**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. Рядок, що має вигляд "abcdefg123" перетворити наступним чином "321gfedcba.

2. Тестування таких властивостей як: strip, capitalize, title, upper, lower.

3. Виконати пошук дискримінанту.

**Хід виконання завдання:**

**1.** Задав змінній значення (abcdefg123) та допомогою команди str[::-1] отримав зворотній рядок та вивів результат на екран.

**2.1**  Задав змінній значення (.lock.) та за допомогою функції strip(".") видалив всі крапки, які знаходяться на початку і в кінці рядка.

**2.2** Задав змінній значення (lock, local) та за допомогою функції capitalize() вивів перше слово з великої букви на екран.

**2.3** Задав змінній значення (lock, local) та за допомогою функції title() вивів всі слова з великої букви.

**2.4** Задав змінній значення (lock) та за допомогою функції upper() вивів слово великими буквами.

**2.5** Задав змінній значення (LOCK) та за допомогою функції lower() вивів слово маленькими літерами.

**3.** Задаю назву функції. Функція дискримінант з трьома параметрами також прописую формулу для D.

Повертаю значення D за допомогою команди return та задаю значення параметрам через консоль.

Потім повертаю назад D яке дорівнює дискримінанту. Result присвоюю discriminant та вивожу.

**Текст програми:**

**1.** str = ("abcdefg123")

print (str[::-1])

**2.** str = ".lock." # Видалення символів '.' з обох кінців рядка.

str = str.strip(".")

print(str)

str = "lock, local"

str = str.capitalize() # Перше слово з великої літери

print(str)

str = "lock, local"

str = str.title() # Всі слова з великої літери

print(str)

str = "lock"

str = str.upper() # Всі слова великими літерами

print(str)

str = "LOCK"

str = str.lower() # Всі слова маленькими літерами

print(str)

**3.**

def discriminant(a,b,c):

D = b\*b-4\*a\*c

return D

a = int(input("Please enter coef a : "))

b = int(input("Please enter coef b : "))

c = int(input("Please enter coef c : "))

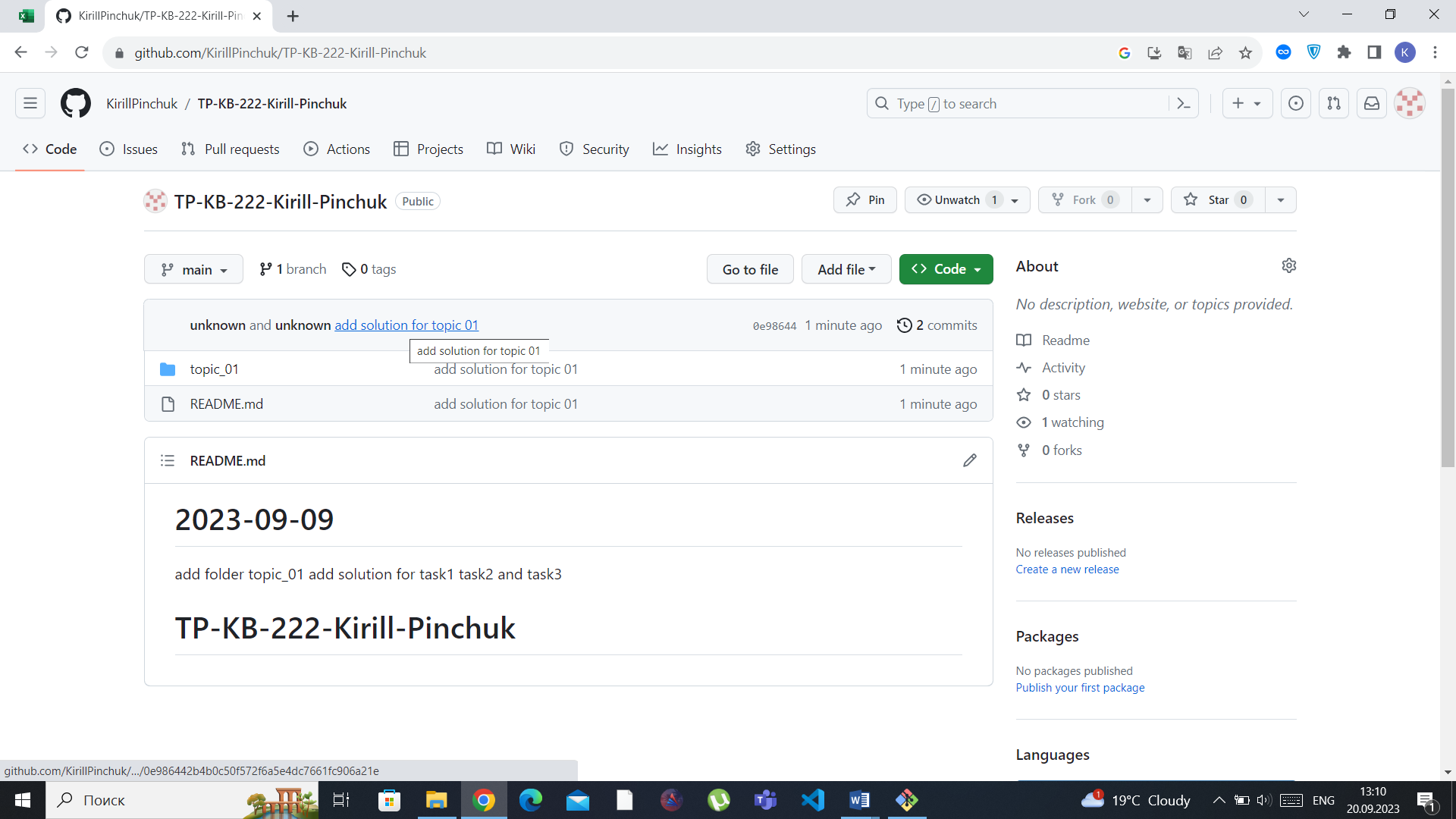
D = discriminant(a,b,c)

result = discriminant(a,b,c)

print(result)

**Посилання на github:** https://github.com/KirillPinchuk

**Знімок екрану з посилання на github:**



**Звіт до Теми №2**

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

1. Написати функцію для квадратного рівняння з використанням умови для D: D < 0, D == 0, D > 0.

2. Написати програму калькулятор з використанням if, elif.

3. Написати програму калькулятор з використаням match.

