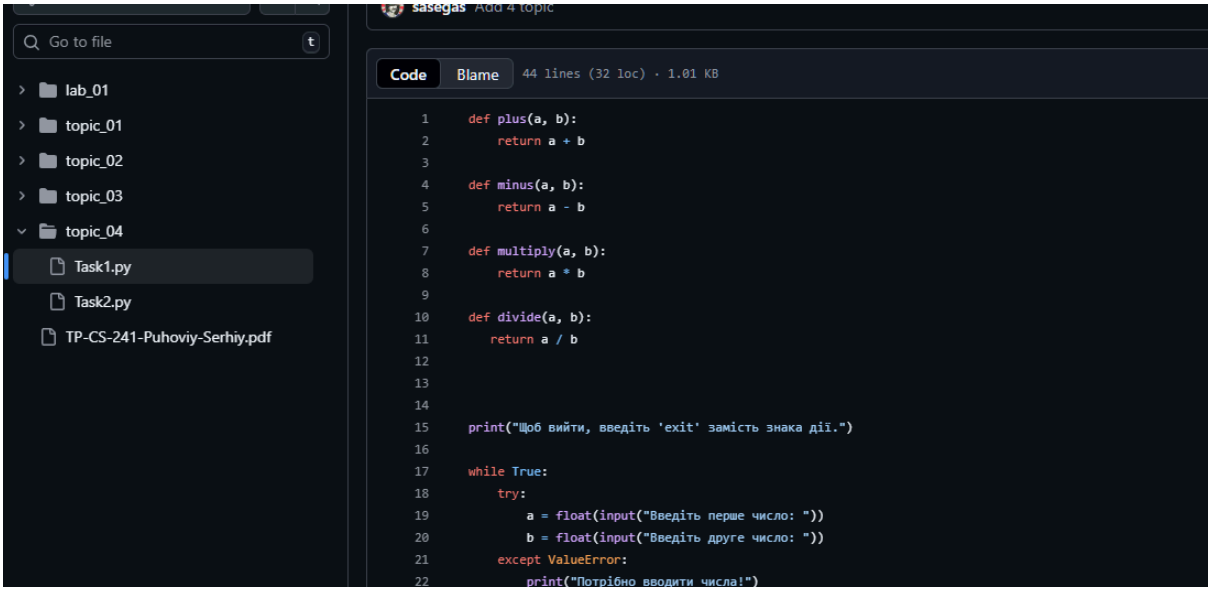

Рис.1 Скріншот першого завдання з GitHub

Обробка виняткової ситуації ділення на нуль в калькуляторі

Розширити функцію ділення обробкою виняткової ситуації ділення но нуль .

Хід виконання завдання:

Для виконання цього завдання я додав обробку помилки ZeroDivisionError в функцію ділення, щоб замість відповіді повертався текст помилки.

Текст програми:

```
def plus(a, b):

    return a + b


def minus(a, b):

    return a - b
```

```python
def multiply(a, b):
    return a * b


def divide(a, b):
    try:
        return a / b
    except ZeroDivisionError:
        return "Не можна ділити на нуль"



print("Щоб вийти, введіть 'exit' замість знака дії.")


while True:
    try:
        a = float(input("Введіть перше число: "))
        b = float(input("Введіть друге число: "))
    except ValueError:
        print("Потрібно вводити числа!")
        continue



    sign = input("Введіть дію (+ - * /) або 'exit' для виходу: ")


    if sign.lower() == "exit":
        print("Роботу завершено.")
        break


    match sign:
        case "+":
            result = plus(a, b)
        case "-":
            result = minus(a, b)
        case "*":
```

```
                result = multiply(a, b)
            case "/":
                result = divide(a, b)
            case _:
                result = "Була задана некоректна дія"

        print("Відповідь:", result, "\n")
```

Рис.2 Скріншот другого завдання з GitHub

Звіт до Теми № 5

Бібліотеки

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.

Хід виконання завдання:

Для виконання цього завдання спочатку я створив масив з варіантами значень, а потім створив функцію game, яка вираховує всі можливі варіанти закінчення раунду і створив головну функцію де задаються

значення гравця і комп'ютера, для того щоб комп’ютер обирав випадково варіанти я підключив бібліотеку random і передав змінні в функцію game.

Текст програми:

```
import random

options = ["stone", "scissor", "paper"]


def game(user, bot):
    if user == bot:
        print("Нічия")
    elif (
        (user == "stone" and bot == "scissor") or
        (user == "paper" and bot == "stone") or
        (user == "scissor" and bot == "paper")):
        print("Ви перемогли!")
    else:
        print("Комп'ютер переміг")


def main():
    while True:
        user_choise = input("Введіть своє значення (stone, scissor, paper) або 'exit' для виходу: ").lower()
        if user_choise == "exit":
            print("Гру завершено.")
            break
        elif user_choise not in options:
            print("Неправильне значення! Оберіть stone, scissor або paper.")
            continue
        else:
            computer_choice = random.choice(options)
```

```
            print(f"Комп’ютер обрав: {computer_choice}")
            game(user_choise, computer_choice)
            continue
    main()
```

Рис.1 Скріншот першого завдання з GitHub

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:

В цьому завданні спочатку я підключив модуль requests для обробки API, після чого підключив шлях по якому буду отримувати курс і створив функцію яка робить запит по API і повертає курси валют в форматі Json і записав за допомогою циклу потрібні значення в словник rates, після чого в функції main вивів поточний курс, після чого зробив конвертатор цих валют в гривні.

