

# **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 елемента 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)
```

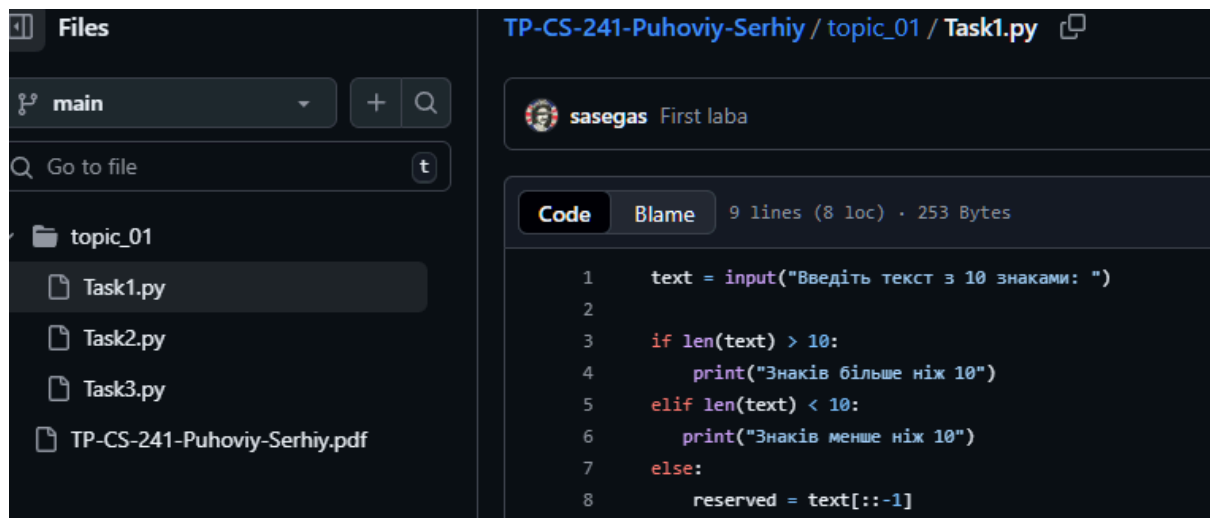


Рис.1 Скріншот першого завдання з GitHub

Посилання: [https://github.com/sasegas/TP-CS-241-Puhoviy-Serhiy/blob/main/topic\\_01/Task1.py](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 регістрі,

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())
```

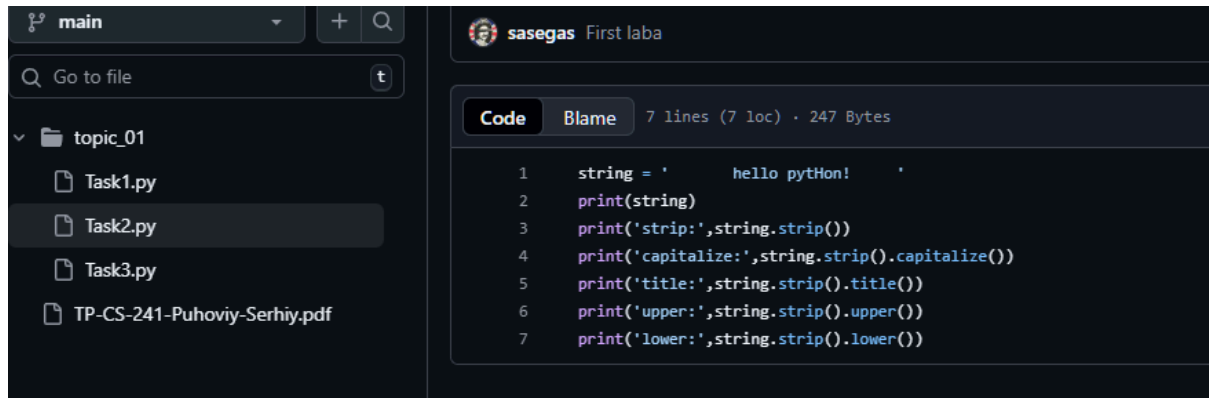


Рис.2 Скріншот другого завдання з GitHub

Посилання:[https://github.com/sasegas/TP-CS-241-Puhoviy-Serhiy/blob/main/topic\\_01/Task2.py](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)