**Хід виконання завдання:**

1. Ввожу значення a, b, c за допомогою imput. Функція discr(a, b, c) обчислює дискримінант (D) за формулою D = b\*\*2 – 4\*a\*c та повертає його значення. Потім перевіряю D > 0, D<0 або D == 0 та вивожу результат від залежності значення дискримінанту. Також добавлена функція обчислення коренів при D > 0, D<0 або D == 0.
2. За допомогою **input**, **op** задаю значення a, b та операцію яку треба виконати. Додаю чотири функції для виконання математичних операцій: sum, minus, multiplication та division. В залежності від введеної операції, програма виконує відповідну функцію та зберігає результат в змінну result. Умовний оператор перевіряє введену операцію та виконує відповідну гілку коду. Якщо введена операція не відповідає жодній з чотирьох допустимих операцій ("+", "-", "\*", "/"), то програма виводить "Invalid operation".Результат обчислення або повідомлення про помилку виводяться на екран.
3. За допомогою **input**, **op** задаю значення a, b та операцію яку треба виконати.. Використовуючи match програма перевіряє значення op і визначає, яку операцію виконувати. У відповідності до значення op програма виконує одну з наступних дій: sum, minus, multiplication та division. Якщо op дорівнює "/", то програма перевіряє, чи b не дорівнює нулю. Якщо b не дорівнює нулю, то виконується функція division(a, b), і результат (частка a на b) виводиться на екран. В іншому випадку, якщо b дорівнює нулю, виводиться повідомлення "Invalid operation".

**Текст програми:**

**1**. a = int(input("Enter a: "))

b = int(input("Enter b: "))

c = int(input("Enter c: "))

def discr(a, b, c):

D = b\*\*2 - 4 \* a \* c

return D

D = discr(a, b, c)

if D > 0:

result = "D more 0"

print(D)

elif D == 0:

result = "D equal 0 "

else:

result = "No solution"

print(result)

**2.** a = int(input("Enter a: "))

b = int(input("Enter b: "))

op = input("Operation: ")

def sum(a, b):

return a + b

def minus(a,b):

return a - b

def multiplication(a,b):

return a \* b

def division(a, b):

if b != 0:

return a / b

else:

print("Division by zero is not allowed")

if op == "+":

result = sum(a,b)

print (result)

elif op == "-":

result = minus(a,b)

print (result)

elif op == "\*":

result = multiplication(a,b)

print (result)

elif op == "/":

result = division(a,b!=0)

print (result)

else:

print("Invalid operation")

**3.** a = int(input("Enter a: "))

b = int(input("Enter b: "))

op = (input("Operation: "))

match op:

case "+":

def sum(a, b):

return a + b

print (a + b)

case "-":

def minus(a,b):

return a - b

print (a -b)

case "\*":

def multiplication(a,b):

return a \* b

print (a \* b)

case "/":

if b != 0:

def division(a, b):

return a / b

result = division(a, b)

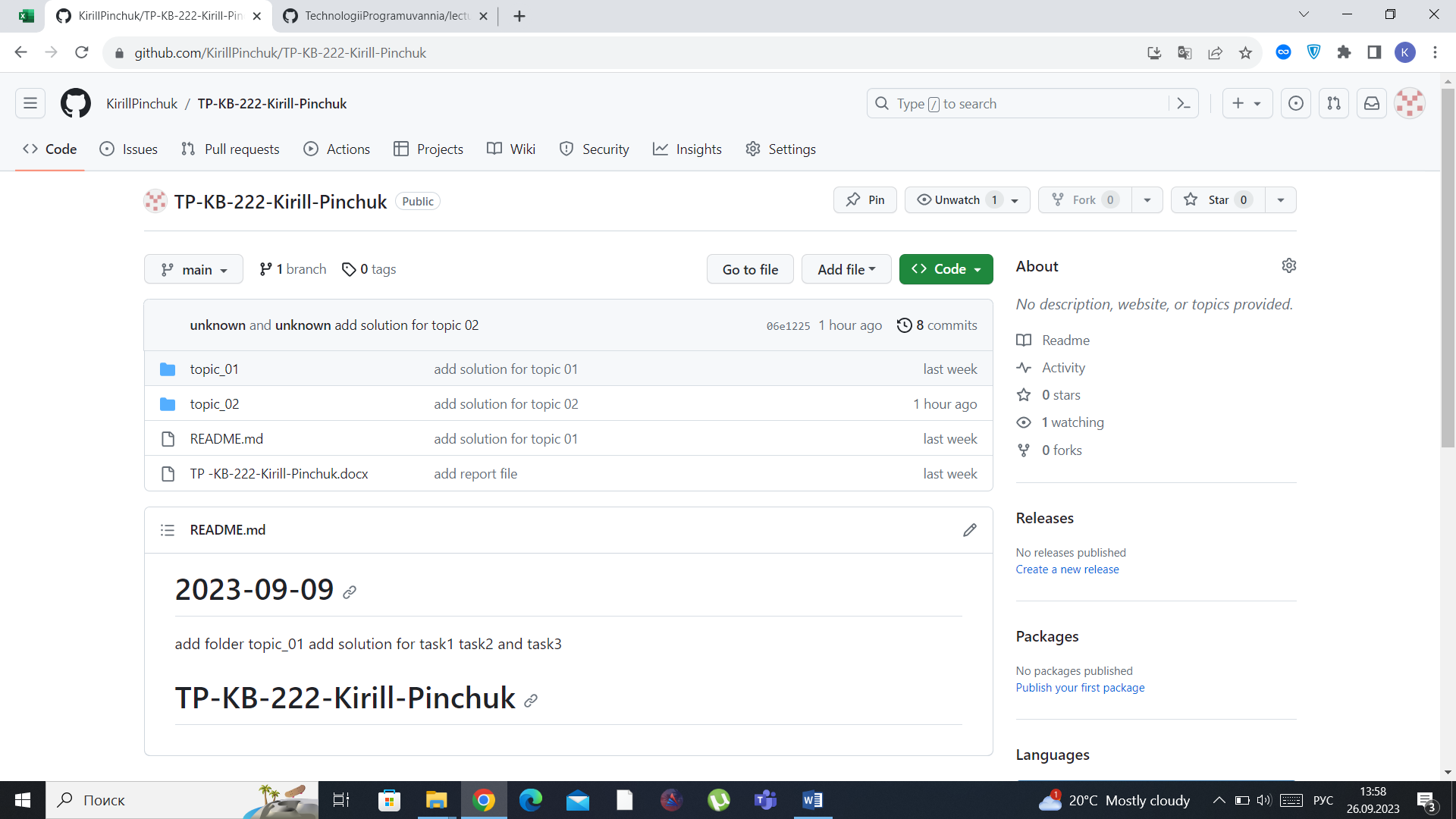
print(result)

else:

print("Invalid operation")

Посилання на github: **https://github.com/KirillPinchuk**

Знімок екрану з посилання на github:



**Звіт до Теми №3**

Функції та змінні

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()
3. Написати програму тестування функцій словників таких як: update(), del(), clear(), keys(), values(), items()
4. Маючи відсортований список, написати функцію пошуку позиції для вставки нового елементу в список.

