**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:

**Обернення рядка**

Хід виконання завдання:

1. Для початку я створив змінну text в яку можно вписати текст.
2. Добавив функцію виводу print (text), щоб побачити результат.
3. Потім за допомогою приставки [::-1] я перевернув рядок, і print видав його в зворотньому порядку

Код програми:

text = input("Введіть ваше слово: \n")

print(text[::-1])

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

Хід виконання завдання:

1. Створив змінну s якій надав текст «koledin oLexandr »
2. Вивів цю змінну через команду print додавши до неї (.strip()) для того щоб видалити пробіли з початку і кінця
3. Вивів цю змінну через команду print додавши до неї (.capitalize()) для того щоб зробити першу літеру великою а інші маленькими
4. Вивів цю змінну через команду print додавши до неї (.title()) для того щоб кожне слово починалось з великої літери
5. Вивів цю змінну через команду print додавши до неї (.upper()) для того щоб всі літери були великими
6. Вивів цю змінну через команду print додавши до неї (.lower()) для того щоб всі літери були маленькими

Код програми:

s=("koledin oLexandr  ")

print(s.strip())       # Видаляє пробіли з початку і кінця

print(s.capitalize())  # Перша літера велика, інші маленькі

print(s.title())       # Кожне слово з великої літери

print(s.upper())       # Всі літери великі

print(s.lower())       # Всі літери маленькі

**Написати функцію пошуку дискримінанту квадратного рівняння.**

Хід виконання завдання:

1. Створив функцію “calculate\_discriminant” з return який повертає дискримінант квадратного рівняння за формулою
2. Створив можливість вводити числа a, b, c
3. Створив змінну D, яка виводить наш дискримінант
4. Вивів данні через print

Код програми:

def calculate\_discriminant(a, b, c):

    return b\*\*2 - 4 \* a \* c

# Приклад використання

a = float(input("Введіть коефіцієнт a: "))

b = float(input("Введіть коефіцієнт b: "))

c = float(input("Введіть коефіцієнт c: "))

D = calculate\_discriminant(a, b, c)

print("Дискримінант: ", D)

посилання на Git:

<https://github.com/MinigamerTY/TP-KB-231-Koledin-Olexandr/tree/main/topic_01>

Звіт до Теми №2

Умовний перехід

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

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

Хід виконання завдання:

1. Створив функцію “calculate\_discriminant” з return який повертає дискримінант квадратного рівняння за формулою
2. Створив функцію “find\_roots” яка підставляє результат дискримінанту для обчислення коренів квадратного рівнняя за формулою
3. Додав перевірку значенню дискримінанта щоб дізнатися кількість коренів
4. Додав у функцію “find\_roots” можливість вивести результати якщо вони є, або «Немає дійсних коренів.» якщо результат дискримінанту меньше 0
5. Додав можливість вводити a,b,c. Викликав функцію “find\_roots”

Код програми:

def calculate\_discriminant(a, b, c):

    #Функція для обчислення дискримінанта квадратного рівняння

    return b\*\*2 - 4 \* a \* c

def find\_roots(a, b, c):

    #Функція для пошуку коренів квадратного рівняння

    D = calculate\_discriminant(a, b, c)

    if D > 0:

        x1 = (-b + D\*\*0.5) / (2 \* a)

        x2 = (-b - D\*\*0.5) / (2 \* a)

        print("Корені: x1 =", x1, "x2 =", x2)

    elif D == 0:

        x = -b / (2 \* a)

        print("Єдиний корінь: x =", x)

    else:

        print("Немає дійсних коренів.")