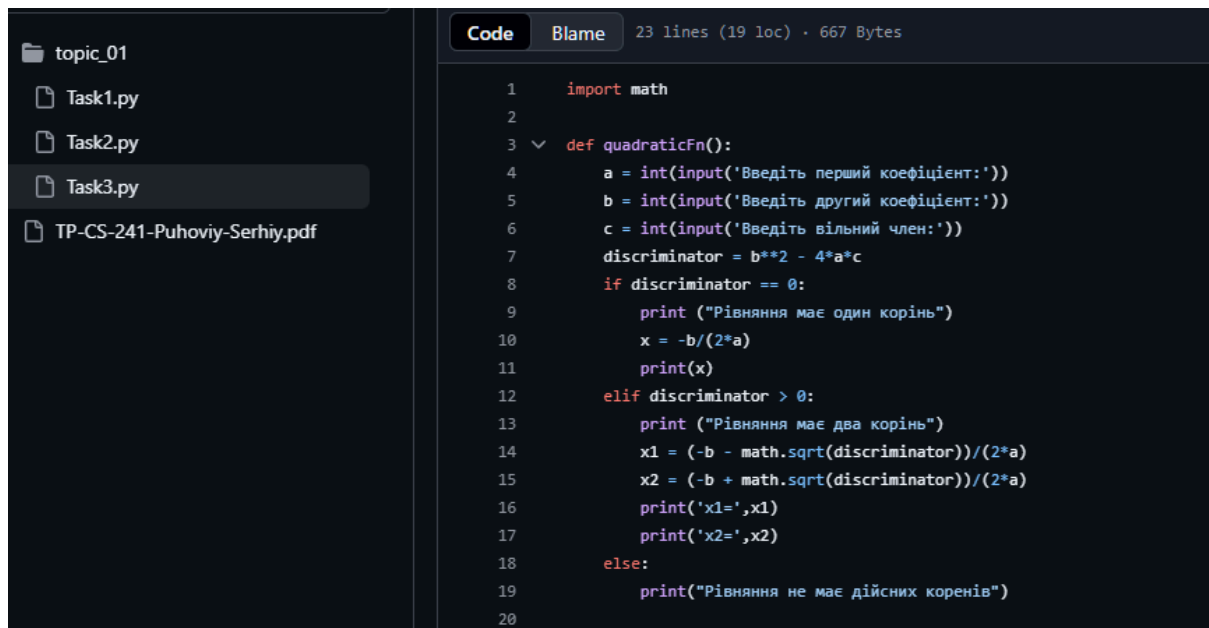
        print('x1=',x1)

        print('x2=',x2)

    else:

        print("Рівняння не має дійсних коренів")

quadraticFn()
```



```
1 import math
2
3 def quadraticFn():
4     a = int(input('Введіть перший коефіцієнт:'))
5     b = int(input('Введіть другий коефіцієнт:'))
6     c = int(input('Введіть вільний член:'))
7     discriminator = b**2 - 4*a*c
8     if discriminator == 0:
9         print("Рівняння має один корінь")
10        x = -b/(2*a)
11        print(x)
12    elif discriminator > 0:
13        print("Рівняння має два коріння")
14        x1 = (-b - math.sqrt(discriminator))/(2*a)
15        x2 = (-b + math.sqrt(discriminator))/(2*a)
16        print('x1=', x1)
17        print('x2=', x2)
18    else:
19        print("Рівняння не має дійсних коренів")
20
```

Рис.3 Скріншот третього завдання з GitHub

Посилання: [https://github.com/sasegas/TP-CS-241-Puhoviy-Serhiy/blob/main/topic\\_01/Task2.py](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)

```

main

+

Q

Go to file

t

lab\_01

topic\_01

topic\_02

Task1.py

Task2.py

Task3.py

TP-CS-241-Puhoviy-Serhiy.pdf

sasegas Second topic

Code Blame 29 lines (22 loc) · 666 Bytes

```

1  import math
2
3  def discriminant(a,b,c):
4      return b**2 - 4*a*c
5
6
7
8  def quadratic_roots(a, b, c):
9      d = discriminant(a, b, c)
10
11     if d > 0:
12         print("Рівняння має два корені")
13         x1 = (-b - math.sqrt(d)) / (2*a)
14         x2 = (-b + math.sqrt(d)) / (2*a)
15         print("x1 =", x1)
16         print("x2 =", x2)
17         return x1, x2
18     elif d == 0:
19         print("Рівняння має один корінь")
20         x = -b / (2*a)
21         print("x =", x)
22         return x,
23     else:
24         print("Рівняння не має дійсних коренів")
25         return None
26
27
28     a, b, c = 3, -18, 27
29     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, яка 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

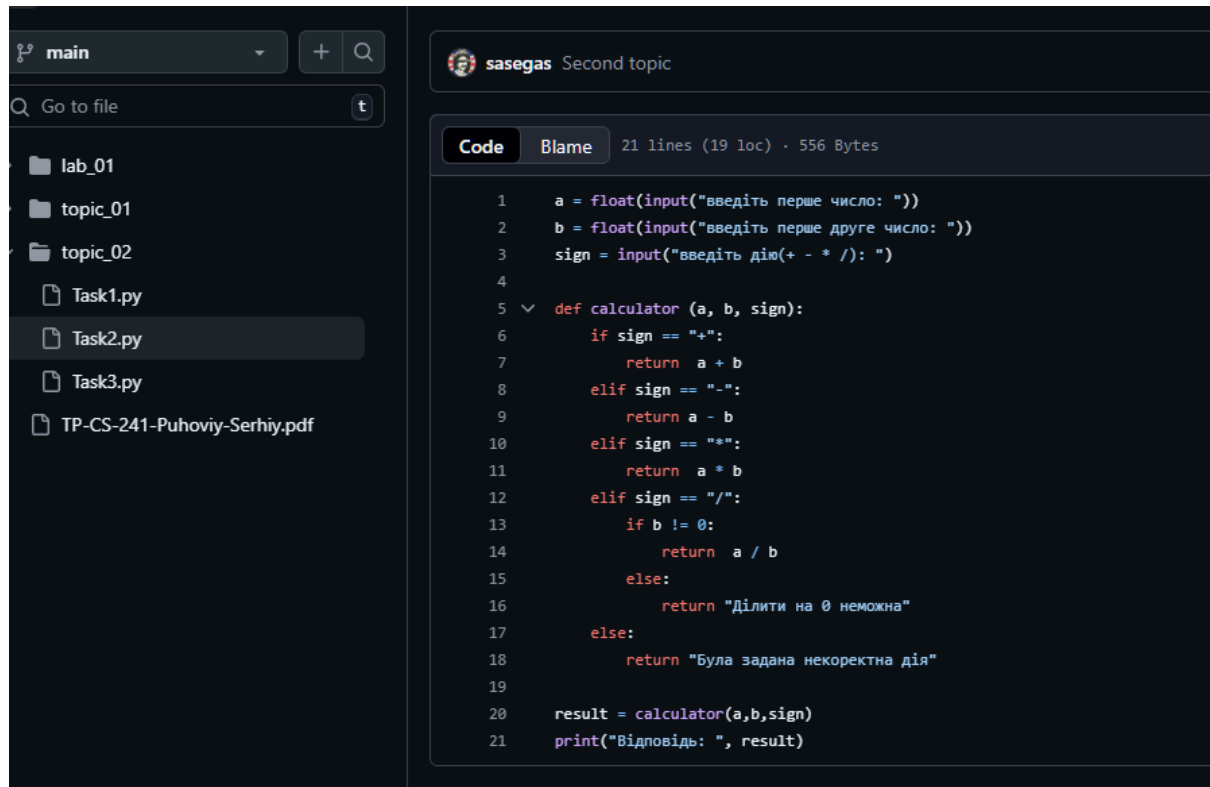
Текст програми:

```
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)
```



```
1 a = float(input("введіть перше число: "))
2 b = float(input("введіть перше друге число: "))
3 sign = input("введіть дію(+ - * /): ")
4
5 def calculator(a, b, sign):
6     if sign == "+":
7         return a + b
8     elif sign == "-":
9         return a - b
10    elif sign == "*":
11        return a * b
12    elif sign == "/":
13        if b != 0:
14            return a / b
15        else:
16            return "Ділити на 0 неможна"
17    else:
18        return "Була задана некоректна дія"
19
20 result = calculator(a,b,sign)
21 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 запуслав 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)
```

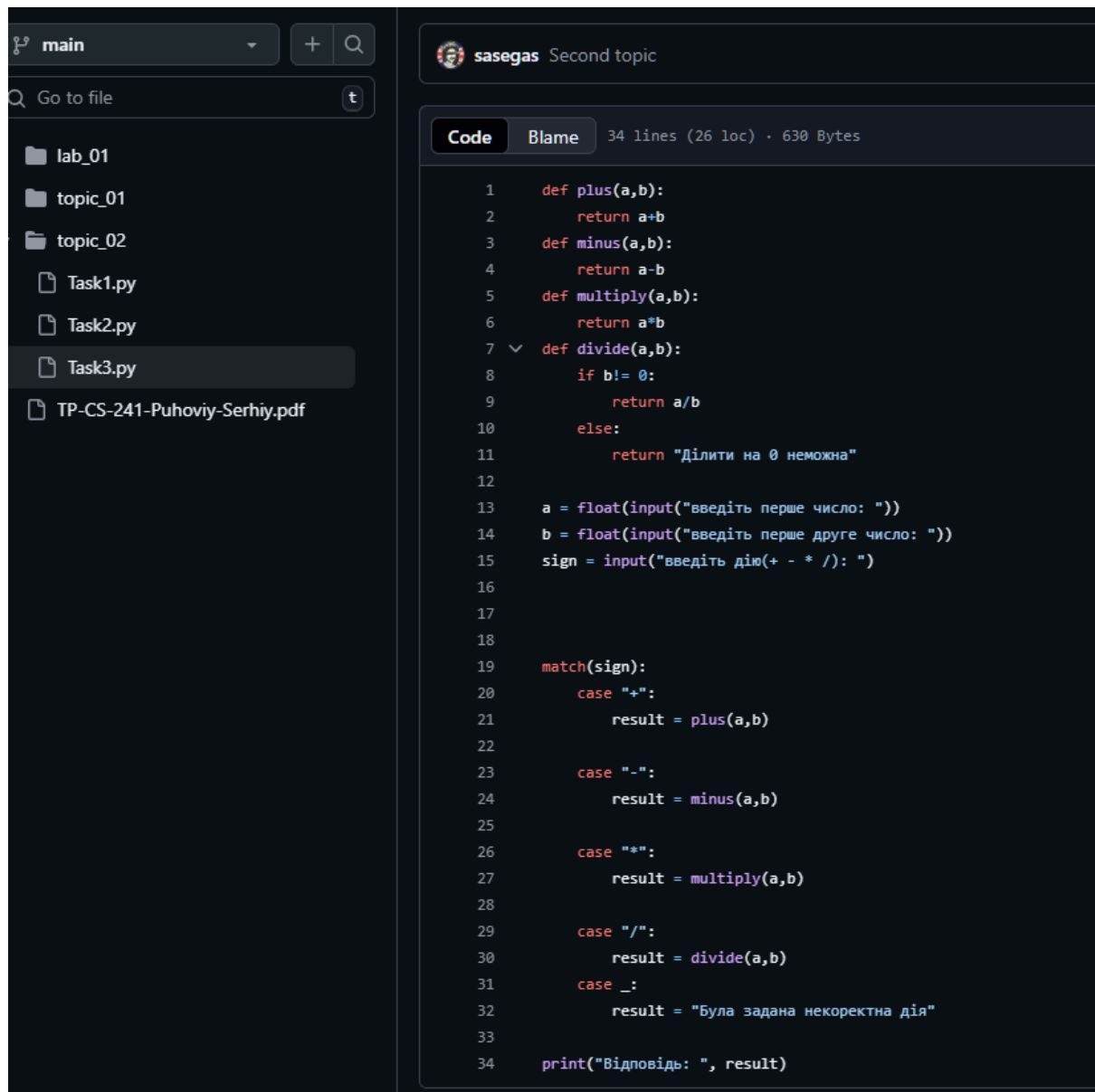


Рис.3 Скріншот третього завдання з GitHub

## 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.")
```

```
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")
```