**Хід виконання завдання:**

**1.** Програма запускається у циклі while True, що дозволяє безкінечно повторювати запити, поки користувач не введе правильну операцію. Визначаються функції sum, minus, multiplication, та division, які виконують відповідні математичні операції за допомогою перевірки "OP". Якщо операція введена неправильно (не "+", "-", "\*", або "/"), програма виводить повідомлення про неправильну операцію і повторює запит. Цикл while продовжується, поки користувач не введе правильну операцію. Після виведення результату програма завершує роботу.

**2.** Використання extend для додавання рядка до списку:

Починаю зі списку list, який має значення ["Emma", "Jon", "Bob"].

Використовуєте метод extend, щоб додати рядок "Liza" до списку.

Результат виводу у список ["Emma", "Jon", "Bob", "L", "i", "z", "a"].

Використання append для додавання елементу до списку:

Список list, який має значення ["Emma", "Jon", "Bob"].

Використовуєте метод append, щоб додати рядок "Liza" до списку.

Результат виводу у список ["Emma", "Jon", "Bob", "Liza"].

Використання insert для вставки елемента до списку за певним індексом:

Список list зі значеннями ["Emma", "Jon", "Bob"].

Застосовуєте метод insert, щоб вставити рядок "Liza" на позицію з індексом 1

Результат виводу у список ["Emma", "Liza", "Jon", "Bob"].

Використання remove для видалення конкретного елемента зі списку:

Список list зі значеннями ["Emma", "Jon", "Bob"].

Використовуєте метод remove, щоб видалити рядок "Jon" зі списку.

Результат виводу у список ["Emma", "Bob"].

Використання clear для очищення списку:

Список list зі значеннями ["Emma", "Jon", "Bob"].

Використовуєте метод clear, щоб очистити список.

Результат виводу у порожній список [].

Використання sort для сортування списку:

Список list зі значеннями ["Emma", "Jon", "Bob"].

Використовуєте метод sort, щоб відсортувати список в алфавітному порядку.

Результат виводу у список ["Bob", "Emma", "Jon"].

Використання reverse для обертання списку:

Список list зі значеннями ["Emma", "Jon", "Bob"].

Використовуєте метод reverse, щоб обернути порядок елементів у списку.

Результат виводу у список ["Bob", "Jon", "Emma"].

Список list зі значеннями ["Emma", "Jon", "Bob"].

Використовуєте метод copy, щоб створити копію списку, яка має ті ж самі значення.

Результат виводу копії списку ["Emma", "Jon", "Bob"].

**3.** Вивожу початковий словник dict з ключами "name", "age", "city" і відповідними значеннями.

Вивожу оригінальний словник, використовуючи print("Original =", dict).

Додавання нового ключа та значення до словника:

Використовую метод update, щоб додати новий ключ "country" і його значення "USA" до словника.

Виводите словник після оновлення, використовуючи print(dict).

Видалення ключа та відповідного значення зі словника:

Використовую оператор del, щоб видалити ключ "age" та відповідне значення зі словника.

Виводите словник після видалення, використовуючи print(dict).

Отримання списку ключів словника:

Використовую метод keys(), щоб отримати список ключів словника.

Виводите список ключів, використовуючи print(keys).

Отримання списку значень словника:

Використовую метод values(), щоб отримати список значень словника.

Виводите список значень, використовуючи print(values).

Отримання списку пар ключ-значення словника:

Використовую метод items(), щоб отримати список кортежів, кожний з яких містить пару ключ-значення словника.

Виводите список пар, використовуючи print(items).

Очищення словника:

Використовую метод clear(), щоб видалити всі ключі та значення зі словника.

Виводите словник після очищення, використовуючи print(dict).

Це послідовність дій для кожного з операцій, які ви виконуєте зі словником dict.

**4.** Спочатку, ми маємо функцію InsterToPosition, яка приймає відсортований список sorted\_list та новий елемент NewValue. У функції ви створюєте змінну position і встановлюєте її значення на 0. Запускаю цикл for, який проходиться по елементах у відсортованому списку sorted\_list.

У циклі перевіряю, чи NewValue менше за поточний елемент i в sorted\_list. Якщо це так, то ви збільшуєте значення position на 1. Після завершення циклу повертаю значення position, яке вказує на позицію, на яку слід вставити NewValue в список sorted\_list.

Викликаю функцію InsterToPosition, передаючи їй відсортований список list і NewValue, і зберігаєте результат у змінній insert\_position. Встановлюю NewValue в список list на позицію insert\_position за допомогою методу insert().

Вивожу позицію вставки (insert\_position) і оновлений список list за допомогою функції print().

**Текст програми:**

1. while True:

a = int(input("Enter a: "))

b = int(input("Enter b: "))

op = input("Operation: ")

def sum(a, b):

return a + b

def minus(a,b):

return a - b

def multiplication(a,b):

return a \* b

def division(a, b):

if b != 0:

return a / b

else:

print("Division by zero is not allowed")

if op == "+":

result = sum(a,b)

break

elif op == "-":

result = minus(a,b)

break

elif op == "\*":

result = multiplication(a,b)

break

elif op == "/":

result = division(a,b!=0)

break

print("result =", result)

**2.** list = ["Emma", "Jon", "Bob"]

list.extend("Liza")

print(list)

list = ["Emma", "Jon", "Bob"]

list.append("Liza")

print(list)

list = ["Emma", "Jon", "Bob"]

list.insert(1, "Liza")

print(list)

list = ["Emma", "Jon", "Bob"]

list.remove("Jon")

print(list)

list = ["Emma", "Jon", "Bob"]

list.clear()

print(list)

list = ["Emma", "Jon", "Bob"]

list.sort()

print(list)

list = ["Emma", "Jon", "Bob"]

list.reverse()

print(list)

list = ["Emma", "Jon", "Bob"]

list.copy()

print(list)

**3**. dict = {"name": "John", "age": 30, "city": "New York"}

print("Original =", str)

dict.update({"country": "USA"})

print(dict)

del dict ["age"]

print(dict)

keys = dict.keys()

print(keys)

values = dict.values()

print(values)

items = dict.items()

print(items)

dict.clear()

print(dict)

**4.** def InsterToPosition(sorted\_list, NewValue):

position = 0

for i in sorted\_list:

if NewValue < i:

position +=1

return position

list = ["aa", "ee", "zz"]