# Введення даних

a = float(input("Введіть коефіцієнт a: "))

b = float(input("Введіть коефіцієнт b: "))

c = float(input("Введіть коефіцієнт c: "))

find\_roots(a, b, c)

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

Хід виконання завдання:

1. Створив функції для кожної операції в калькуляторі а саме (+,-,\*,/). Кожна з цих функції повертає результат двох чисел залежачи від операції через команду return
2. Створив функцію “calculator”. Надав можливісти користувачу вибрати дію створивши змінну “choice”. Також додав команди для можливості вписати число для обчислення
3. Створив if else конструкцію яка буде викликати функцію яка прив’язана до операції яку вибрав користувач роблячи перевірку командами if, else, elif
4. Додав винятки при діленні на 0, та помилку якщо користувач обере не правильну операцію
5. Викликав функцію калькулятора

Код програми:

def add(a, b):

    return a + b

def subtract(a, b):

    return a - b

def multiply(a, b):

    return a \* b

def divide(a, b):

    if b != 0:

        return a / b

    else:

        return "Помилка! Ділення на нуль."

def calculator():

    print("Оберіть операцію:")

    print("1. Додавання")

    print("2. Віднімання")

    print("3. Множення")

    print("4. Ділення")

    choice = input("Введіть номер операції (1/2/3/4): ")

    if choice in ['1', '2', '3', '4']:

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

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

        if choice == '1':

            print("Результат: ", add(a, b))

        elif choice == '2':

            print("Результат: ", subtract(a, b))

        elif choice == '3':

            print("Результат: ", multiply(a, b))

        elif choice == '4':

            print("Результат: ", divide(a, b))

    else:

        print("Невірний вибір!")

# Виклик калькулятора

calculator()

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

Хід виконання завдання:

1. Створив функції для кожної операції в калькуляторі а саме (+,-,\*,/). Кожна з цих функції повертає результат двох чисел залежачи від операції через команду return. Додавши виняток ділення на 0
2. Створив функцію “calculator”. Надав можливісти користувачу вибрати дію створивши змінну “choice”. Також додав команди для можливості вписати число для обчислення.
3. Створив match конструкцію яка буде викликати функцію яка прив’язана до операції яку вибрав користувач. Роблячи перевірку командою case. Додав помилку при виборі всіх неправильніх варіантів операцій
4. Викликав калькулятор

Код програми:

def add(a, b):

    return a + b

def subtract(a, b):

    return a - b

def multiply(a, b):

    return a \* b

def divide(a, b):

    if b != 0:

        return a / b

    else:

        return "Помилка! Ділення на нуль."

def calculator():

    print("Оберіть операцію:")

    print("1. Додавання")

    print("2. Віднімання")

    print("3. Множення")

    print("4. Ділення")

    choice = input("Введіть номер операції (1/2/3/4): ")

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

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

    match choice:

        case '1':

            print("Результат: ", add(a, b))

        case '2':

            print("Результат: ", subtract(a, b))

        case '3':

            print("Результат: ", multiply(a, b))

        case '4':

            print("Результат: ", divide(a, b))

        case \_:

            print("Невірний вибір!")

# Виклик калькулятора

calculator()

посилання на Git:

<https://github.com/MinigamerTY/TP-KB-231-Koledin-Olexandr/tree/main/topic_02>

Звіт до Теми №3

Цикли

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

**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:

1. Створив функцію “calculator” зробивши її нескінечною за допомою команди While
2. Додав можливість обрати операцію користувачу
3. Додав можливість закінчити програму через команду break вписавши «exit»
4. Зробив виключення якщо користувач обрав щось неправильне, яке повертає його на стадію вибору операції калькулятора
5. Додав можливість користувачу ввести числа, також додав помилки при неправильному вводі
6. Через конструкцию if else, зробив перевірку операції яку обрав користувач і в залежності результату вивів обчислення в тій же стрічці
7. Викликав калькулятор

Код програми:

def calculator():

    while True:

        print("Калькулятор:")

        print("Введіть операцію (+, -, \*, /) або 'exit' для виходу.")

        operation = input("Операція: ")

        if operation.lower() == 'exit':

            print("Завершення роботи.")

            break

        if operation not in ('+', '-', '\*', '/'):

            print("Неправильна операція. Спробуйте знову.")

            continue

        try:

            a = float(input("Перше число: "))

            b = float(input("Друге число: "))

        except:

            print("Помилка. Введіть числа.")

            continue

        if operation == '+':

            print(f"Результат: {a + b}")

        elif operation == '-':

            print(f"Результат: {a - b}")

        elif operation == '\*':

            print(f"Результат: {a \* b}")

        elif operation == '/':

            if b != 0:

                print(f"Результат: {a / b}")

            else:

                print("Помилка: Ділення на нуль.")