Текст програми:

```
import requests
```

```
URL =
"https://bank.gov.ua/NBUStatService/v1/statdirectory/exchange?json"


def get_exchange_rates():
    response = requests.get(URL)
    if response.status_code != 200:
        print("Помилка отримання даних з НБУ!")
        return None

    data = response.json()
    rates = {}

    for item in data:
        if item["cc"] in ["USD", "EUR", "PLN"]:
            rates[item["cc"]] = item["rate"]

    return rates


def main():
    rates = get_exchange_rates()
    if not rates:
        return

    print("Актуальні курси валют (за 1 одиницю):")
    for code, rate in rates.items():
        print(f"{code}: {rate} грн")

    while True:
        currency = input("\nВведіть валюту (USD, EUR, PLN) або 'exit' для
виходу: ").upper()
```

```
        if currency == "EXIT":
            print("Програму завершено.")
            break
        if currency not in rates:
            print("Неправильна валюта! Оберіть USD, EUR або PLN.")
            continue
        try:
            amount = float(input("Введіть суму: "))
            result = amount * rates[currency]
            print(f"{amount} {currency} = {result:.2f} грн")
        except ValueError:
            print("Введіть числове значення суми!")


main()
```

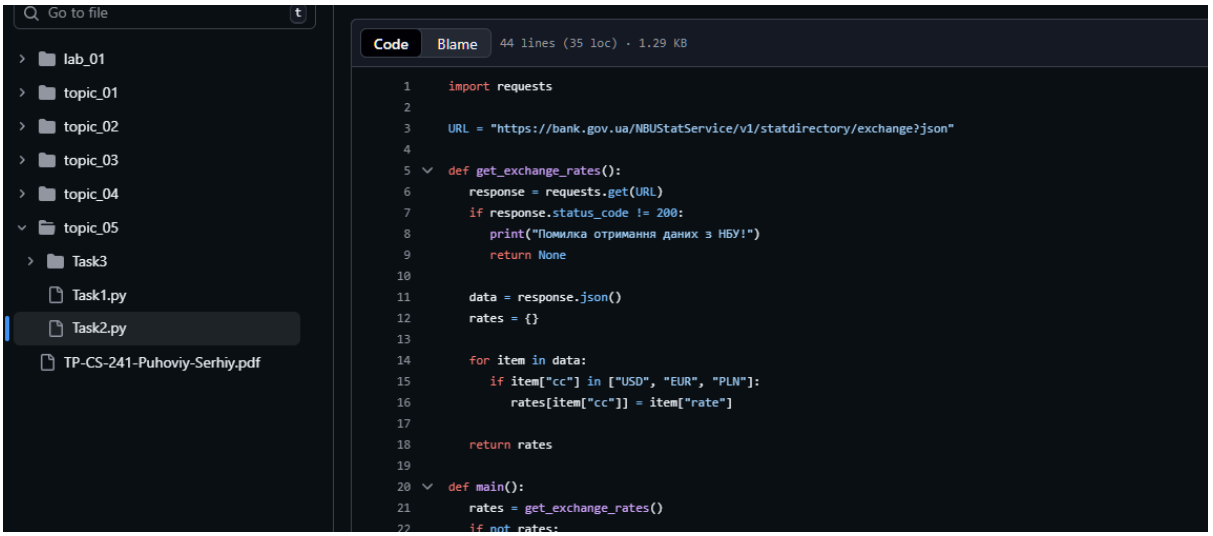

Рис.2 Скріншот першого завдання з GitHub

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:

Я створив 3 файли functions.py, operations.py, calc.py, після чого в першому файлі створив функції які виконують математичний обрахунок, в другому файлі я імпортував всі функції з першого і створив дві функції, перша функція отримує числа від користувача, а друга отримує знак і виконує функції з минулого файлу, а в головному файлі calc я імпортую функції операцій і спочатку отримую числа від користувачів, а потім передаю їх у функцію з обрахунком.

Текст програми:

functions.py

```
def plus(a, b):
    return a + b


def minus(a, b):
    return a - b


def multiply(a, b):
    return a * b


def divide(a, b):
    try:
        return a / b
    except ZeroDivisionError:
        return "Не можна ділити на нуль"
```

operations.py

```
from functions import plus, minus, divide, multiply
```

```python
def get_number():
    try:
        a = float(input("Введіть перше число: "))
        b = float(input("Введіть друге число: "))
        return a, b
    except ValueError:
        print("Помилка: введіть числове значення!")
        return None, None


def perform_operation(a, b):
    while True:
        sign = input("Введіть дію (+ - * /) або 'exit' для виходу: ")

        if sign == "exit":
            print("Вихід з програми.")
            break

        match sign:
            case "+":
                print(f"Результат: {plus(a, b)}")
            case "-":
                print(f"Результат: {minus(a, b)}")
            case "*":
                print(f"Результат: {multiply(a, b)}")
            case "/":
                print(f"Результат: {divide(a, b)}")
            case _:
                print("Була задана некоректна дія.")
```

calc.py

```python
from operations import perform_operation, get_number


def main():
    a, b = get_number()
    perform_operation(a,b)
main()
```

Рис.3 Скріншот першого завдання з GitHub