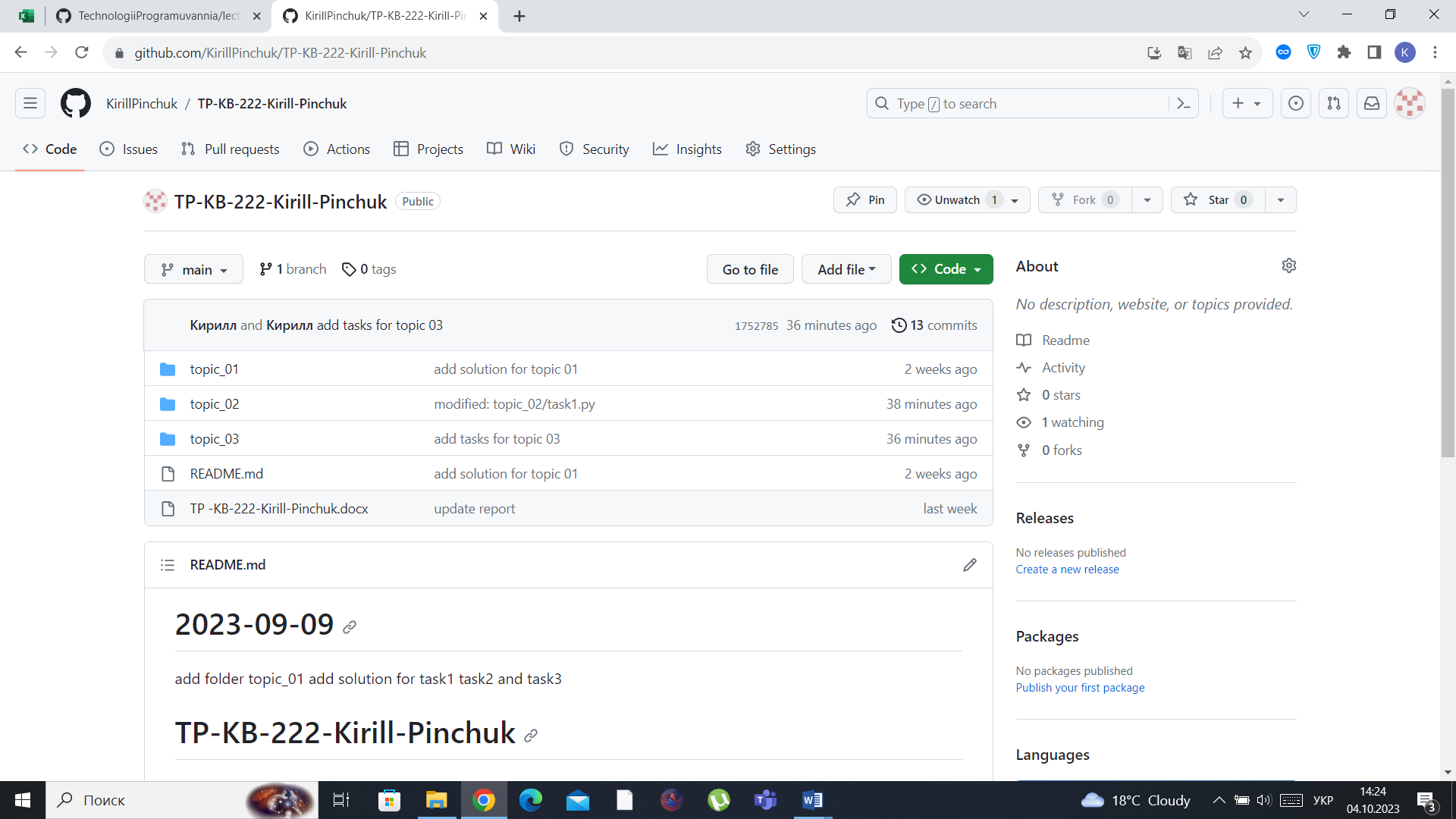
NewValue = input("New value: ")

insert\_position = InsterToPosition(list, NewValue)

list.insert(insert\_position, NewValue)

print(f"position: {insert\_position}, list: {list}")

Посилання на github: <https://github.com/KirillPinchuk>

Знімок екрану з посилання на github: 

**Звіт до Теми №4**

Функції та змінні

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

**Хід виконання завдання:**

Включаю функцію sum(a, b), яка приймає два аргументи a і b і повертає їхню суму, minus(a, b), яка приймає два аргументи a і b і повертає різницю між ними,

multiplication(a, b), яка приймає два аргументи a і b і повертає їхній добуток ,

division(a, b), яка приймає два аргументи a і b і виконує ділення a на b. Якщо b дорівнює нулю, вона обробляє виняток ZeroDivisionError і виводить повідомлення про те, що ділення на нуль не допускається.

Включаю функція Value(prompt), яка запитує користувача ввести ціле число, виводить повідомлення prompt, та повертає введене число. Вона обробляє виняток ValueError, якщо користувач введе нечисловий рядок, і виводить повідомлення про помилку.

Починається безкінечний цикл while True, який питає користувача, чи він бажає вийти з програми. Якщо користувач введе "yes", програма виводить повідомлення "Quitting the program" і завершує виконання.

Якщо користувач введе "no" або інше значення, програма продовжує виконання і питає користувача ввести два цілих числа a і b.

Після отримання чисел a і b, програма запитує користувача ввести операцію op (+, -, \*, /).

Перевіряється, чи op є однією з допустимих операцій (+, -, \*, /). Якщо ні, програма виводить повідомлення про помилку та повертається до початку циклу для нового вводу операції.

Якщо op є допустимою операцією, програма викликає відповідну функцію (sum, minus, multiplication або division) з введеними a і b та виводить результат.

Після виведення результату програма повертається до початку циклу і запитує користувача, чи він бажає вийти з програми або продовжити виконання операцій.

**Текст програми:**

def sum(a, b):

return a + b

def minus(a, b):

return a - b

def multiplication(a, b):

return a \* b

def division(a, b):

try:

return a / b

except ZeroDivisionError:

print("Division by zero is not allowed")

def Value(prompt):

while True:

try:

return int(input(prompt))

except ValueError:

print("Invalid input. Please enter a valid integer")

while True:

exit = input("Do you want to exit the program? (yes/no): ")

if exit.lower() == "yes":

print("Quitting the program")

break

a = Value("Enter a: ")

b = Value("Enter b: ")

op = input("Operation (+, -, \*, /)")

if op not in ("+", "-", "\*", "/"):

print("Invalid operation. Please +, -, \*, /")

continue

if op == "+":

result = sum(a, b)

elif op == "-":

result = minus(a, b)

elif op == "\*":