calculator()

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

Хід виконання завдання:

1. Створив функцію “test\_list” в якій будуть міститися всі команди
2. Створив список my\_list, в який записав випадкові числа
3. Вивів список my\_list додавши (.copy()) для того щоб створити копію списку не змінюючи оригінальний список
4. Вивів список my\_list додавши (.append()) для того щоб додати новий елемент в кінець списку
5. Вивів список my\_list додавши (.extend()) для того щоб додати нові елементи в кінець списку
6. Вивів список my\_list додавши (.insert()) для того щоб вставили новий елемент на позицію з якимось індексом
7. Вивів список my\_list додавши (.remove()) для того щоб прибрати елемент з списку
8. Вивів список my\_list додавши (.reverse()) для того щоб перевернути список
9. Вивів список my\_list додавши (.sort()) для того щоб відсортувати список за зростанням
10. Вивів список my\_list додавши (.clear()) для того щоб очистити список видаливши все
11. Викликав функцію test\_list

Код програми:

def test\_list():

    my\_list = [1,50,56,21,14,18,15,10,10,10,6,88]

    print("Тестування функцій списків:")

    copy\_list = my\_list.copy()

    print("copy():", copy\_list)

    my\_list.append(3)

    print("append(3):", my\_list)

    my\_list.extend([10, 15])

    print("extend([10, 15]):", my\_list)

    my\_list.insert(1, 7)

    print("insert(1, 7):", my\_list)

    my\_list.remove(10)

    print("remove(10):", my\_list)

    my\_list.reverse()

    print("reverse():", my\_list)

    my\_list.sort()

    print("sort():", my\_list)

    my\_list.clear()

    print("clear():", my\_list)

test\_list()

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

Хід виконання завдання:

1. Створив функцію “test\_dict” в якій буду міститися всі команди
2. Створив словник dict з двома парами ключ – значення
3. Вивів наш словник додавши (.update()) для того щоб
4. Видалив елемент словника через команду del
5. Отримав всі ключи з словника через команду keys() та вивів їх
6. Отримав всі значення з словника через команду values() та вивів їх
7. Отримав всі пари ключ-значення з словника через команду items() та вивів їх
8. Видалив все з словника за допомогою команди clear()

Код програми:

def test\_dict():

    dict = {'a': 1, 'b': 2}

    print("Тестування функцій словників:")

    dict.update({'c': 3})

    print("update({'c': 3}):", dict)

    del dict['a']

    print("del['a']:", dict)

    keys = dict.keys()

    print("keys():", keys)

    values = dict.values()

    print("values():", values)

    items = dict.items()

    print("items():", items)

    dict.clear()

    print("clear():", dict)

test\_dict()

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

Хід виконання завдання:

1. Створив функцію “find\_insert\_position”. З значеннями: sorted\_list - наш відсортований список, value = значення нового елементу
2. Проходимо по кожному елементу списку (з індексом i та значенням elem)
3. Якщо значення, яке вставляємо, менше або дорівнює поточному елементу, повертаємо індекс, де можна вставити значення через команду return
4. Додав виключення якщо значення value більше за всі елементи списку, воно має бути вставлено в кінець списку
5. Додав елементи в список, ввів значення value
6. Вивів позицію для вставки нового елементу в список

Код програми:

def find\_insert\_position(sorted\_list, value):

    for i, elem in enumerate(sorted\_list):

        if value <= elem:

            return i

    return len(sorted\_list)

# Тест

sorted\_list = [1, 3, 5, 7, 9, 86]

value = 85

position = find\_insert\_position(sorted\_list, value)

print(f"Позиція для вставки {value}: {position}")

посилання на Git:

<https://github.com/MinigamerTY/TP-KB-231-Koledin-Olexandr/tree/main/topic_03>

Звіт до Теми №4

Виняткові ситуації

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

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

Хід виконання завдання:

1. В існуючий калькулятор, додав винятки, які виводять «Неправильно операція спробуйте знову» при помилковому виборі операції
2. Додав також «Помилка. Введіть числа» коли замість чисел користувач вписує літери або особливі символи