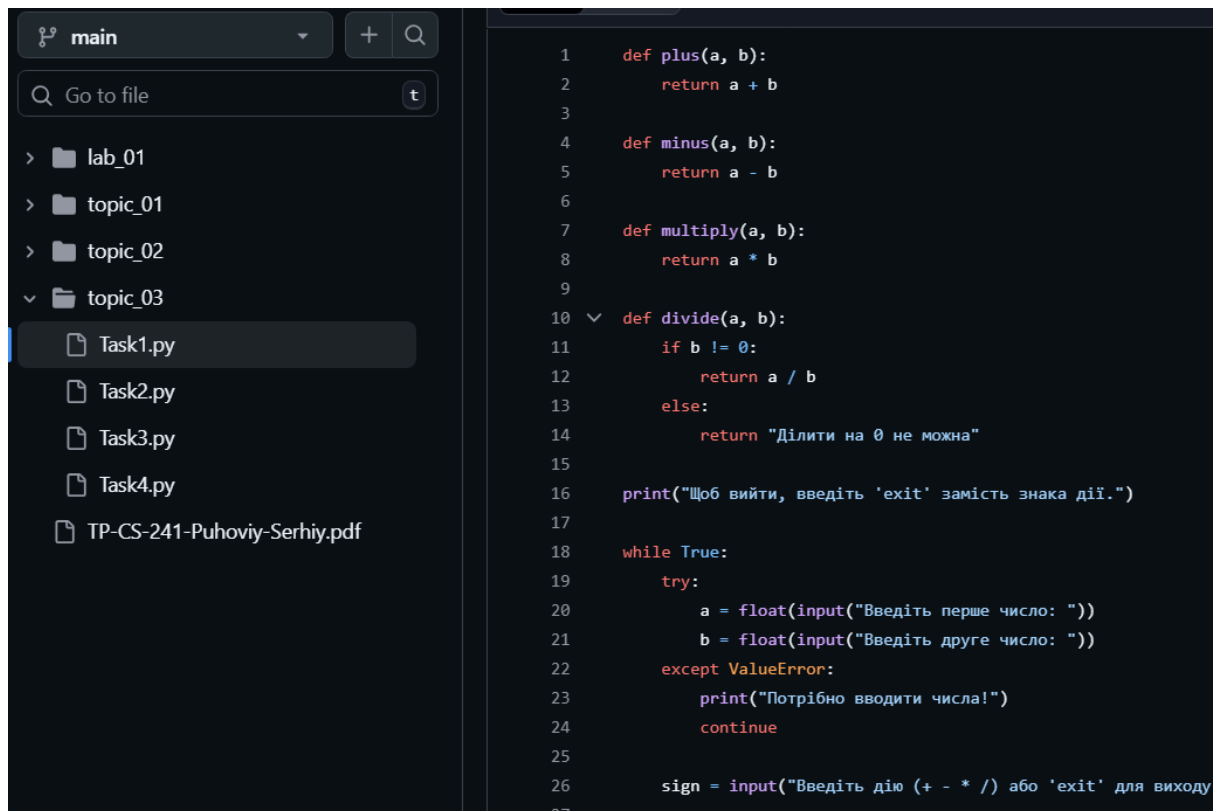


Рис.1 Скріншот першого завдання з GitHub

## 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)
```

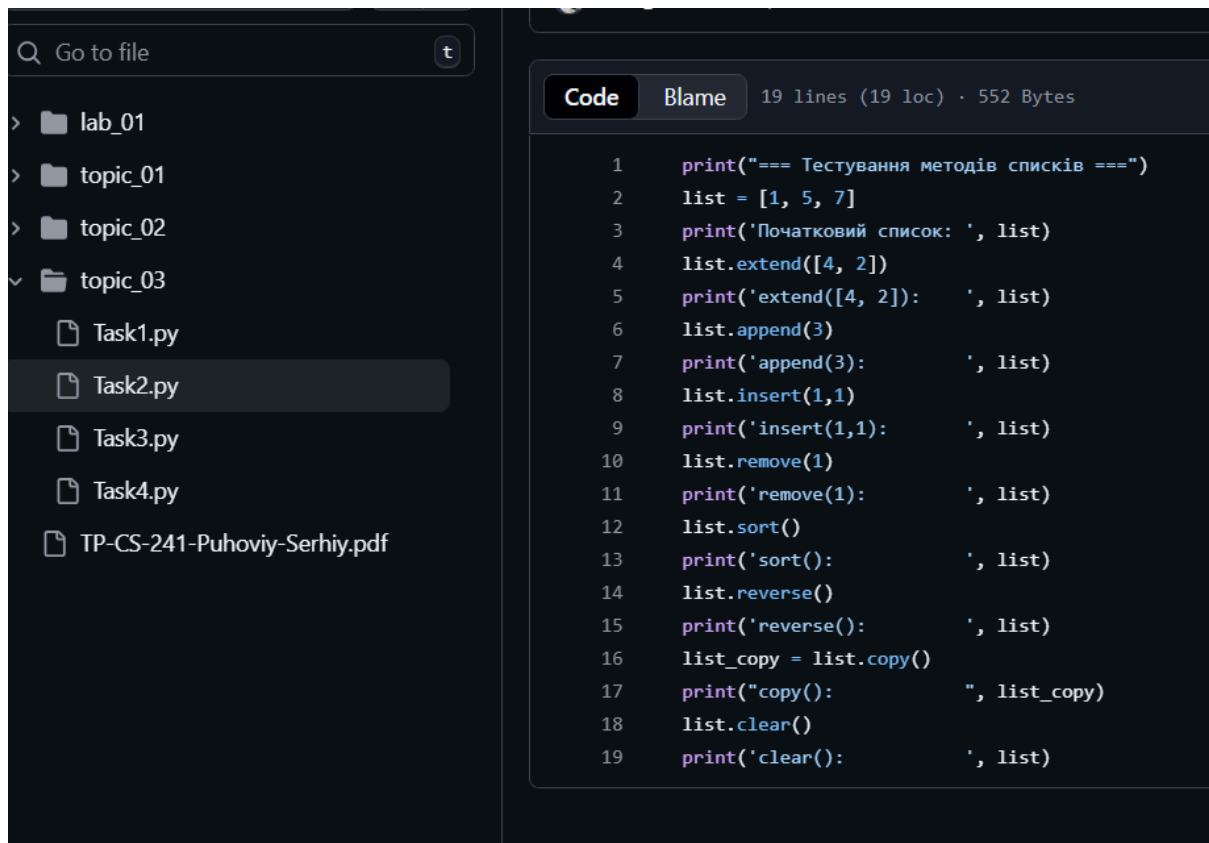


Рис.2 Скріншот другого завдання з GitHub

### 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)
```

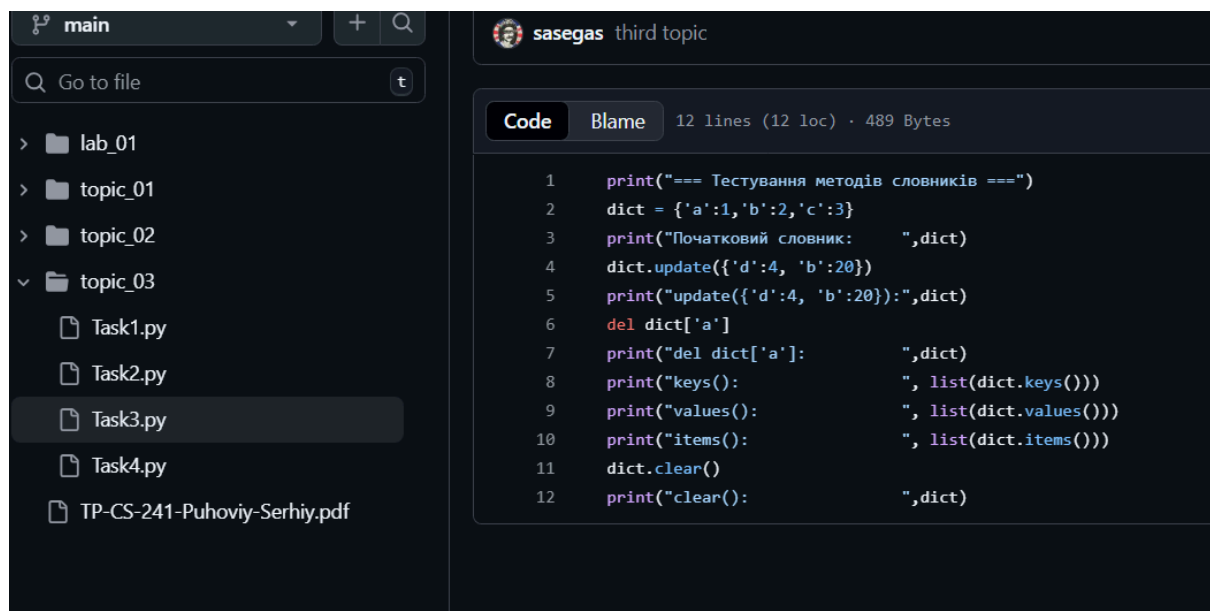


Рис.3 Скріншот третього завдання з GitHub

### 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 `insert` 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)
```

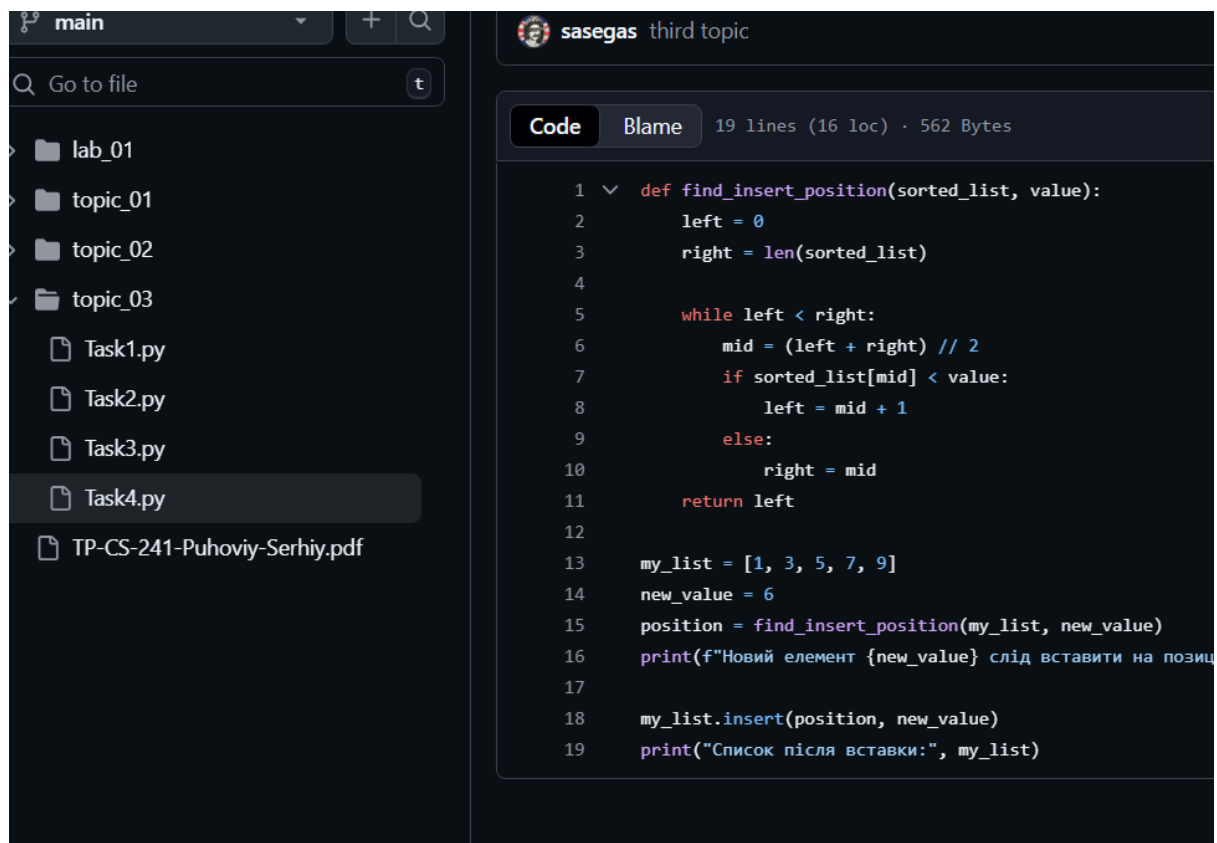


Рис.4 Скріншот четвертого завдання з GitHub