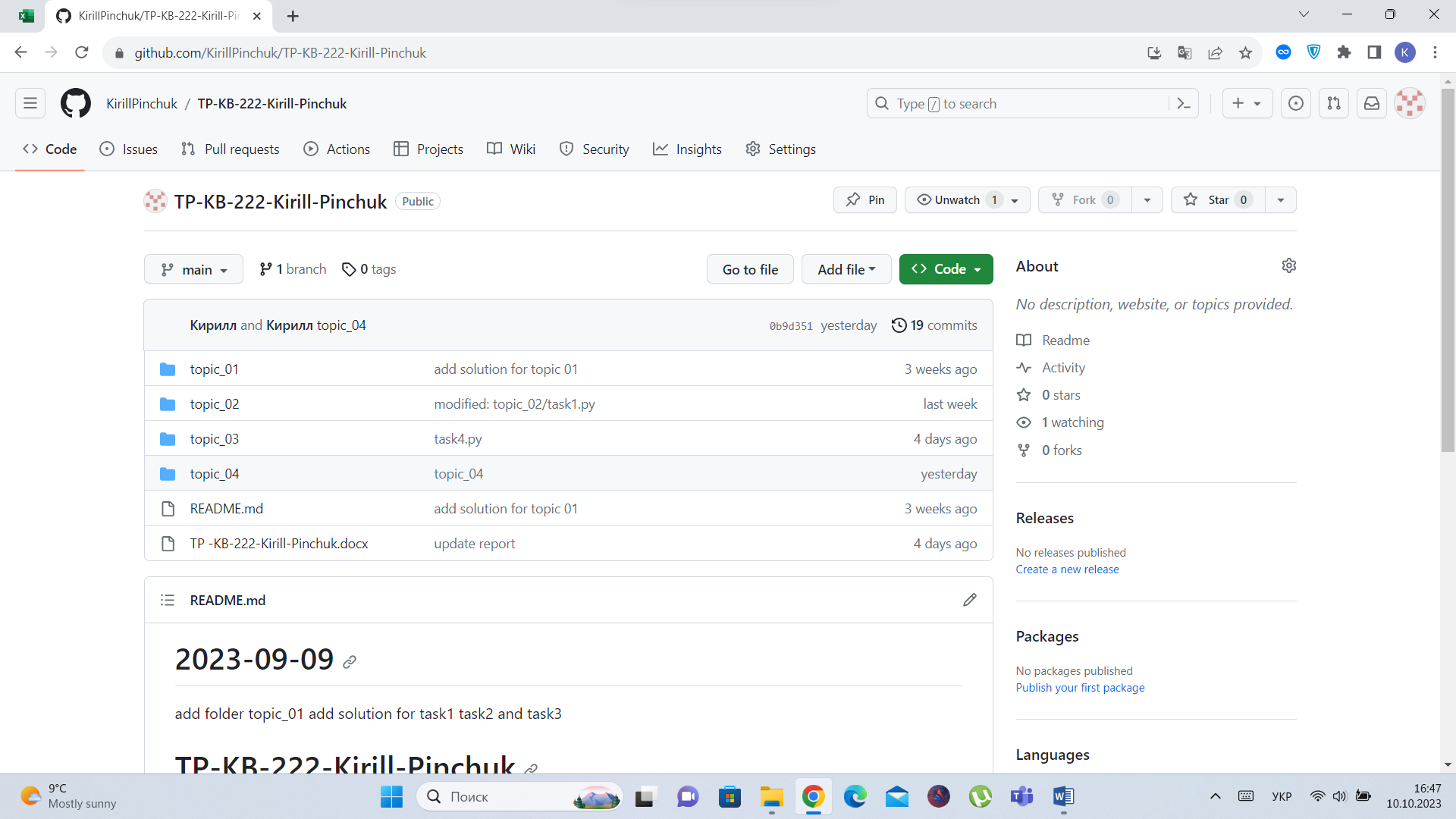
result = multiplication(a, b)

elif op == "/":

result = division(a, b)

print("Result =", result)

Посилання на github: **https://github.com/KirillPinchuk**

Знімок екрану з посилання на github: ****

**Звіт до Теми №5**

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

1.Гра камінь, ножиці, папір.

2.Конвентатор валют

3.Модули для програми калькулятор

**Хід виконання завдання:**

1. Використовую імпорт функції choice з модуля random. Ця функція використовується для вибору випадкового елемента зі списку. Створюю список choise, який містить можливі варіанти: "rock", "scissors", "paper". Користувачу пропонується ввести свій вибір, використовуючи функцію input. Введений вибір зберігається у змінній my\_choice.

Комп'ютер вибирає випадковий елемент зі списку choise за допомогою функції choice. Результат зберігається у змінній comp\_choice.

Вибір користувача і вибір комп'ютера виводяться на екран за допомогою функції print.

Можливі варіанти:

Якщо вибір користувача і вибір комп'ютера однакові, виводиться "It's a draw".

Якщо вибір користувача - "rock", а вибір комп'ютера - "scissors", то виводиться "You Win".

Якщо вибір користувача - "scissors", а вибір комп'ютера - "paper", то також виводиться "You Win".

Якщо вибір користувача - "paper", а вибір комп'ютера - "rock", знову ж таки, виводиться "You Win".

У всіх інших випадках виводиться "Lose".

**2.** Імпортую бібліотеку requests для взаємодії з веб-сайтом.

Виконую запит за допомогою requests.get до URL і зберігаєте результат у змінну url.

За допомогою методу json() отримую дані у форматі JSON з цього URL і зберігаєте їх у змінну source.

Створюю порожній словник currency\_rates, в якому зберігатиму курси обміну для підтримуваних валют.

За допомогою циклу for прохожу по кожному об'єкту в source і додаю курс обміну до словника currency\_rates з ключем як символьний код валюти (rate['cc']) та відповідним значенням (rate['rate']).

Створюю список supported\_currencies, що містить підтримувані валюти, які користувач може вибрати.

Запускаю цикл while, який триватиме доти, доки користувач не введе підтримувану валюту. Використовую input для отримання валютного коду від користувача та перевожу його в верхній регістр за допомогою .upper().

Перевіряю, чи введена валюта входить до списку supported\_currencies. Якщо ні, виводите повідомлення про те, що ця валюта не підтримується, і цикл продовжується. Якщо так, цикл завершується і користувач продовжує введення. Користувач вводить суму валюти, яку він бажає конвертувати, за допомогою input. Сума зберігається у змінній amount.

Обчислює конвертовану суму, помножуючи введену суму (amount) на курс обміну для вибраної валюти (currency\_rates[currency]) і зберігаєте результат у змінну converted\_amount.

За допомогою print виводите результат на екран, показуючи введену суму та обрану валюту, а також конвертовану суму в гривнях.

3. Імпортую бібліотеку requests для взаємодії з веб-сайтом.

Виконую запит за допомогою requests.get до URL і зберігаєте результат у змінну url.

За допомогою методу json() отримую дані у форматі JSON з цього URL і зберігаєте їх у змінну source.

Створюю порожній словник currency\_rates, в якому зберігатиму курси обміну для підтримуваних валют.

За допомогою циклу for прохожу по кожному об'єкту в source і додаю курс обміну до словника currency\_rates з ключем як символьний код валюти (rate['cc']) та відповідним значенням (rate['rate']).

Створюю список supported\_currencies, що містить підтримувані валюти, які користувач може вибрати.

