

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)
```

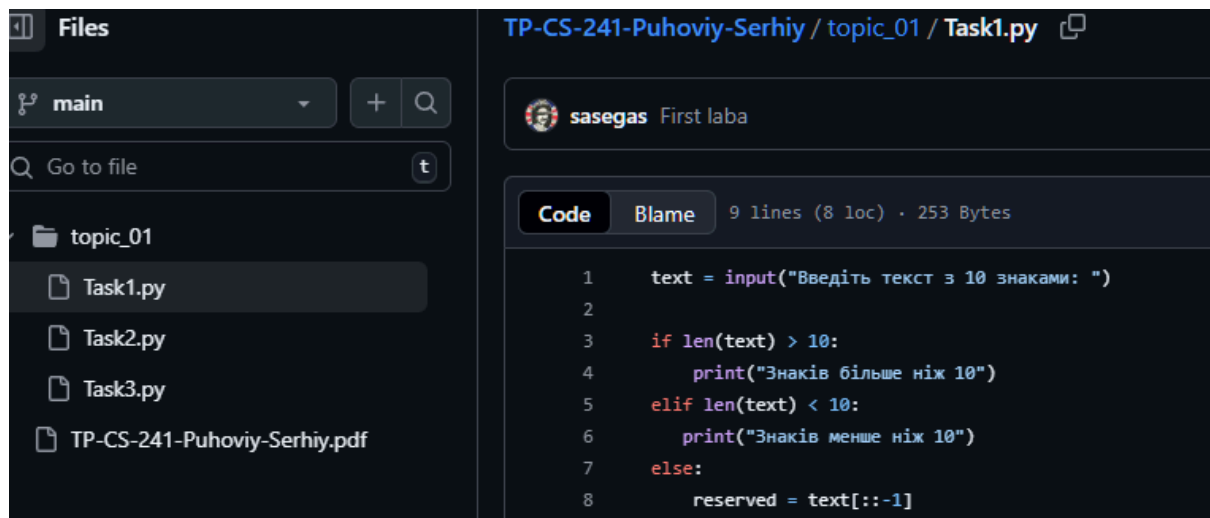


Рис.1 Скріншот першого завдання з GitHub

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

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())
```

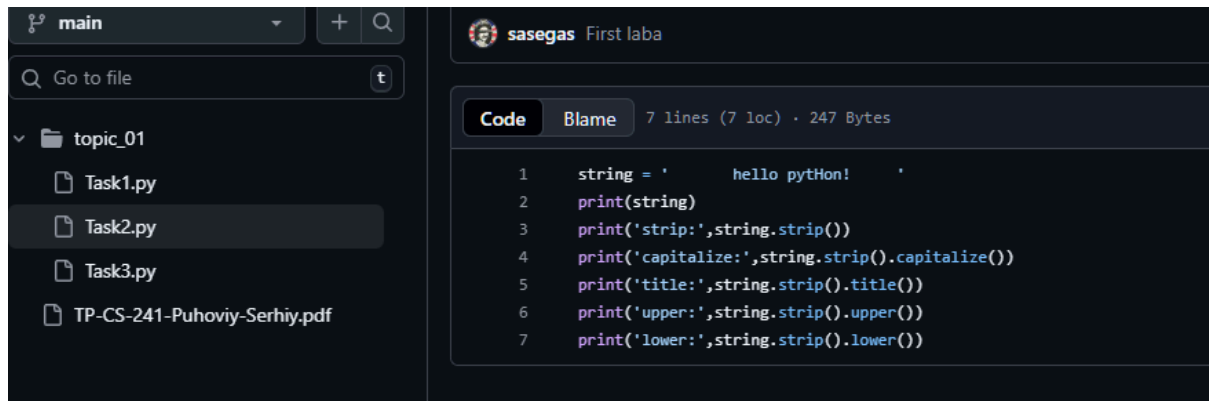


Рис.2 Скріншот другого завдання з GitHub

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

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

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)

```

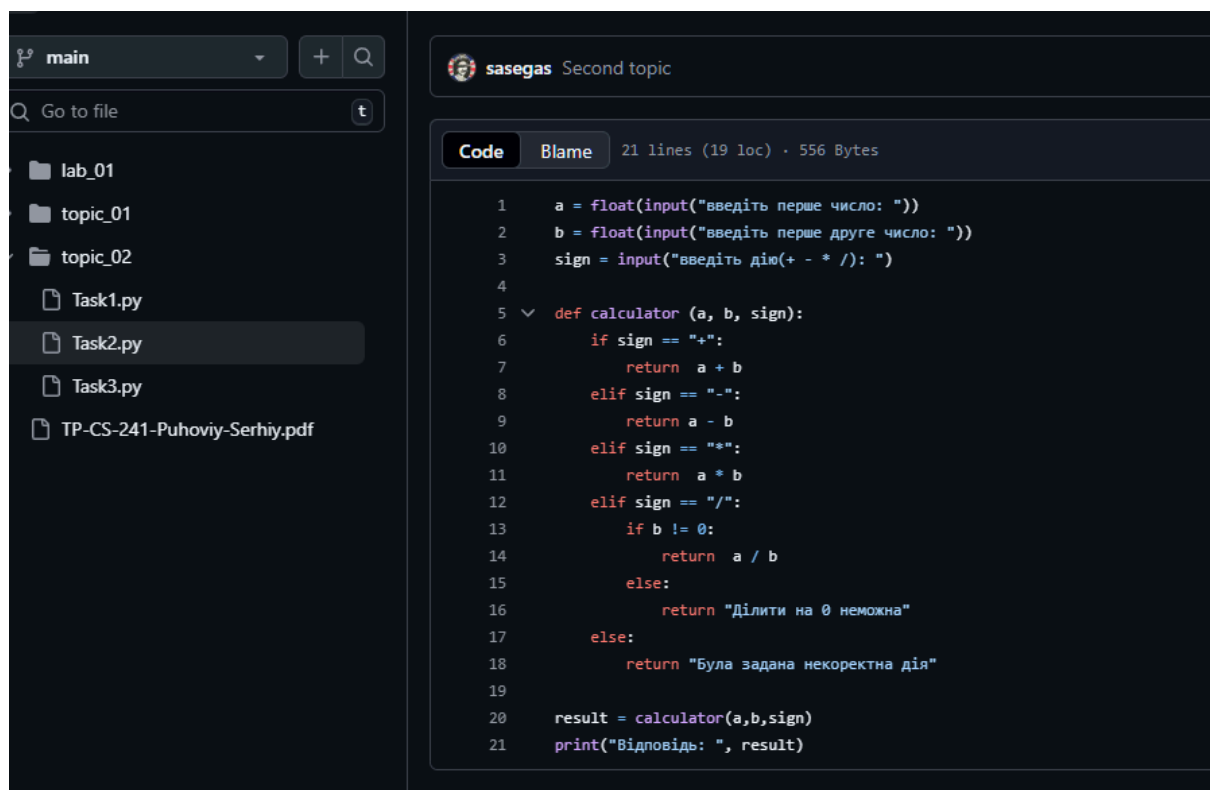


Рис.2 Скріншот другого завдання з GitHub

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)
```