**Розширити функцію ділення обробкою виняткової ситуації ділення но нуль.**

Хід виконання завдання:

1. Додав помилку при діленні на нуль

Код програми:

def calculator():

    while True:

        print("Калькулятор:")

        print("Введіть операцію (+, -, \*, /) або 'exit' для виходу.")

        operation = input("Операція: ")

        if operation.lower() == 'exit':

            print("Завершення роботи.")

            break

        if operation not in ('+', '-', '\*', '/'):

            print("Неправильна операція. Спробуйте знову.")

            continue

        try:

            a = float(input("Перше число: "))

            b = float(input("Друге число: "))

        except:

            print("Помилка. Введіть числа.")

            continue

        if operation == '+':

            print(f"Результат: {a + b}")

        elif operation == '-':

            print(f"Результат: {a - b}")

        elif operation == '\*':

            print(f"Результат: {a \* b}")

        elif operation == '/':

            if b != 0:

                print(f"Результат: {a / b}")

            else:

                print("Помилка: Ділення на нуль.")

calculator()

посилання на Git:

<https://github.com/MinigamerTY/TP-KB-231-Koledin-Olexandr/tree/main/topic_04>

Звіт до Теми №5

Бібліотеки

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

**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:

1. Додав модуль random для вибору випадкового варіанта
2. Створив список choices з 3 варіантами розвитку подій
3. Створив функцію яка обирає переможця через конструкцію if else elif
4. Додав можливість користувачу вибрати камінь ножиці чи папір
5. Створив винятки якщо користувач обирає некоректний варіант розвитку подій
6. Через модуль random зробив випадковий вибір камінь ножиць чи папіру з боку компьютера
7. Створив змінну result, яка виводить переможця через print

Код програми:

import random

choices = ["камінь", "ножиці", "папір"]

# Функція для визначення переможця

def game(player\_choice, computer\_choice):

    if player\_choice == computer\_choice:

        return "Нічия!"

    elif (

         player\_choice == "камінь" and computer\_choice == "ножиці" or

         player\_choice == "ножиці" and computer\_choice == "папір" or

         player\_choice == "папір" and computer\_choice == "камінь"

         ):

        return "Ви виграли!"

    else:

        return "Переміг комп'ютер!"

# Запитуємо користувача на вибір

player\_choice = input("Оберіть камінь, ножиці чи папір: \n").lower()

# Перевіряємо, чи є введення в списку виборів

if player\_choice not in choices:

    print("Некоректний вибір! Будь ласка, оберіть камінь, ножиці чи папір.")

else:

    computer\_choice = random.choice(choices)

    print("Комп'ютер вибрав: ", computer\_choice)

    # Визначаємо результат гри

    result = game(player\_choice, computer\_choice)

    print(result)

**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:

1. Імпортував бібліотеку requests для виконання HTTP-запитів
2. Написав функція для отримання актуальних курсів валют з НБУ, в якій реалізував вивід данних про іноземні валюти з курсу НБУ
3. Перевірив чи успішний запит на вивід даних, якщо запит неуспішний додав вивід помилки
4. Перетворив відповідь у Python-об'єкт (список словників)
5. Створив словник exchange\_rates для збереження курсу валют
6. Додав курси USD, EUR, PLN та значеннях до них в словник
7. Створив функцію convert\_currency для концвертації валюти в гривні
8. Додав перевірку чи введений курс валют є серед USD, EUR, PLN
9. Написав команду для виводу курсу валюти помноженої на кількість гривень
10. Створив функцію main для логічного зв’язку всіх функцій
11. Додав туди вивід доступних валют і можливість обрати користувачу валюту та кількість грошей яку треба конвертувати
12. Додав вивід особливих випадків якщо користувач вводить щось не логічне
13. Додав вивід результату конвертуванню валюти
14. Викликав основну функцію main

Код програми:

import requests

# Функція для отримання актуальних курсів валют з НБУ

def info():

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

    response = requests.get(url)

    if response.status\_code != 200:

        print("Не вдалося отримати курси валют. Спробуйте пізніше.")

        return None

    data = response.json()

    # Створюємо словник з курсами валют

    exchange\_rates = {}

    for item in data:

        if item['cc'] in ['USD', 'EUR', 'PLN']:  # Тільки для цих валют

            exchange\_rates[item['cc']] = item['rate']

    return exchange\_rates

# Функція для конвертації валюти в гривні

def convert\_currency(amount, currency, exchange\_rates):

    if currency not in exchange\_rates:

        print(f"Курс для валюти {currency} не знайдений.")

        return None

    rate = exchange\_rates[currency]

    return amount \* rate

# Головна функція програми

def main():

    # Отримуємо курси валют

    exchange\_rates = info()

    if not exchange\_rates:

        return

    print("Доступні валюти для конвертації: EUR, USD, PLN.")

    currency = input("Введіть валюту, яку хочете конвертувати (EUR, USD, PLN): \n").upper()

    if currency not in ['EUR', 'USD', 'PLN']:

        print("Невірна валюта. Доступні лише EUR, USD, PLN.")

        return

    try:

        amount = float(input("Введіть кількість валютних одиниць: "))

    except ValueError:

        print("Некоректне введення кількості.")

        return

    # Конвертуємо валюту в гривні

    result = convert\_currency(amount, currency, exchange\_rates)

    if result is not None:

        print(f"{amount} {currency} дорівнює {result:.2f} гривень.")

main()

**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:

1. В файлі **functions.py** написав функції математичних дій нашого калькулятора
2. Зробив вивід результату через return
3. Додав виключення при діленні на 0
4. В файлі **operations.py** зробив функцію select\_operations з можливістю обрати математичну дію
5. Створив функцію get\_numbers для вводу числа користувачем
6. В основний файл **calc.py** імпортував функції з файлів **functions.py** та **operations.py** командою import
7. Написав функцію main яка буде видавати результат в залежності від того що обере користувач
8. Зациклив main до нескінченності поки користувач сам не зупинить програму

Код програми:

**calc.py**

from functions import add, subtract, multiply, divide

from operations import get\_numbers, select\_operation

def main():

    while True:

        operation = select\_operation()

        if operation in {'1', '2', '3', '4'}:

            a, b = get\_numbers()

            if operation == '1':

                print(f"Результат: {add(a, b)}")

            elif operation == '2':

                print(f"Результат: {subtract(a, b)}")

            elif operation == '3':

                print(f"Результат: {multiply(a, b)}")

            elif operation == '4':

                try:

                    print(f"Результат: {divide(a, b)}")

                except ValueError as e:

                    print(e)

        else:

            print("Неправильний вибір. Спробуйте ще раз.")

        again = input("Бажаєте продовжити? (так/ні): ").lower()

        if again != 'так':

            print("До побачення!")

            break

**functions.py**

def add(a, b):

    return a + b

def subtract(a, b):

    return a - b

def multiply(a, b):

    return a \* b

def divide(a, b):

    if b == 0:

        raise ValueError("Ділення на нуль неможливе.")

    return a / b

**operations.py**

def get\_numbers():

    try:

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

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

        return a, b

    except ValueError:

        print("Будь ласка, вводьте лише числа.")

        return get\_numbers()

def select\_operation():

    print("Оберіть операцію:")

    print("1 - Додавання")

    print("2 - Віднімання")

    print("3 - Множення")

    print("4 - Ділення")

    return input("Ваш вибір: ")

посилання на Git:

<https://github.com/MinigamerTY/TP-KB-231-Koledin-Olexandr/tree/main/topic_05>

Звіт до Теми №6

Робота з файлами

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

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

Попередні умови: Реалізована програма калькулятор в файлі calc.py, до якого підключають файл functions.py та operations.py. Інструкції до оформлення вмісту файлів розміщенні в завданні 3 до теми 5.

Хід виконання завдання:

1. Скопіював реалізовану програму калькулятор в файлі calc.py, до якого підключають файл functions.py та operations.py.
2. Створив функцію log\_action, яка буде створювати або відкривати текстовий документ calculator\_log.txt та записувати в нього результат обчислення або помилки користувача
3. Додав цю фунцкію у код для зберігання інформації про введені дані, виконану операцію, результат виконання операції над даними, помилками в виборі операцій, та завершення програми користувачем
4. Додав до коду команду /n щоб кожна помилка була надрукована в окремому рядку

Код програми:

**functions.py**

def add(a, b):

    return a + b

def subtract(a, b):

    return a - b

def multiply(a, b):

    return a \* b

def divide(a, b):

    if b == 0:

        raise ValueError("Ділення на нуль неможливе.")

    return a / b

**operations.py**

def get\_numbers():

    try:

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

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

        return a, b

    except ValueError:

        print("Будь ласка, вводьте лише числа.")

        return get\_numbers()

def select\_operation():

    print("Оберіть операцію:")

    print("1 - Додавання")

    print("2 - Віднімання")

    print("3 - Множення")

    print("4 - Ділення")

    return input("Ваш вибір: ")

**calc.py**

from functions import add, subtract, multiply, divide

from operations import get\_numbers, select\_operation

# Функція для запису логів у файл

def log\_action(message):

    with open("calculator\_log.txt", "a") as log\_file:

        log\_file.write(message + "\n")

def main():

    while True:

        operation = select\_operation()

        if operation in {'1', '2', '3', '4'}:

            a, b = get\_numbers()

            try:

                if operation == '1':

                    result = add(a, b)

                    log\_action(f"Додавання: {a} + {b} = {result}")

                elif operation == '2':

                    result = subtract(a, b)

                    log\_action(f"Віднімання: {a} - {b} = {result}")

                elif operation == '3':

                    result = multiply(a, b)

                    log\_action(f"Множення: {a} \* {b} = {result}")

                elif operation == '4':

                    result = divide(a, b)

                    log\_action(f"Ділення: {a} / {b} = {result}")

                print(f"Результат: {result}")

            except ValueError as e:

                log\_action(f"Помилка: {e}")

                print(e)

        else:

            log\_action("Неправильний вибір операції")

            print("Неправильний вибір. Спробуйте ще раз.")

        again = input("\nБажаєте продовжити? (так/ні): ").strip().lower()

        if again != 'так':

            log\_action("Програма завершена користувачем.")

            print("До побачення!")

            break

**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:

1. Створив список students, заповнив його ім’ям та оцінкою студента
2. Створив дві змінні в яких виконав сортування через функцію lambda
3. Провів сортування списка

Код програми:

students = [

    {"name": "Олександр", "score": 85},

    {"name": "Ялад", "score": 92},

    {"name": "Дмитро", "score": 98},

    {"name": "Олена", "score": 88}

]

# Сортування за іменем

sorted\_by\_name = sorted(students, key=lambda x: x["name"])

print("Сортування за іменем:", sorted\_by\_name)

# Сортування за оцінкою

sorted\_by\_score = sorted(students, key=lambda x: x["score"])

print("Сортування за оцінкою:", sorted\_by\_score)

Звіт до Теми №7

ООП

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

**Ознайомитись з існуючими за замовченням методами класу по типу \_\_init\_\_(self) \_\_str\_\_(self)\_\_ та надати приклади використання.**

Хід виконання завдання:

1. Створює клас Person з атрибутами name та age
2. Створив функцію \_\_init\_\_ ініціалізує два атрибути: name (ім'я) і age (вік)
3. Створив функцію \_\_str \_\_ яка відображає об’єкт у вигляді рядку
4. Створив об’єкт
5. Вивід дані через прінт

Код програми:

**task2\_init.py**

class Person:

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

        self.name = name

        self.age = age

# Створення об'єкта

human = Person("Олександр", 25)

print(human.name)

print(human.age)

**task2\_str.py**

class Person:

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

        self.name = name

        self.age = age

    def \_\_str\_\_(self):

        return f"Ім'я: {self.name}, Вік: {self.age}"

# Створення об'єкта

human = Person("Олександр", 30)

print(human)

**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**

Хід виконання завдання:

1. Створив клас Student в якому \_\_init \_\_ ініціалізує два атрибути: name (ім'я) і age (вік), та метод \_\_str\_\_який визначає, як об'єкт Student буде виглядати у вигляді рядка
2. Створив список студентів та заповнив його, кожен новий студент це об’єкт класу Student
3. Відсортував за віком і вивів всіх студентів

Код програми:

class Student:

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

        self.name = name

        self.age = age

    def \_\_str\_\_(self):

        return f"Ім'я: {self.name}, Вік: {self.age}"

# Створення списку об'єктів Student

students = [

    Student("Олександр", 20),

    Student("Катч", 232),

    Student("Дмитро", 19),

    Student("Олександр2", 21)

]

# Сортування за віком

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

# Виведення відсортованих студентів

print("Сортування за віком:")

for student in sorted\_students:

    print(student)

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

Хід виконання завдання:

**Для коду logger4.py:**

1. Створив клас Logger для запису повідомлень у журнал
2. Створив функцію log\_action який приймає self - посилання на поточний екземпляр класу, та message - текст повідомлення, який потрібно записати у файл.
3. Створив змогу автоматично записувати дані в файл calculator\_log.txt та записувати в нього дані після чого закривати файл

**Для коду calc4.py:**

1. Імпортував класи з інших програм
2. Зробив клас CalculatorApp
3. Створив функцію run яка дозволяє калькулятору працювати завжди
4. Додав умови при яких від вибору користувача калькулятор буде обчислювати та виводити результат відповідно вибраної дії
5. Додав виключення при неправильному вводі
6. Та можливісти продовжити та обрати нову операцію

**Для коду userInput4.py:**

1. Створив клас UserInput, додав функцію get\_numbers за допомогою якої користувач вводить числа через input
2. Додав виключення для неправильних введених даних
3. Створив список доступних операцій і можливість їх вибору

**Для коду operations 4.py:**

1. Створив клас Operations в якому створив 4 функції для обчислення множення/ділення/додавання/віднімання
2. Додав виключення при діленні на 0

Код програми:

**calc4.py**

from operations4 import Operations

from userInput4 import UserInput

from logger4 import Logger

class CalculatorApp:

    def \_\_init\_\_(self):

        self.operations = Operations()

        self.userInput4 = UserInput()

        self.logger = Logger()

    def run(self):

        while True:

            operation = self.userInput4.select\_operation()

            if operation in {'1', '2', '3', '4'}:

                a, b = self.userInput4.get\_numbers()

                try:

                    if operation == '1':

                        result = self.operations.add(a, b)

                        self.logger.log\_action(f"Додавання: {a} + {b} = {result}")

                    elif operation == '2':

                        result = self.operations.subtract(a, b)

                        self.logger.log\_action(f"Віднімання: {a} - {b} = {result}")

                    elif operation == '3':

                        result = self.operations.multiply(a, b)

                        self.logger.log\_action(f"Множення: {a} \* {b} = {result}")

                    elif operation == '4':

                        result = self.operations.divide(a, b)

                        self.logger.log\_action(f"Ділення: {a} / {b} = {result}")

                    print(f"Результат: {result}")

                except ZeroDivisionError as e:

                    self.logger.log\_action(f"Помилка: {e}")

                    print(e)

            else:

                print("Неправильний вибір. Спробуйте ще раз.")

            again = input("\nБажаєте продовжити? (так/ні): ").strip().lower()

            if again != 'так':

                print("До побачення!")

                break

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

    app = CalculatorApp()

    app.run()

**logger4.py**

class Logger:

    def log\_action(self, message):

        with open("calculator\_log.txt", "a") as log\_file:

            log\_file.write(message + "\n")

**operations4.py**

class Operations:

    def add(self, a, b):

        return a + b

    def subtract(self, a, b):

        return a - b

    def multiply(self, a, b):

        return a \* b

    def divide(self, a, b):

        if b == 0:

            raise ZeroDivisionError("Помилка: ділення на нуль!")

        return a / b

**userInput4.py**

class UserInput:

    def get\_numbers(self):

        while True:

            try:

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

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

                return a, b

            except ValueError:

                print("Будь ласка, введіть коректні числа.")

    def select\_operation(self):

        print("Оберіть операцію:")

        print("1 - Додавання")

        print("2 - Віднімання")

        print("3 - Множення")

        print("4 - Ділення")

        operation = input("Ваш вибір: ")

        return operation