Запускаю цикл while, який триватиме доти, доки користувач не введе підтримувану валюту. Використовую input для отримання валютного коду від користувача та перевожу його в верхній регістр за допомогою .upper().

Перевіряю, чи введена валюта входить до списку supported\_currencies. Якщо ні, виводите повідомлення про те, що ця валюта не підтримується, і цикл продовжується. Якщо так, цикл завершується і користувач продовжує введення. Користувач вводить суму валюти, яку він бажає конвертувати, за допомогою input. Сума зберігається у змінній amount.

Обчислює конвертовану суму, помножуючи введену суму (amount) на курс обміну для вибраної валюти (currency\_rates[currency]) і зберігаєте результат у змінну converted\_amount.

За допомогою print виводите результат на екран, показуючи введену суму та обрану валюту, а також конвертовану суму в гривнях.

Текст програми:

1. from random import choice

choise = ["rock", "scissor", "paper"]

my\_choice = input("Select: rock, scissors, paper:")

comp\_choice = choice(choise)

print(f"You: {my\_choice}")

print(f"Computer: {comp\_choice}")

if my\_choice == comp\_choice:

    print("It's a draw")

elif my\_choice == "rock" and comp\_choice == "scissors":

    print("You Win")

elif my\_choice == "scissors" and comp\_choice == "paper":

    print("You Win")

elif my\_choice =="paper" and comp\_choice == "rock":

    print("You Win")

else:

    print("Lose")

2. import requests

url = requests.get("https://bank.gov.ua/NBUStatService/v1/statdirectory/exchange?json")

source = url.json()

currency\_rates = {}

for rate in source:

    currency\_rates[rate['cc']] = rate['rate']

supported\_currencies = ["EUR", "USD", "PLN"]

while True:

    currency = input("Enter the currency (EUR, USD, PLN): ").upper()

    if currency not in supported\_currencies:

        print("This currency is not supported.")

    else:

        break

amount = float(input("Enter the amount of currency: "))

converted\_amount = amount \* currency\_rates[currency]

print(f"{amount} {currency} = {converted\_amount} ₴")

3.

3.1 calc.py

import functions

import operations

while True:

    a = operations.Value("Enter a: ")

    b = operations.Value("Enter b: ")

    op = operations.get\_operation()

    if op == "+":

        result = functions.sum(a, b)

    elif op == "-":

        result = functions.minus(a, b)

    elif op == "\*":

        result = functions.multiplication(a, b)

    elif op == "/":

        result = functions.division(a, b)

    print("Result =", result)

    exit = input("Do you want to exit the program? (yes/no): ")

    if exit.lower() == "yes":

        print("Quitting the program")

        break

3.2 Operations.py

def Value(prompt):

    while True:

        try:

            return int(input(prompt))

        except ValueError:

            print("Invalid input. Please enter a valid integer")

def get\_operation():

    while True:

        op = input("Operation (+, -, \*, /): ")

        if op in ("+", "-", "\*", "/"):

            return op

        else:

            print("Invalid operation. Please  +, -, \*, /")

3.3 Functions.py

def sum(a, b):

    return a + b

def minus(a, b):

    return a - b

def multiplication(a, b):

    return a \* b

def division(a, b):

    try:

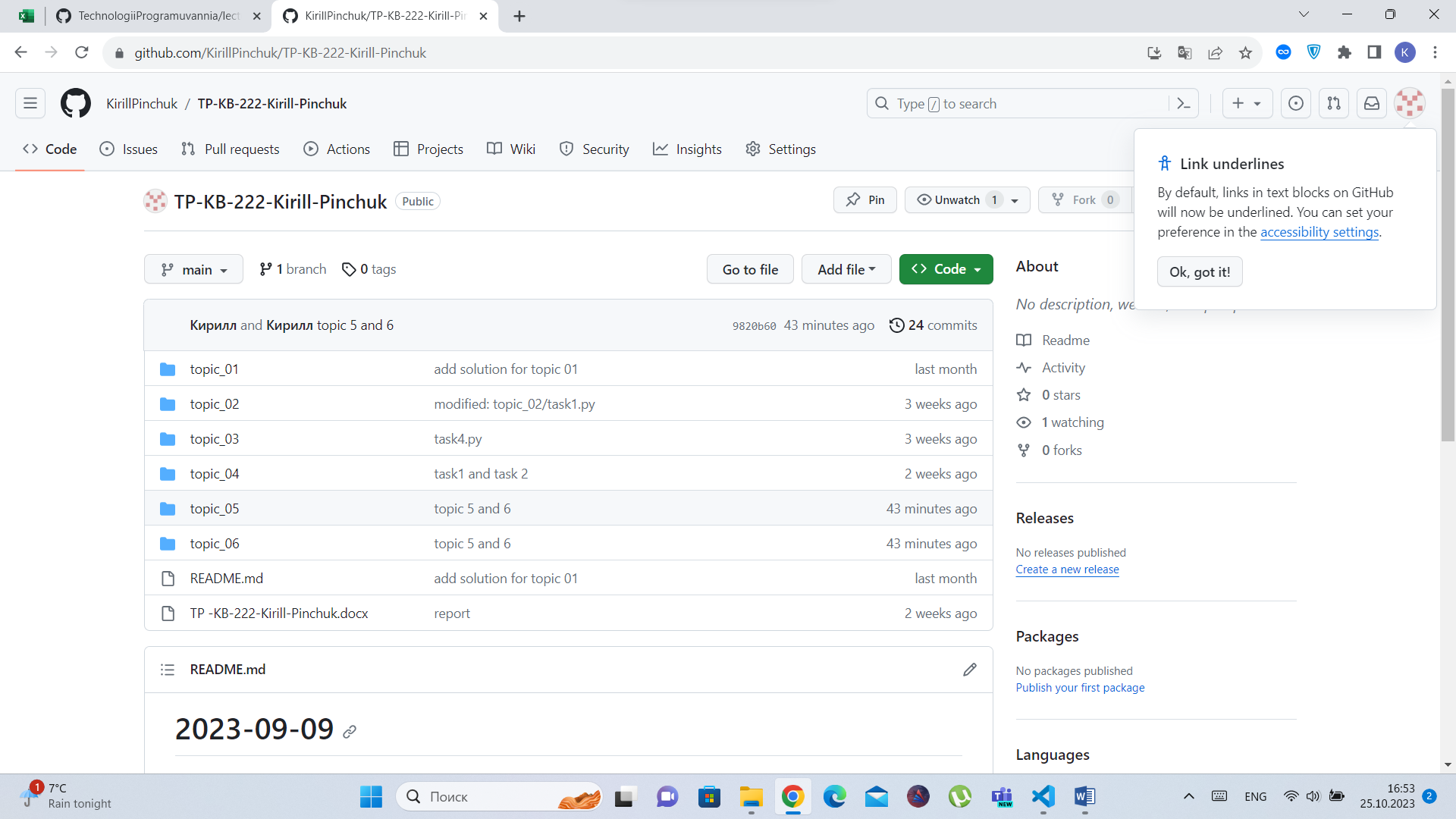
        return a / b

    except ZeroDivisionError:

        print("Division by zero is not allowed")

Посилання на github: <https://github.com/KirillPinchuk>

Знімок екрану з посилання на github:



**Звіт до Теми №6**

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

1. Використання lambda функцій для функції сортування
2. Логування всіх дій в застосунку Калькулятор

**Хід виконання завдання:**

**1.** Спочатку список student\_list, де кожен елемент - це словник, що містить ім'я студента і його оцінку.

Створюю новий список sorted\_grade, використовуючи функцію sorted. Вказую key, який визначає, за яким критерієм сортувати, у цьому випадку - за оцінками. reverse = True означає, що ви сортуєте в зворотному порядку (від найвищої до найнижчої оцінки).