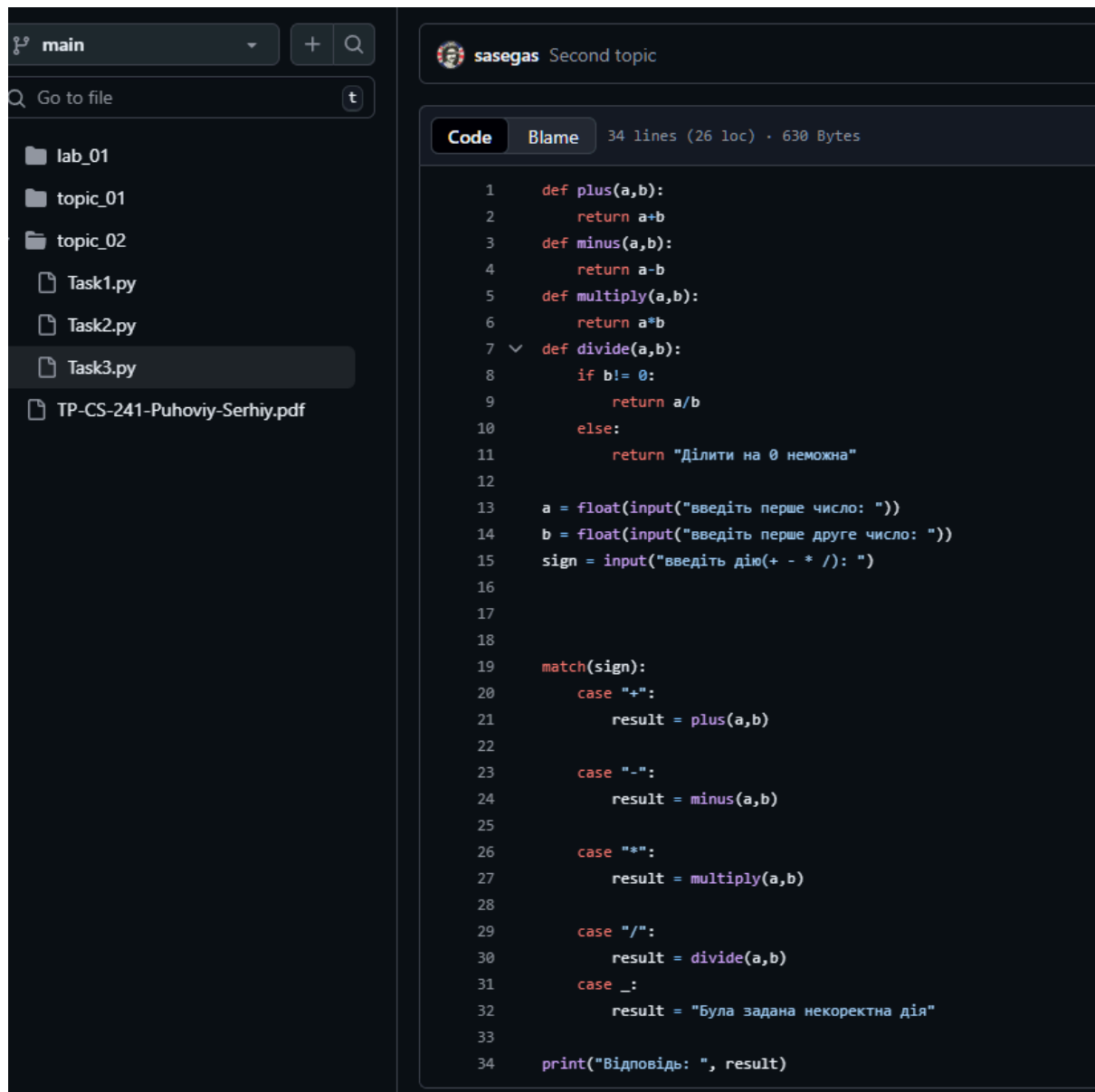


Рис.3 Скріншот третього завдання з GitHub

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.")

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")
```

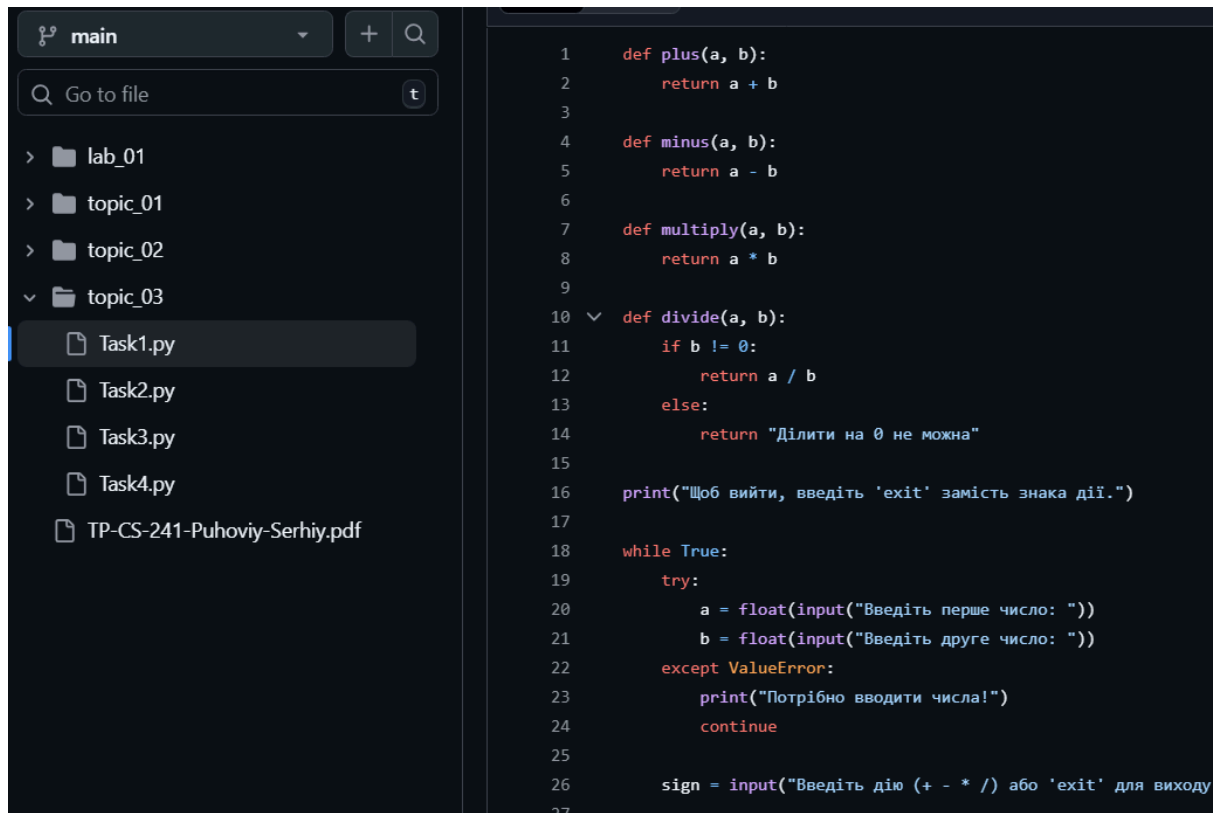


Рис.1 Скріншот першого завдання з GitHub

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)
```

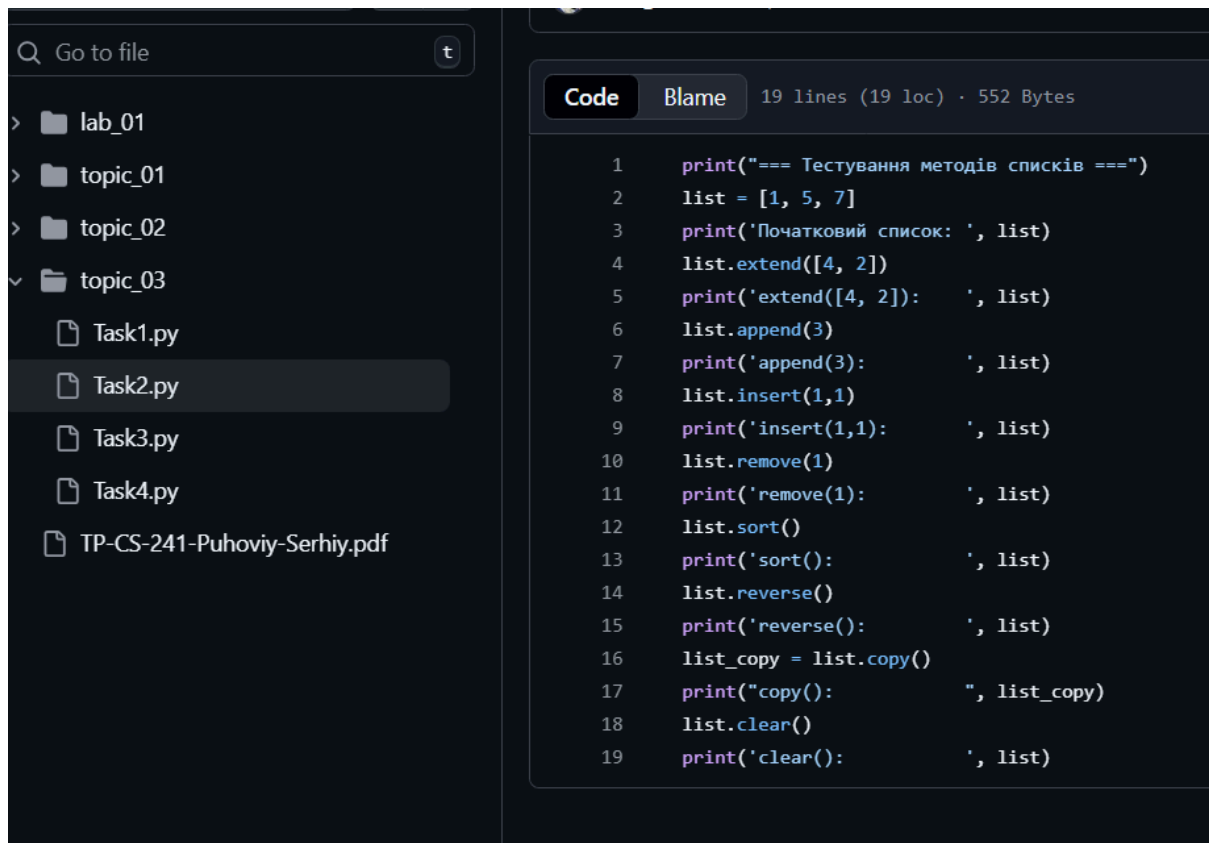


Рис.2 Скріншот другого завдання з GitHub

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)
```

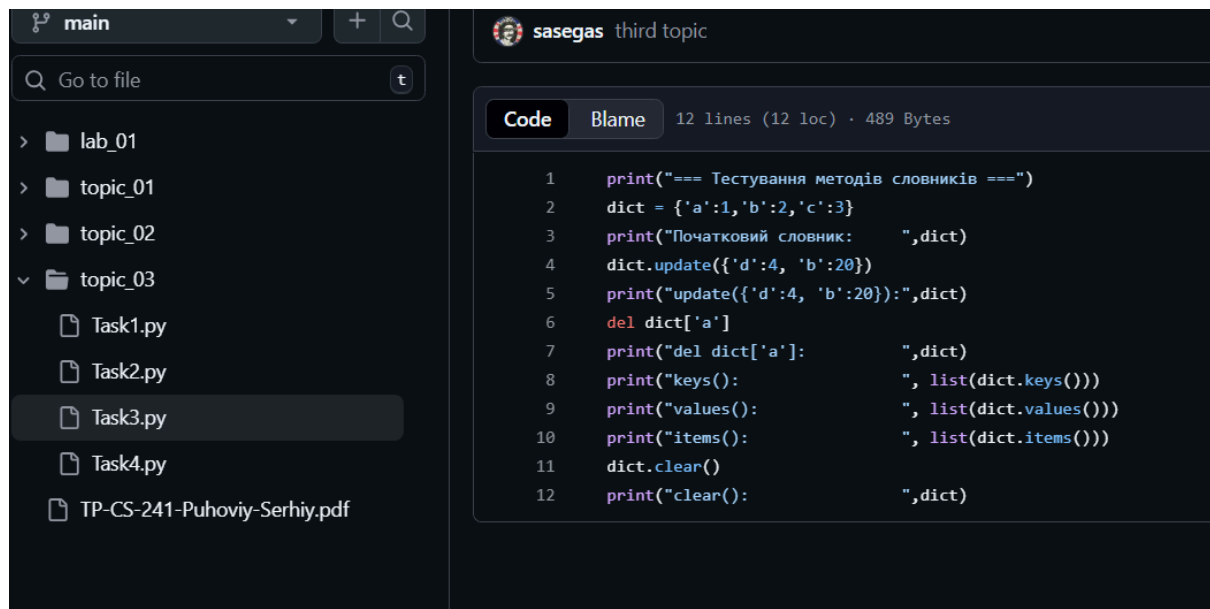


Рис.3 Скріншот третього завдання з GitHub

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)
```

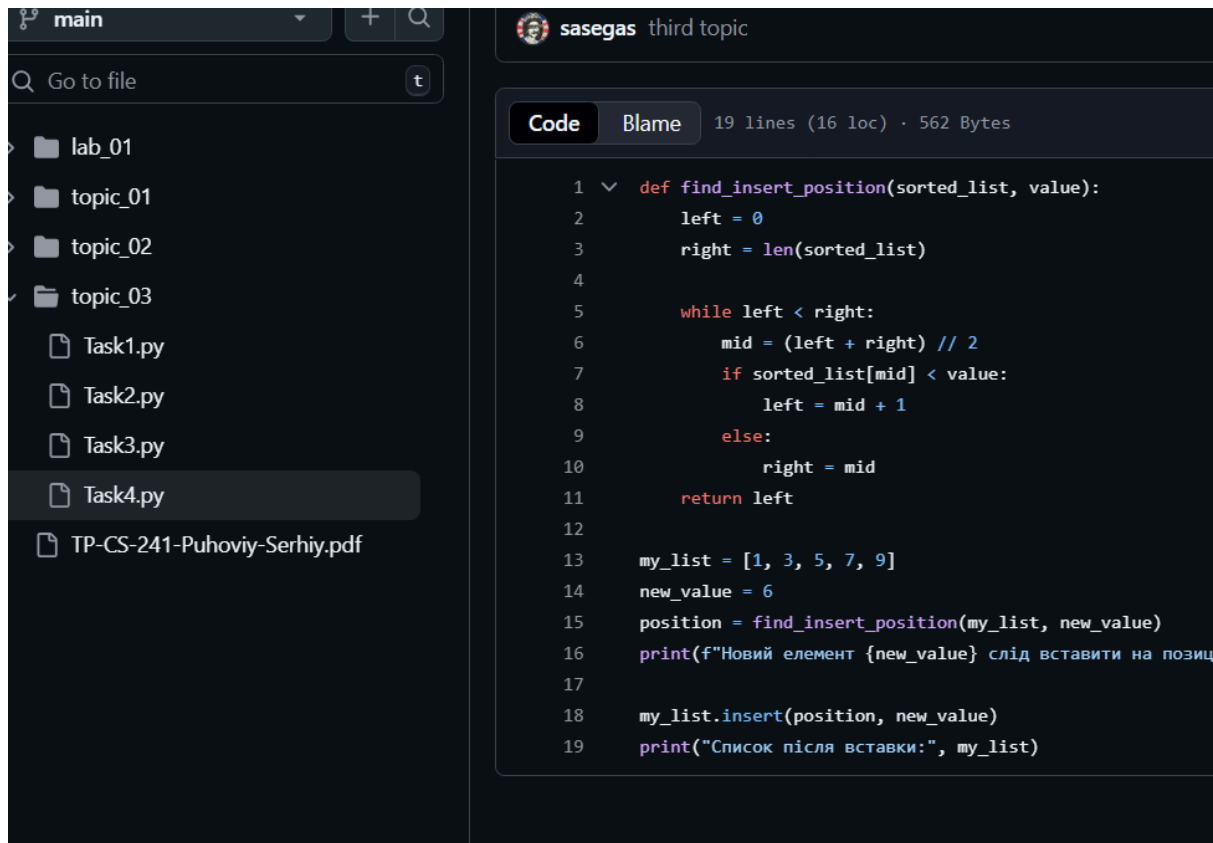


Рис.4 Скріншот четвертого завдання з GitHub

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
```

```
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")

```

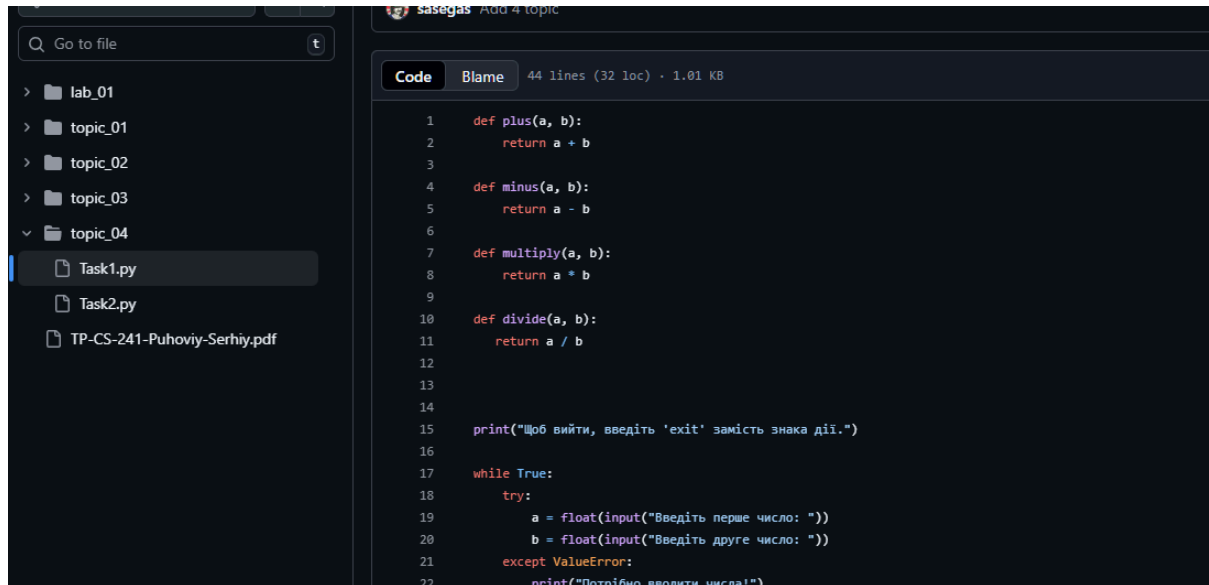


Рис.1 Скріншот першого завдання з GitHub

Обробка виняткової ситуації ділення на нуль в калькуляторі

Розширити функцію ділення обробкою виняткової ситуації ділення на 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

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()
```

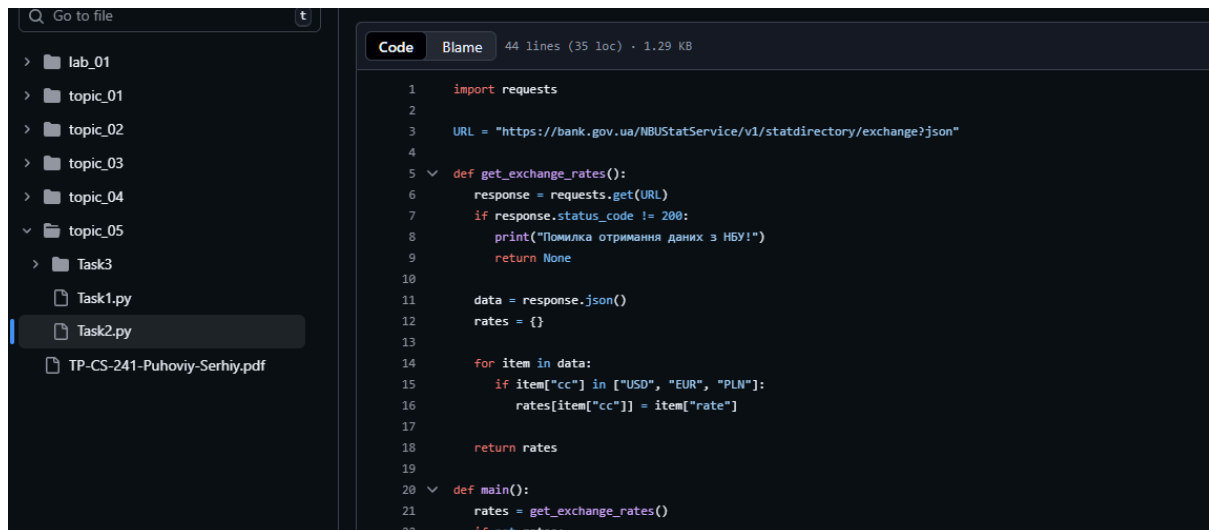


Рис.2 Скріншот другого завдання з GitHub

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 функцію `perform_operation`

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

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():

    while True:

        a, b = get_number()

        if a is None or b is None:

            continue

        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

```
from operations import perform_operation  
  
def main():  
    perform_operation()  
  
if __name__ == "__main__":  
    main()
```

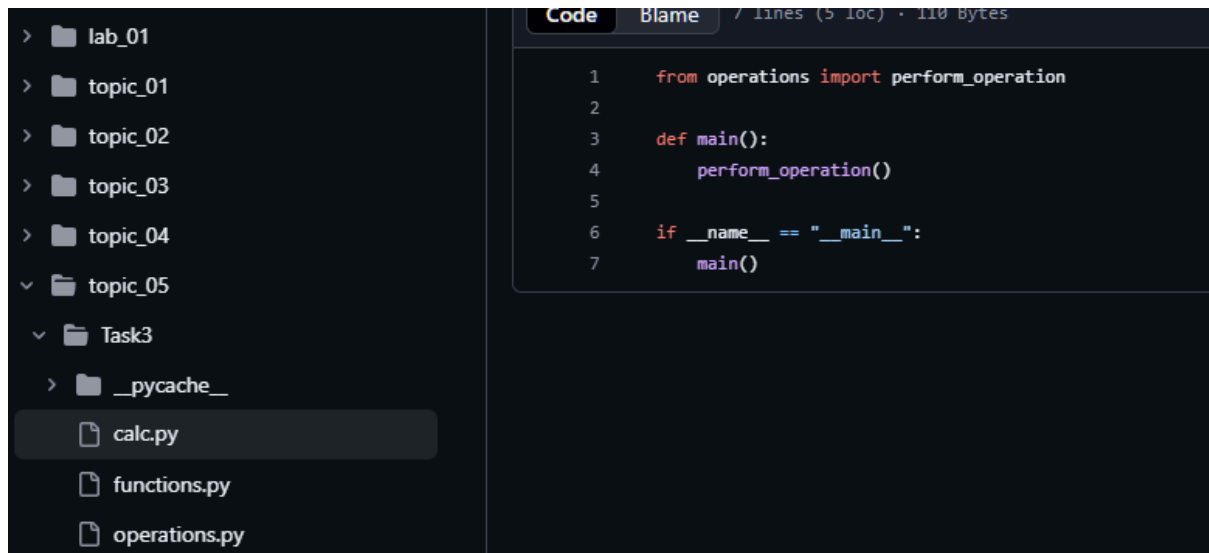


Рис.3 Скріншот третього завдання з GitHub

Звіт до Теми № 6

Робота з файлами

Розробити механізм логування всіх дій, що виконує програма

Забезпечити зберігання інформації про введені дані, виконану операцію та результат виконання операції над даними.

Хід виконання завдання:

Спочатку я створив файл в якому будуть зберігатись логи,потім в файлі functions.py задав змінну з назвою файлу, після чого створив функцію log_custom в яку передаю п'ять значень, 3 обов'язкових і 2 додаткових, в залежності від додаткових значень result або error визначається який матиме статус лог, після чого роблю рядок тексту, який потім вставляю в файл з логами і додаю до кожної дії обробку цієї функції. В файлі operations.py роблю логування початку і завершення роботи, а також обробку декількох помилок.

Текст програми:

functions.py

```
LOG_FILE = "calculator_log.txt"
```

```
def log_custom(sign, a, b, result=None, error=None):

    if error:

        log_level = "ERROR"

        status_info = f"Помилка: {error}"

    else:

        log_level = "INFO"

        status_info = f"Результат: {result}"

    log_entry = f"{log_level} - Операція: {a} {sign} {b} , {status_info}\n"

    with open(LOG_FILE, "a", encoding="utf-8") as file:

        file.write(log_entry)


def plus(a, b):

    result = a + b

    log_custom("+", a, b, result)

    return result


def minus(a, b):

    result = a - b

    log_custom("-", a, b, result)

    return result


def multiply(a, b):
```



```

    result = a * b

    log_custom("*", a, b, result)

    return result

def divide(a, b):
    try:
        result = a / b

        log_custom("/", a, b, result)

        return result

    except ZeroDivisionError:
        error_msg = "Не можна ділити на нуль"

        log_custom("/", a, b, error=error_msg)

        return error_msg

```

operations.py

```

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

def get_number():
    try:
        a = float(input("Введіть перше число: "))
        b = float(input("Введіть друге число: "))

        return a, b

    except ValueError:
        error_msg = "Помилка: введіть числове значення!"

        print(error_msg)

        log_custom("err", "N/A", "N/A", error="Введено не числове значення")

        return None, None

```

```
def perform_operation():  
    log_custom("?", "N/A", "N/A", result="Початок роботи")  
  
    while True:  
        a, b = get_number()  
        if a is None or b is None:  
            continue  
  
        sign = input("Введіть дію (+ - * /) або 'exit' для виходу:  
")  
  
        if sign == "exit":  
            print("Вихід з програми.")  
  
            log_custom("?", "N/A", "N/A", result="Завершення  
роботи")  
  
            break  
  
        result = None  
        match sign:  
            case "+":  
                result = plus(a, b)  
            case "-":  
                result = minus(a, b)  
            case "*":  
                result = multiply(a, b)  
            case "/":  
                result = divide(a, b)  
            case _:
```

```

        error_msg = "Була задана некоректна дія."
        print(error_msg)

        log_custom("err", a, b, error=f"Невідома дія - {sign}")

        continue

    print(f"Результат: {result}")

```

```

Code Blame 42 lines (32 loc) · 1011 Bytes
1 LOG_FILE = "calculator_log.txt"
2
3 def log_custom(sign, a, b, result=None, error=None):
4
5     if error:
6         log_level = "ERROR"
7         status_info = f"Помилка: {error}"
8     else:
9         log_level = "INFO"
10        status_info = f"Результат: {result}"
11
12    log_entry = f"{log_level} - Операція: {a} {sign} {b} , {status_info}\n"
13
14
15    with open(LOG_FILE, "a", encoding="utf-8") as file:
16        file.write(log_entry)
17
18
19 def plus(a, b):
20     result = a + b
21     log_custom("+", a, b, result)
22     return result
23

```

Рис.1 Скріншот першого завдання з GitHub

Сортування списку

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.

Хід виконання завдання:

Спочатку я імпортував csv, потім створив пустий масив для майбутніх даних, далі дістав з файлу table.csv дані про студентів та додав їх до масиву і виконав сортування використовуючи lambda

Текст програми:

```
import csv

studentList = []

with open("table.csv") as file:

    reader = csv.DictReader(file, skipinitialspace=True)

    for row in reader:

        studentList.append({"name":row["StudentName"],
"mark":int(row["StudentMark"])}))

print("--- Сортування ---")

for elem in sorted(studentList, key=lambda x: x['mark']):

    print(f"Name = {elem['name']} mark = {elem['mark']}")
```

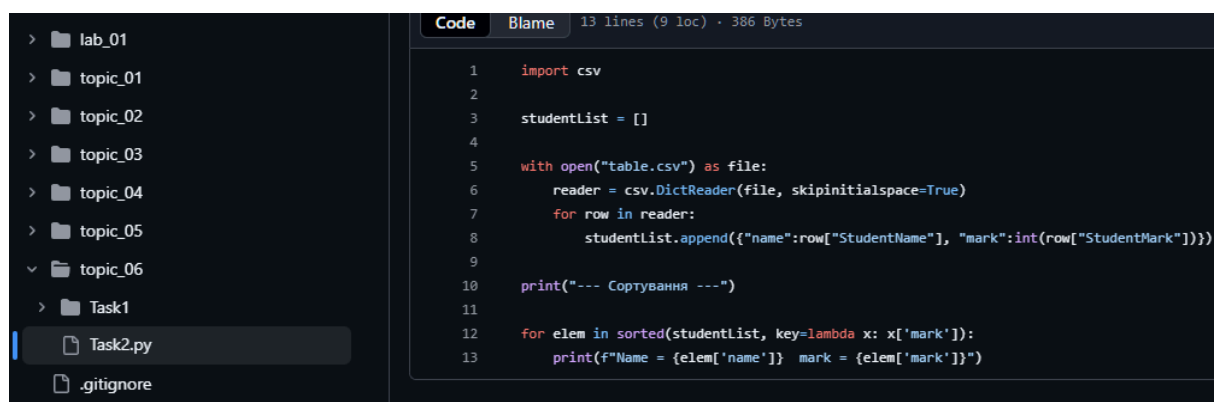


Рис.2 Скріншот другого завдання з GitHub

Звіт до Теми № 7

Об'єктно-орієнтоване програмування

Ознайомитись з існуючими за замовченням методами класу

Ознайомитись з існуючими за замовченням методами класу по типу `__init__(self)` `__str__(self)` та надати приклади використання.

Хід виконання завдання:

Для прикладу я використав три методи, перший це `__init__` - це конструктор, який запускається автоматично при створенні нового екземпляра класу, `__str__` - відповідає за те щоб повертати зрозумілий і читабельний опис об'єкта для користувача і `__repr__` - відповідає за представлення об'єкта для розробника, я зробив клас `Book` який спочатку отримує дані, потім обробляє їх для клієнта і для розробника, потім я задав змінну з цим класом в якій я задав об'єкт.

Текст програми:

```
class Book:

    def __init__(self, title, author):

        self.title = title

        self.author = author

    def __str__(self):

        return f"Книга '{self.title}', автор: {self.author}"

    def __repr__(self):

        return f"Book(title='{self.title}', author='{self.author}')
```

```

my_book = Book("Дюна", "Герберт Френк")

def main():

    print(my_book)

    print(repr(my_book))

main()

```

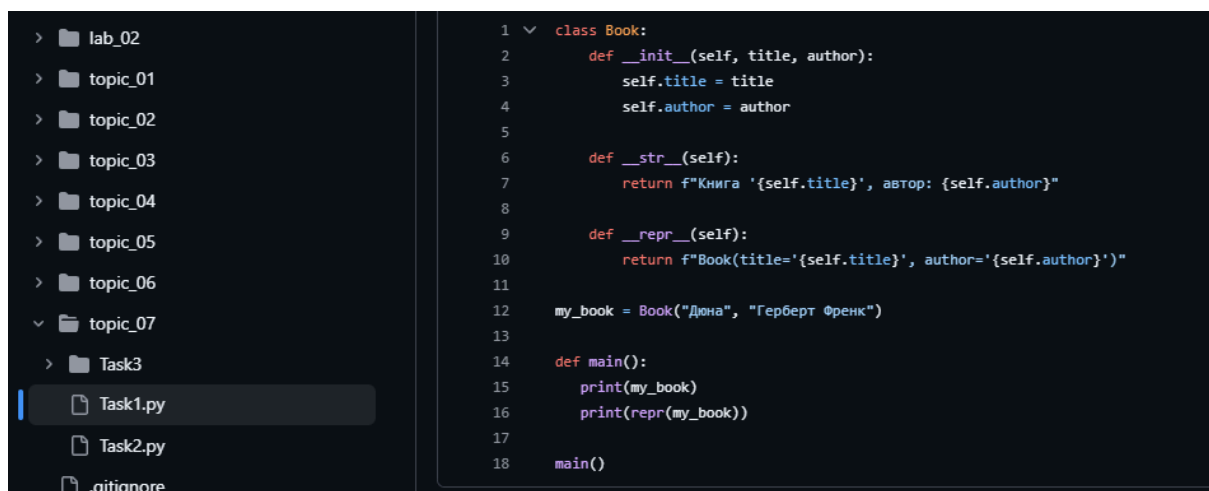


Рис.1 Скріншот першого завдання з GitHub

Розробити клас Student

Розробити клас Student атрибутами якого є 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.

Хід виконання завдання:

Спочатку я створив клас Student в якому ініціалізував ім'я та вік студента, після чого задав коректний вивід даних, потім створив масив зі

студентами, та в main функції виконав 2 сортування в яких ключ визначала lambda, та одне сортування виконувалось за віком, інше за ім'ям, потім за допомогою циклу виводив відсортовані списки.

Текст програми:

```
class Student:

    def __init__(self, name, age):

        self.name = name

        self.age = age

    def __str__(self):

        return f"Студент: {self.name}, Вік: {self.age}"

students = [

    Student("Андрій", 21),

    Student("Олена", 19),

    Student("Богдан", 22),

    Student("Вікторія", 20)

]

def main():

    sorted_by_age = sorted(students, key=lambda student: student.age)

    print("--- Сортування за віком (від найменшого) ---")

    for student in sorted_by_age:

        print(student)

    sorted_by_name = sorted(students, key=lambda student: student.name)
```

```

print("\n--- Сортювання за ім'ям (алфавіт) ---")

for student in sorted_by_name:

    print(student)

main()

```

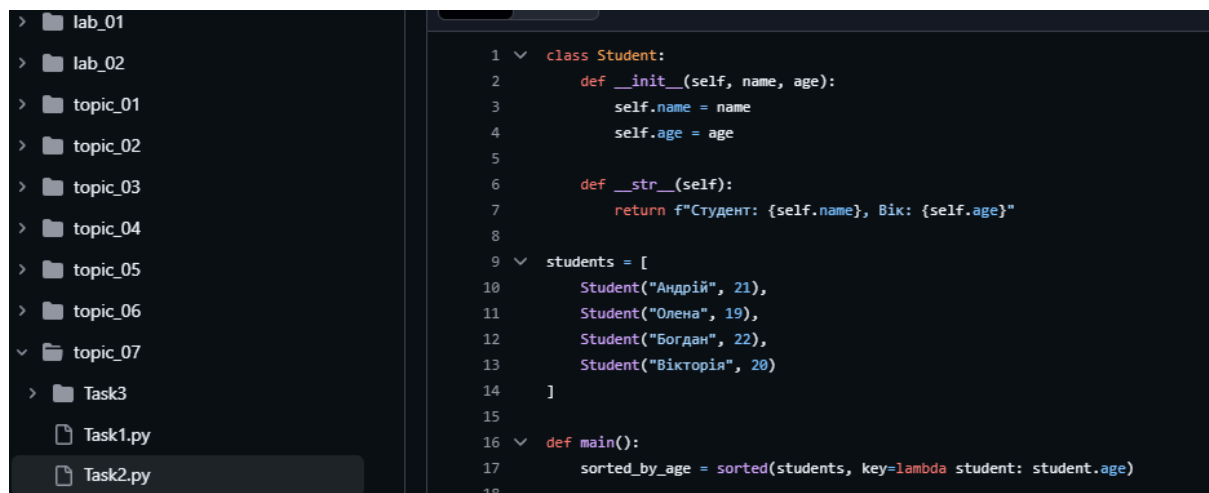


Рис.2 Скріншот другого завдання з GitHub

Калькулятор з використанням ООП

Використовуючи принципи ООП переписати програму Калькулятор. Завдання має бути виконано використовуючи модульний підхід.

Хід виконання завдання:

Спочатку я в файлі `function.py` створив клас `MathOperations`, в якому виконуються всі доступні операції, потім в файлі `operation.py` імпортував клас `MathOperations` і створив інший клас `Operator` де задав змінну `self.engine` яка має клас `MathOperations`, після чого виконую функцію де в залежності від вибору знаку буде виконуватись певна дія і в головному файлі `calc.py` в функції `main` створюється зміна з класом `Operator` і виконується введення даних і виведення результату.

Текст програми:

function.py

```
class MathOperations:
    def plus(self, a, b):
        return a + b

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

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

    def divide(self, a, b):
        if b != 0:
            return a / b
        else:
            raise ValueError("Ділення на нуль неможливе")
```

operation.py

```
from function import MathOperations

class Operator:
    def __init__(self):
        self.engine = MathOperations()

    def perform_calculation(self, a, b, sign):
        match sign:
            case "+":
                return self.engine.plus(a, b)
            case "-":
                return self.engine.minus(a, b)
            case "*":
                return self.engine.multiply(a, b)
            case "/":
                return self.engine.divide(a, b)
            case _:
                error_msg = "Була задана некоректна дія."
                print(error_msg)
```

calc.py

```
from operation import Operator

def main():
    app = Operator()

    try:
        num1 = float(input("Введіть перше число: "))
        num2 = float(input("Введіть друге число: "))
        sign = input("Введіть дію (+ - * /): ")

        result = app.perform_calculation(num1, num2, sign)

        print(f"Відповідь: {result}")

    except ValueError as e:
        print(f"Виникла помилка: {e}")

if __name__ == "__main__":
    main()
```

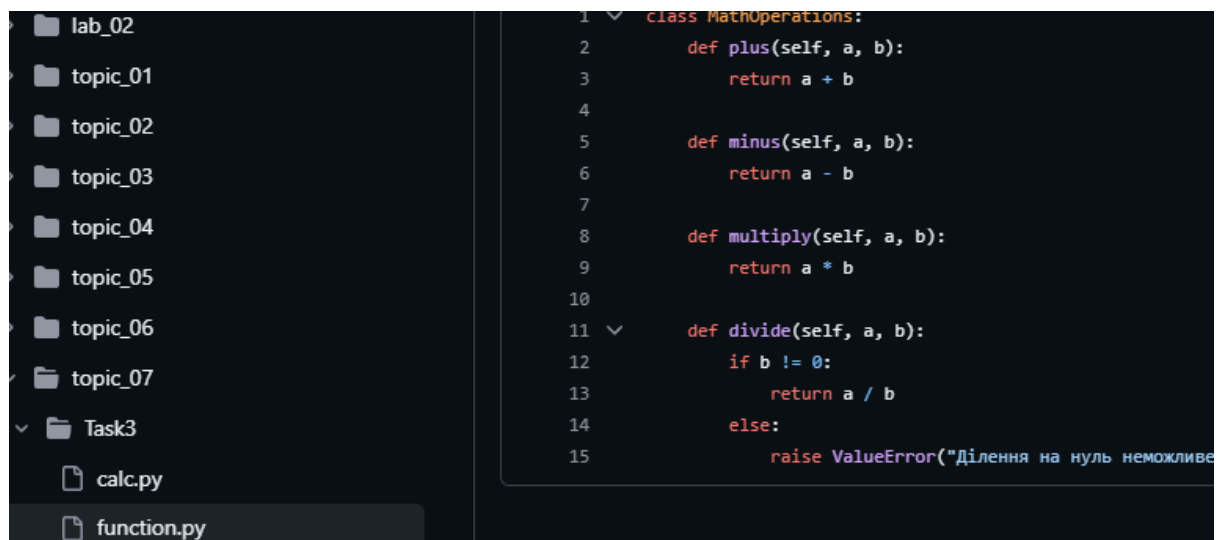


Рис.3 Скріншот третього завдання з GitHub