Вивожу відсортований список sorted\_grade, виводячи ім'я і оцінку кожного студента.

Потім створтую ще один відсортований список sorted\_Name, використовуючи той самий підхід, але замість "grade" використовую "Name" як ключ для сортування.

Вивожу список sorted\_Name, виводячи ім'я та оцінку кожного студента за алфавітом.

**2.** Імпортуєте функції з модулів functions та operations, а також модуль logging для обробки журналу подій.

Налаштовую логування за допомогою logging.basicConfig, вказуючи ім'я файлу (calc.log), рівень логування (INFO) і формат виводу.

Запускається безкінечний цикл while True, що дозволяє користувачеві виконувати калькулятор декілька разів.

Запитую користувача ввести два числа a і b, використовуючи operations.Value() для введення кожного числа. Потім отримую oр за допомогою operations.get\_operation().

В залежності від вибраного оператора (op), ви викликаєте відповідну функцію із модуля functions для виконання відповідної операції (+, -, \*, /).

Обчислення записується у змінну result. Результат виводиться на екран разом із повідомленням "Result = " і записується у журнал подій за допомогою logging.info().

Запитую користувача, чи він бажає вийти з програми, і якщо відповідь "yes", то ви виходите з циклу while, завершуючи програму.

**Текст програми:**

**1.** student\_list = [

    {"Name": "Anna", "grade": 85},

    {"Name": "Petro", "grade": 82},

    {"Name": "Maria", "grade": 48},

    {"Name": "John", "grade": 66},

    {"Name": "Oleg", "grade": 95}

]

sorted\_grade = sorted(student\_list, key = lambda x: x["grade"], reverse = True)

print("Sorting grade:")

for student in sorted\_grade:

    print(f"Name: {student['Name']}, Grade: {student['grade']}")

sorted\_Name = sorted(student\_list, key = lambda x: x["Name"])

print("\nSorting name:")

for student in sorted\_Name:

    print(f"Name: {student['Name']}, Grade: {student['grade']}")

**2.**

2.1 calc.py

import functions

import operations

import logging

logging.basicConfig(filename='calc.log', level=logging.INFO, format='%(asctime)s - %(message)s')

while True:

    a = operations.Value("Enter a: ")

    b = operations.Value("Enter b: ")

    op = operations.get\_operation()

    if op == "+":

        result = functions.sum(a, b)

    elif op == "-":

        result = functions.minus(a, b)

    elif op == "\*":

        result = functions.multiplication(a, b)

    elif op == "/":

        result = functions.division(a, b)

    print("Result =", result)

    logging.info(f"Operation: {a} {op} {b} = {result}")

    exit = input("Do you want to exit the program? (yes/no): ")

    if exit.lower() == "yes":

        print("Quitting the program")

        break

* 1. Operations.py

def Value(prompt):

    while True:

        try:

            return int(input(prompt))

        except ValueError:

            print("Invalid input. Please enter a valid integer")

def get\_operation():

    while True:

        op = input("Operation (+, -, \*, /): ")

        if op in ("+", "-", "\*", "/"):

            return op

        else:

            print("Invalid operation. Please  +, -, \*, /")

* 1. Functions.py

def sum(a, b):

    return a + b

def minus(a, b):

    return a - b

def multiplication(a, b):

    return a \* b

def division(a, b):

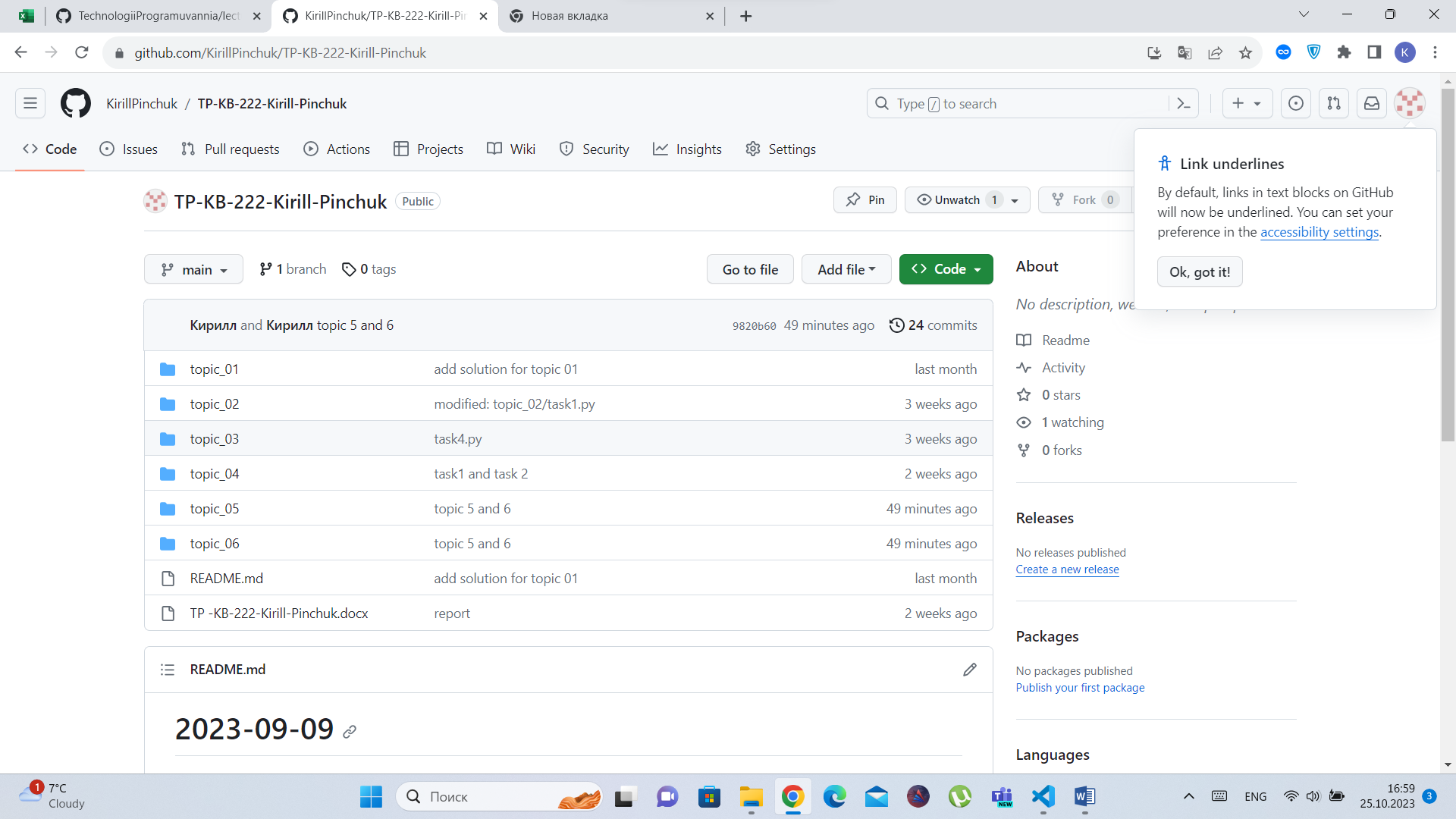
    try:

        return a / b

    except ZeroDivisionError:

        print("Division by zero is not allowed")

Посилання на github: **https://github.com/KirillPinchuk**

Знімок екрану з посилання на github: 

***Звіт до Теми №7***

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

1. Ознайомитись з існуючими за замовченням методами класу по типу \_\_init\_\_(self) \_\_str\_\_(self)\_\_ та надати приклади використання.
2. 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.
3. Використовуючи принципи ООП переписати програму Калькулятор. Завдання має бути виконано використовуючи модульний підхід.

**Хід виконання завдання:**

**1.** Імпортую модуль math, щоб мати доступ до значення pi (число Пі) для обчислення об'єму циліндра.

Створюю клас Cylinder і визначте його конструктор, який ініціалізує об'єкт з параметрами r (радіус) і h (висота). Значення за замовчуванням - 4 для радіусу і 10 для висоти.

Додаю метод volume, який обчислює об'єм циліндра за допомогою формули V = π \* r^2 \* h. У цьому методі використовуйте значення pi з модуля math. Перевизначте метод \_\_str\_\_, щоб об'єкт класу Cylinder можна було вивести як рядок. У цьому методі повернемо об'єм циліндра у вигляді рядка.

Створюємо об'єкт cylinder, передаючи значення радіусу і висоти (4 і 10) в конструктор класу Cylinder:

Вивожу об'єкт cylinder за допомогою функції print. В результаті отримаю об'єм циліндра на екран.

2. Створюю клас Student і визначаю його конструктор, який приймає два параметри: name (ім'я студента) і age (вік студента). Кожний студент має ім'я та вік.

Створюю список student\_list, де кожний елемент буде об'єктом класу Student, і заповнюю його студентами з відповідними іменами та віками.

Відсортую список student\_list за віком студентів за допомогою функції sorted та функцію для сортування за віком.

Прохожусь по відсортованому списку sorted\_student\_list за допомогою циклу for і вивовивожу список відсортований за віком.

**3.** Імпортую модуль logging, який використовується для логування подій в програмі.

Визначаю клас Calculator, який включає методи для виконання арифметичних операцій (додавання, віднімання, множення, ділення) та методи для взаємодії з користувачем.

Конструктор класу ініціалізує логер для реєстрації подій програми.

Методи sum, minus, multiplication та division виконують відповідні арифметичні операції над числами та перевіряють на ділення на нуль.

Метод run містить головний цикл програми, в якому користувачу запропоновано ввести числа та операцію. Результат обчислень виводиться на екран та логується у файлі calc.log. Користувачу також надається можливість завершити програму.

Методи get\_value та get\_operation використовуються для отримання чисел та операції від користувача. Якщо введені дані не відповідають вимогам, програма виведе повідомлення про помилку та знову запитає введені дані.

В методі \_\_main\_\_, налаштовується логування за допомогою logging.basicConfig, задаються параметри логування, такі як рівень логування (INFO) та форматування логів. Потім створюється екземпляр класу Calculator та викликається його метод run для запуску програми.

**Текст програми:**

**1.** import math

class Cylinder:

    def \_\_init\_\_(self, r=4, h=10):

        self.radius = r

        self.height = h

    def volume(self):

        return self.radius \* self.height \* math.pi

    def \_\_str\_\_(self):

        return str(self.volume())

cylinder = Cylinder(4, 10)

print(cylinder)

**2.** class Student:

    def \_\_init\_\_(self, name, age):

        self.name = name

        self.age = age

student\_list = [

    Student("Anna", 22),

    Student("Petro", 18),

    Student("Maria", 26),

    Student("John", 36),

    Student("Oleg", 21)

]

sorted\_student\_list = sorted(student\_list, key = lambda student: student.age)

for student in sorted\_student\_list:

    print(f"Name: {student.name} , age: {student.age}")

**3.** import logging

class Calculator:

    def \_\_init\_\_(self):

        self.logger = logging.getLogger(\_\_name\_\_)

    def sum(self, a, b):

        return a + b

    def minus(self, a, b):

        return a - b

    def multiplication(self, a, b):

        return a \* b

    def division(self, a, b):

        if b == 0:

            raise ZeroDivisionError("Division by zero is not allowed")

        return a / b

    def run(self):

        while True:

            a = self.get\_value("Enter a: ")

            b = self.get\_value("Enter b: ")

            op = self.get\_operation()

            if op == "+":

                result = self.sum(a, b)

            elif op == "-":

                result = self.minus(a, b)

            elif op == "\*":

                result = self.multiplication(a, b)

            elif op == "/":

                result = self.division(a, b)

            print("Result =", result)

            self.logger.info(f"Operation: {a} {op} {b} = {result}")

            exit\_program = input("Do you want to exit the program? (yes/no): ")

            if exit\_program.lower() == "yes":

                print("Quitting the program")

                break

    def get\_value(self, prompt):

        while True:

            try:

                return int(input(prompt))

            except ValueError:

                print("Invalid input. Please enter a valid integer")

    def get\_operation(self):

        while True:

            op = input("Operation (+, -, \*, /): ")

            if op in ("+", "-", "\*", "/"):

                return op

            else:

                print("Invalid operation. Please use +, -, \*, /")

if \_\_name\_\_ == "\_\_main\_\_":

    logging.basicConfig(filename='calc.log', level=logging.INFO, format='%(asctime)s - %(message)s')

    calculator = Calculator()

    calculator.run()

Посилання на github: <https://github.com/KirillPinchuk>

Знімок екрану з посилання на github:

