**Звіт про виконання практичних завдань до лекцій з курсу Технології програмування на мові Python**

Чуб Софія, КБ-231

Звіт до Теми №1

Функції та змінні

Під час виконання практичного завдання до Теми №1 було надано варіанти рішення до наступних задач:

1. Перетворення рядка.

Необхідно рядок, що має вигляд "room318" перетворити наступним чином "813moor", вважаючи сталою довжину рядку в 7 символів.

Хід виконання завдання:

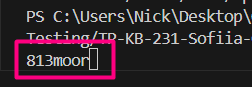
Для виконання завдання треба використати функцію наш рядок записати у змінну number і вивести змінну з операцією зрізу (slice) [::-1] за допомогою функції input. Операція зрізу має формат [start:stop:step], де: start – індекс, з якого починається зріз (включно); stop — індекс, до якого триває зріз (не включено); step – крок, з яким вибираються елементи (якщо крок від'ємний, елементи беруться у зворотному порядку).

Текст програми:

number = "room318"

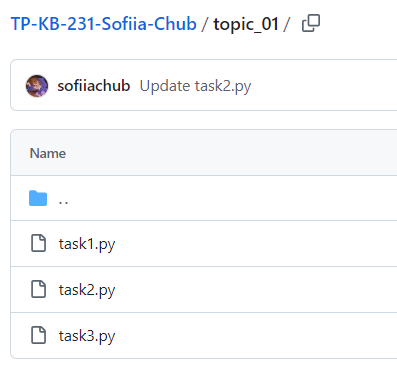
input(number[::-1])

Запущений код:



Посилання на github: <https://github.com/sofiiachub/TP-KB-231-Sofiia-Chub/blob/main/topic_01/task1.py>

Знімок екрану з посилання на github:



1. Базові функції.

Необхідно протестувати функції, що працюють з рядками: strip(), capitalize(), title(), upper(), lower().

Хід виконання завдання:

У змінну text записали текст “ Good day today” який буде виводитись функцією print та змінюватись при використанні таких функції:

* Функція strip() видаляє всі пробіли з початку та кінця рядка (не всередині тексту). Результатом виконання стане: “Good day today”;
* Функція capitalize() робить першу літеру рядка великою, а всі інші – малими. Результатом виконання стане: “ good day today”;
* Функція title() робить кожне слово в рядку таким, що починається з великої літери. Результатом виконання стане: “ Good Day Today”;
* Функція upper() перетворює всі символи в рядку на великі. Результатом виконання стане: “ GOOD DAY TODAY”;
* Функція lower() перетворює всі символи в рядку на малі літери. Результатом виконання стане: “ good day today”.

Текст програми:

text = " Good day today"

print(text)

print(text.strip())

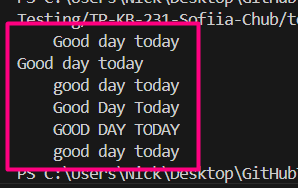
print(text.capitalize())

print(text.title())

print(text.upper())

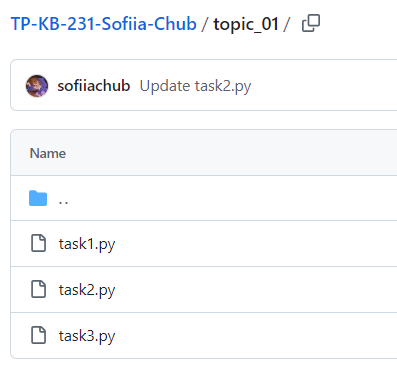
print(text.lower())

Запушений код:



Посилання на github: <https://github.com/sofiiachub/TP-KB-231-Sofiia-Chub/blob/main/topic_01/task2.py>

Знімок екрану з посилання на github:



1. Функція пошуку дискримінанту.

Необхідно написати функцію пошуку дискримінанту квадратного рівняння. Програма приймає 3 значення, обраховує та виводить результат.

Хід виконання завдання:

Програма починає роботу з виклику функції askParameters(), що запитує користувача 3 значення (a, b, c). Кожна змінна виводить повідомлення, а користувач вказує значення для кофіцієнту. Змінна визначає тип данних int. Після цього значення змінних a, b, c повертаються за допомогою return. Функція calcDiscriminant(a, b, c) обчислює дискримінант для квадратного рівняння за формулою та повертає результат у result. Після цього функція print виводить повідомлення "Discriminant = (число) ", замість (числа) підставляється result.

Текст програми:

def askParameters():

a = int(input("What's a: "))

b = int(input("What's b: "))

c = int(input("What's c: "))

return a, b, c

def calcDiscriminant(a, b, c):

result = (b \*\* 2) - 4 \* a \*c

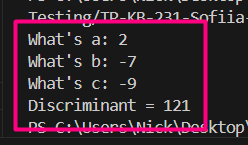
return result

param1, param2, param3 = askParameters()

result = calcDiscriminant(param1, param2, param3)

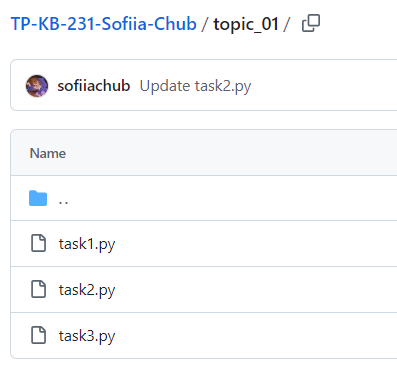
print("Discriminant =", result)

Запущений код:



Посилання на github: <https://github.com/sofiiachub/TP-KB-231-Sofiia-Chub/blob/main/topic_01/task3.py>

Знімок екрану з посилання на github:



Звіт до Теми №2

Умовний перехід

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

1. Функція пошуку коренів квадратного рівняння.

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

Хід виконання завдання:

Основу коду було взято з 3 завдання 1 теми.

Принцип роботи: Викликається функція askParameters(), яка просить користувача ввести три параметри: a, b та c. Функція повертає значення.

Для обчислення дискримінанту викликається функція calcDiscriminant(a, b, c), яка приймає коефіцієнти a, b та c. Вона обчислює D за формулою і повертає його значення.

Для обчислення коренів рівняння викликається функція calcRoots(a, b, c, D), яка приймає коефіцієнти та значення дискримінанту. В залежності від значення дискримінанту, функція визначає кількість коренів

Після обчислення коренів, програма виводить результат "Roots : ...".

Припустимо, що користувач вводить такі значення a=1, b=-3, c=2. Обчисливши дискримінант ми отримуємо , тобто , отже коренями будуть та .

Текст програми:

import math

def askParameters():

a = int(input("What's a: "))

b = int(input("What's b: "))

c = int(input("What's c: "))

return a, b, c

def calcDiscriminant(a, b, c):

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

return D

def calcRoots(a, b, c, D):

if D > 0:

x1 = (-b + math.sqrt(D)) / (2 \* a)

x2 = (-b - math.sqrt(D)) / (2 \* a)

return x1, x2

elif D == 0:

x = -b / (2 \* a)

return x

else:

root = "There are no"

return root

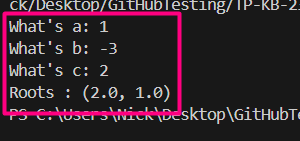
param1, param2, param3 = askParameters()

discriminant = calcDiscriminant(param1, param2, param3)

roots = calcRoots(param1, param2, param3, discriminant)

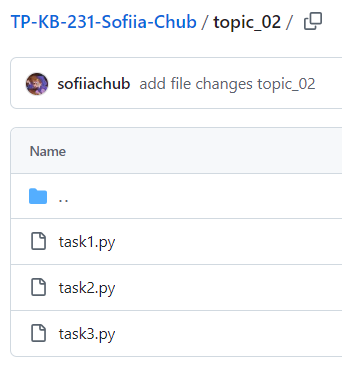
print("Roots :", roots)

Запущений код:



Посилання на github: <https://github.com/sofiiachub/TP-KB-231-Sofiia-Chub/blob/main/topic_02/task1.py>

Знімок екрану з посилання на github:



1. Програма калькулятор з конструкцією if else.

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

Хід виконання завдання:

Введення чисел та операції:

Принцип роботи: Користувач запрошується ввести два числа: num1 та num2. Користувач також вибирає математичну операцію (+, -, \*, /).

Викоикається функція calculator(num1, num2, operation), яка приймає два числа та обрану операцію. У функції calculator в залежності від значення змінної operation, викликається відповідна математична функція:

* Якщо операція +, викликається функція add(num1, num2).
* Якщо операція -, викликається функція subtract(num1, num2).
* Якщо операція \*, викликається функція multiply(num1, num2).
* Якщо операція /, викликається функція divide(num1, num2).
* Якщо введена невідома операція, функція повертає "Error.".

У функції divide, якщо num2 дорівнює 0, повертається повідомлення "Error: Division by zero.". Інакше виконується ділення num1 / num2.

Обчислений результат повертається з функції calculator і виводиться на екран у форматі "Result: ...".

Припустимо, що користувач вводить такі значення a=10, b=2 та operation=/. Результатом буде число 5.

Текст програми:

def add(num1, num2):

return num1 + num2

def subtract(num1, num2):

return num1 - num2

def multiply(num1, num2):

return num1 \* num2

def divide(num1, num2):

if num2 != 0:

return num1 / num2

else:

return "Error: Division by zero."

def calculator(num1, num2, operation):

if operation == "+":

result = add(num1, num2)

elif operation == "-":

result = subtract(num1, num2)

elif operation == "\*":

result = multiply(num1, num2)

elif operation == "/":

result = divide(num1, num2)

else:

result = "Error."

return result

num1 = int(input("What's a: "))

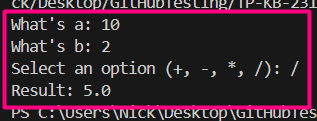
num2 = int(input("What's b: "))

operation = input("Select an option (+, -, \*, /): ")

result = calculator(num1, num2, operation)

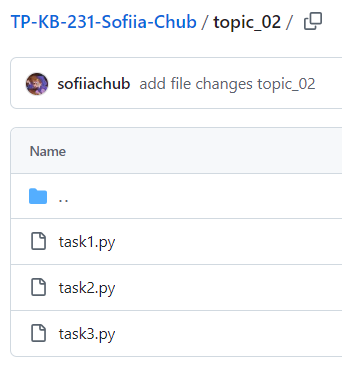
print("Result:", result)

Запущений код:



Посилання на github: <https://github.com/sofiiachub/TP-KB-231-Sofiia-Chub/blob/main/topic_02/task2.py>

Знімок екрану з посилання на github:



1. Програма калькулятор з конструкцією match.

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

Хід виконання завдання:

Принцип роботи: Користувач запрошується ввести два числа: num1 та num2. Користувач також вибирає математичну операцію (+, -, \*, /).

Викоикається функція calculator(num1, num2, operation), яка приймає два числа та обрану операцію. Викликається функція calculator(num1, num2, operation), яка приймає два числа та обрану операцію.

У функції calculator використовується конструкція match-case для визначення операції:

* Якщо операція +, викликається функція add(num1, num2).
* Якщо операція -, викликається функція subtract(num1, num2).
* Якщо операція \*, викликається функція multiply(num1, num2).
* Якщо операція /, викликається функція divide(num1, num2).
* Якщо введена невідома операція (жоден з case не спрацює), повертається "Error.".

У функції divide, якщо num2 дорівнює 0, повертається повідомлення "Error: Division by zero.". Інакше виконується ділення num1 / num2.

Обчислений результат повертається з функції calculator і виводиться на екран у форматі "Result: ...".

Припустимо, що користувач вводить такі значення a=3, b=8 та operation=\*. Результатом буде число 24.

Текст програми:

def add(num1, num2):

return num1 + num2

def subtract(num1, num2):

return num1 - num2

def multiply(num1, num2):

return num1 \* num2

def divide(num1, num2):

if num2 != 0:

return num1 / num2

else:

return "Error. Division by zero."

def calculator(num1, num2, operation):

match operation:

case "+":

return add(num1, num2)

case "-":

return subtract(num1, num2)

case "\*":

return multiply(num1, num2)

case "/":

return divide(num1, num2)

case \_:

return "Error."

num1 = int(input("What`s a: "))

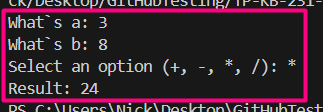
num2 = int(input("What`s b: "))

operation = input("Select an option (+, -, \*, /): ")

result = calculator(num1, num2, operation)

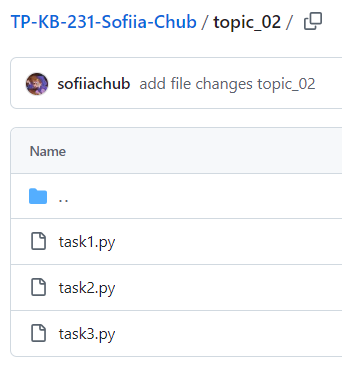
print("Result:", result)

Запущений код:



Посилання на github: <https://github.com/sofiiachub/TP-KB-231-Sofiia-Chub/blob/main/topic_02/task3.py>

Знімок екрану з посилання на github:



Звіт до Теми №3

Цикли

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

1. Програма калькулятор з запитами.

Необхідно написати програму калькулятор з постійними запитами на введення нових даних та операцій. За основу взяла програму калькулятор з попередньої теми. Реалізувала механізм завершення програми після отримання відповідної команди.

Хід виконання завдання:

Програма запитує користувача, чи бажає він розпочати роботу з калькулятором. Якщо відповідь “Yes”, запускається цикл. У цьому циклі користувач вводить два числа, обирає арифметичну операцію, після чого викликається функція calculator, яка обчислює результат. Отриманий результат виводиться на екран. Після цього програма запитує користувача, чи хоче він далі продовжити роботу. Якщо відповідь знову “Yes”, цикл повторюється. Якщо користувач вводить “No” програма завершується.

Текст програми:

def calculator(num1, num2, operation):

match operation:

case "+":

return num1 + num2

case "-":

return num1 - num2

case "\*":

return num1 \* num2

case "/":

if num2 != 0:

return num1 / num2

else:

return "Error. Division by zero."

case \_:

return "Error."

start = input("This is a calculator. Will we start?(Yes/No)")

while start == "Yes":

num1 = int(input("What`s a: "))

num2 = int(input("What`s b: "))

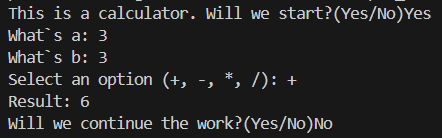
operation = input("Select an option (+, -, \*, /): ")

result = calculator(num1, num2, operation)

print("Result:", result)

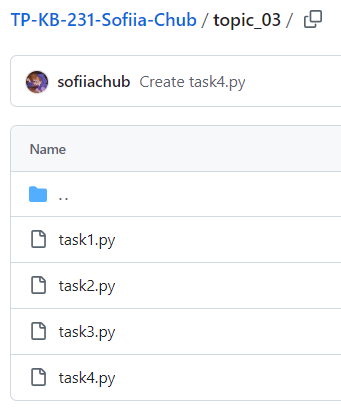
start = input("Will we continue the work?(Yes/No)")

Запущений код:



Посилання на github: <https://github.com/sofiiachub/TP-KB-231-Sofiia-Chub/blob/main/topic_03/task1.py>

Знімок екрану з посилання на github:



1. Тестування функцій списків.

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

Хід виконання завдання:

* Функція extend додає всі елементи з ітерованого об'єкта до кінця списку: childrenList.extend(['Ilona', 'Nata']) додає елементи 'Ilona' і 'Nata' до childrenList.
* Функція append додає один елемент до кінця списку: childrenList.append('Jusel') додає 'Jusel' в кінець списку.
* Функця insert вставляє елемент на вказану індекс у списку: childrenList.insert(5, 'Rita') вставляє 'Rita' на шосту позицію індекс 5.
* Функція remove видаляє перший елемент зі списку, що дорівнює вказаному значенню: childrenList.remove('Stacey') видаляє 'Stacey' зі списку.
* Функція clear очищає список, видаляючи всі його елементи: childrenListCopy.clear() робить childrenListCopy порожнім списком.
* Функція sort сортує елементи списку в порядку зростання (за замовчуванням алфавітно для рядків): childrenList.sort() відсортовує childrenList в алфавітному порядку.
* Функція reverse змінює порядок елементів списку на протилежний:childrenList.reverse() перевертає порядок елементів у списку.
* Функція copy створює поверхневу копію списку і повертає її: childrenListCopy = childrenList.copy() створює новий список childrenListCopy, який містить ті ж елементи, що й childrenList, але є незалежним від оригіналу.

Текст програми:

childrenList = ['Stacey', 'Thomas', 'Newt', 'Jery']

print(childrenList)

childrenList.extend(['Ilona', 'Nata'])

print(childrenList)

childrenList.append('Jusel')

print(childrenList)

childrenList.insert(5, 'Rita')

print(childrenList)

childrenList.remove('Stacey')

print(childrenList)

childrenList.sort()

print(childrenList)

childrenList.reverse()

print(childrenList)

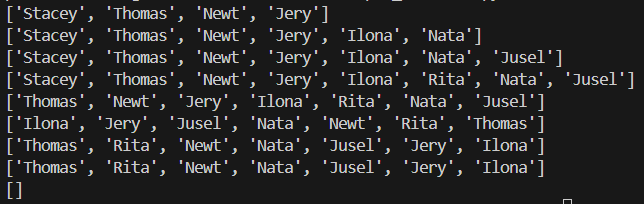
childrenListCopy = childrenList.copy()

print(childrenListCopy)

childrenListCopy.clear()

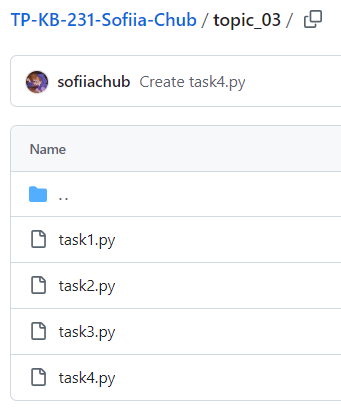
print(childrenListCopy)

Запущений код:



Посилання на github: <https://github.com/sofiiachub/TP-KB-231-Sofiia-Chub/blob/main/topic_03/task2.py>

Знімок екрану з посилання на github:



1. Тестування функцій для словників.

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

Хід виконання завдання:

* Функція update додає пари ключ-значення з іншого словника до поточного словника: pincilCase.update({'sharpener': 4, 'pen': 6}) додає нову пару 'sharpener': 4 та оновлює значення ключа 'pen' до 6.
* Оператор del видаляє елемент з словника за вказаним ключем: del pincilCase['eraser'] видаляє пару ключ-значення, де ключ 'eraser'.
* Функція clear очищає словник, видаляючи всі його пари ключ-значення: pincilCase.clear() робить pincilCase порожнім словником.
* Функція keys повертає об'єкт, що містить усі ключі словника: print(pincilCase.keys()) виводить усі ключі словника у вигляді об'єкта dict\_keys.
* Функція values повертає об'єкт, що містить усі значення словника: print(pincilCase.values()) виводить усі значення словника у вигляді об'єкта dict\_values.
* Функція items повертає об'єкт, що містить усі пари ключ-значення словника у вигляді кортежів: print(pincilCase.items()) виводить пари ключ-значення у вигляді об'єкта dict\_items.

Текст програми:

pincilCase = {'pen': 3, 'pencil': 5, 'eraser': 2, 'ruler': 1}

print(pincilCase)

pincilCase.update({'sharpener': 4, 'pen': 6})

print(pincilCase)

del pincilCase['eraser']

print(pincilCase)

print(pincilCase.keys())

print(pincilCase)

print(pincilCase.values())

print(pincilCase)

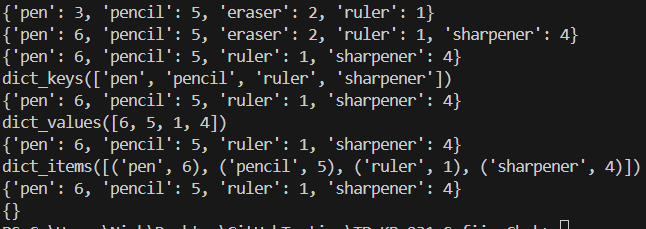
print(pincilCase.items())

print(pincilCase)

pincilCase.clear()

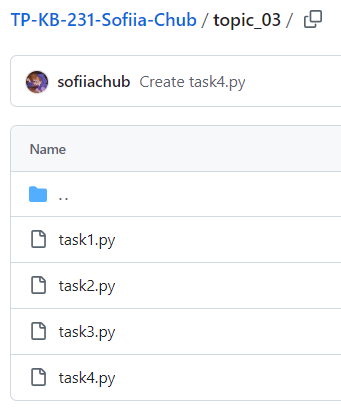
print(pincilCase)

Запущений код:



Посилання на github: <https://github.com/sofiiachub/TP-KB-231-Sofiia-Chub/blob/main/topic_03/task3.py>

Знімок екрану з посилання на github:



1. Функція пошуку позиції для вставки

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

Хід виконання завдання:

Для визначення позиції, на якій можна вставити певне число в відсортований список, спочатку створюється список numberList, що містить кілька чисел. Цей список сортується за допомогою функції sort, після чого його виводять на екран. Користувача просять ввести число, яке буде використовуватися для визначення позиції вставки.

Функція findPosition перебирає всі елементи списку за допомогою циклу for. Якщо введене число менше або дорівнює елементу на поточній позиції, функція повертає індекс цього елемента. Якщо ж число більше за всі елементи, функція повертає довжину списку, що вказує на необхідність вставки числа в кінець.

Після цього, використовуючи знайдену позицію, введене число вставляється в список за допомогою функції insert, а оновлений список виводиться на екран.

Текст програми:

def findPosition(numberList, number):

insertPosition = 0

for item in numberList:

if number > item:

insertPosition += 1

else:

break

return insertPosition

numberList = [1, 3, 4, 2, 9, 10]

numberList.sort()

print("Наш сортований лист:", numberList)

number = int(input("Введіть число: "))

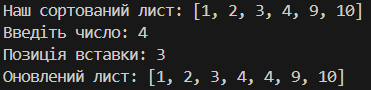
position = findPosition(numberList, number)

print("Позиція вставки:", position)

numberList.insert(position, number)

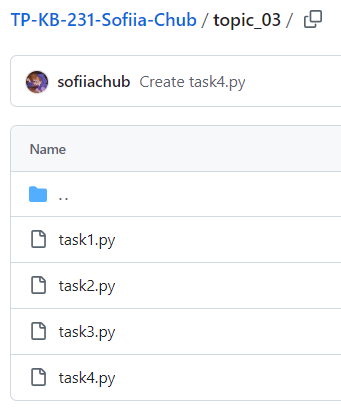
print("Оновлений лист:", numberList)

Запущений код:



Посилання на github: <https://github.com/sofiiachub/TP-KB-231-Sofiia-Chub/blob/main/topic_03/task4.py>

Знімок екрану з посилання на github:



Звіт до Теми №4

Виняткові ситуації

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

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

Хід виконання завдання:

До реалізованої програми калькулятор, що використовує метод нескінченного введення даних для обробки. Всі дії (додавання, віднімання, множення, ділення) реалізовані як окремі функції та використовуються у відповідних місцях додали функцію перевірки значень ValueError, яка при отриманні іншого типу даних виводить повідомлення про помилку.

Текст програми:

def add(num1, num2):

return num1 + num2

def subtract(num1, num2):

return num1 - num2

def multiply(num1, num2):

return num1 \* num2

def divide(num1, num2):

if num2 != 0:

return num1 / num2

else:

return "Error. Division by zero"

def calculator(num1, num2, operation):

match operation:

case "+":

return add(num1, num2)

case "-":

return subtract(num1, num2)

case "\*":

return multiply(num1, num2)

case "/":

return divide(num1, num2)

case \_:

return "Error"

start = input("This is a calculator. Will we start? (Yes/No): ")

while start == "Yes":

try:

num1 = float(input("What's a: "))

num2 = float(input("What's b: "))

operation = input("Select an option (+, -, \*, /): ")

if operation not in ["+", "-", "\*", "/"]:

print("An invalid operation was entered")

else:

result = calculator(num1, num2, operation)

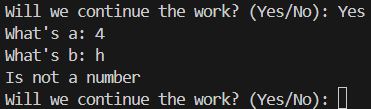
print("Result:", result)

except ValueError:

print("Is not a number")

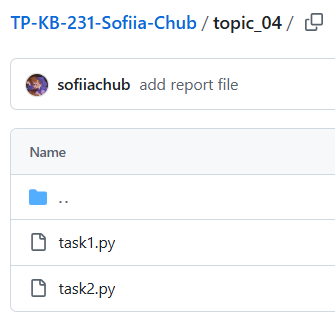
start = input("Will we continue the work? (Yes/No): ")

Запущений код:



Посилання на github: <https://github.com/sofiiachub/TP-KB-231-Sofiia-Chub/blob/main/topic_04/task1.py>

Знімок екрану з посилання на github:



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

Хід виконання завдання:

До коду з першого завдання в функцію ділення додали функцію виняткової ситуації raise, яка виводить повідомлення коли користувач намагається число поділити на нуль.

Текст програми:

def add(num1, num2):

return num1 + num2

def subtract(num1, num2):

return num1 - num2

def multiply(num1, num2):

return num1 \* num2

def divide(num1, num2):

if num2 != 0:

return num1 / num2

else:

raise ZeroDivisionError

def calculator(num1, num2, operation):

match operation:

case "+":

return add(num1, num2)

case "-":

return subtract(num1, num2)

case "\*":

return multiply(num1, num2)

case "/":

return divide(num1, num2)

case \_:

return "Error"

start = input("This is a calculator. Will we start? (Yes/No): ")

while start == "Yes":

try:

num1 = float(input("What's a: "))

num2 = float(input("What's b: "))

operation = input("Select an option (+, -, \*, /): ")

if operation not in ["+", "-", "\*", "/"]:

print("An invalid operation was entered")

else:

result = calculator(num1, num2, operation)

print("Result:", result)

except ValueError:

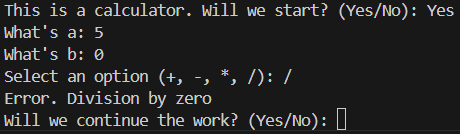
print("Is not a number")

except ZeroDivisionError:

print("Error. Division by zero")

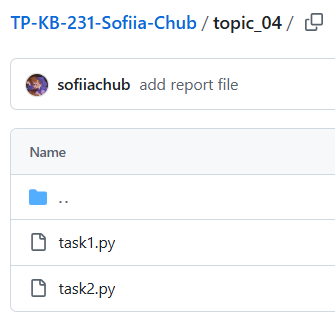
start = input("Will we continue the work? (Yes/No): ")

Запущений код:



Посилання на github: <https://github.com/sofiiachub/TP-KB-231-Sofiia-Chub/blob/main/topic_04/task2.py>

Знімок екрану з посилання на github:



Звіт до Теми №5

Бібліотеки

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

1. 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 програма визначає переможця.

Хід виконання завдання:

Функція checkTheAnswer(choice, result):

* Отримує два параметри: choice — це вибір користувача (камінь, ножиці або папір), а result — це вибір комп'ютера.
* Порівнюються ці вибори за допомогою умовних операторів: Якщо вибір користувача та комп'ютера однакові, то результат — нічия; якщо вибір користувача перемагає комп'ютера (за правилами гри: камінь перемагає ножиці, ножиці — папір, папір — камінь), то користувач виграє; в інших випадках переможе комп'ютер.

Програма запитує у користувача його вибір через функцію input(). Користувач може ввести "stone", "scissor" або "paper". Комп'ютер випадковим чином вибирає один з трьох варіантів за допомогою random.choice().

Виведення результату:

Програма виводить, який вибір зробив комп'ютер, і потім викликає функцію checkTheAnswer(), щоб визначити результат гри (виграв користувач, комп'ютер або нічия).

Текст програми:

import random

def checkTheAnswer(choice, result):

if choice == result:

return "Draw"

elif (choice == "stone" and result == "scissor") or (choice == "scissor" and result == "paper") or (choice == "paper" and result == "stone"):

return "Winner: user"

else:

return "Winner: computer"

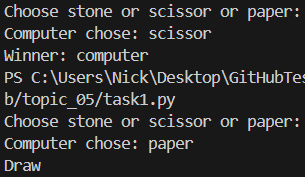
choice = input("Choose stone or scissor or paper: ")

result = random.choice(["stone", "scissor", "paper"])

print("Computer chose:", result)

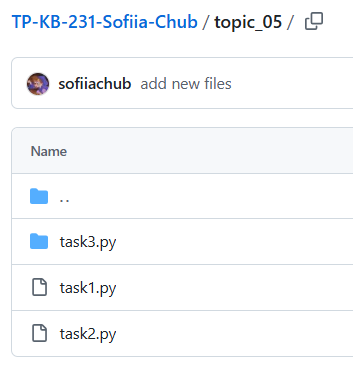
print(checkTheAnswer(choice, result))

Запущений код:



Посилання на github: <https://github.com/sofiiachub/TP-KB-231-Sofiia-Chub/blob/main/topic_05/task1.py>

Знімок екрану з посилання на github:



1. Програма конвертування іноземної валюти в українську гривню. Для отримання актуальних курсів валют використовуємо 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:

Введення даних користувачем:

* Користувач вводить кількість валюти (value) і тип валюти (currency), наприклад, EUR, USD або PLN. Валюта вводиться у вигляді трьох літер (наприклад, EUR для євро).
* Введене значення валюти приводиться до верхнього регістру за допомогою .upper(), щоб уникнути проблем з регістром при порівнянні.

Отримання даних з API НБУ:

* За допомогою бібліотеки requests виконується запит до API НБУ, який повертає список курсів валют у форматі JSON.
* URL запиту: https://bank.gov.ua/NBUStatService/v1/statdirectory/exchange?json

Обробка відповіді API:

* Отриману відповідь перетворюємо на формат JSON за допомогою r.json().
* Потім перебираємо всі елементи списку r.json() (який містить інформацію про різні валюти).

Знаходження потрібної валюти:

* Для кожного елемента списку перевіряється, чи збігається код валюти (елемент cc) з введеним користувачем типом валюти.
* Якщо знайдено відповідну валюту, то курс валюти (елемент rate) множиться на введену кількість валюти, і результат виводиться у гривнях (UAH).
* Якщо валюта не знайдена, виводиться повідомлення, що курс цієї валюти не знайдено.

Результат:

Якщо валюта знайдена, програма виведе, скільки це буде в гривнях (UAH), округлене до двох знаків після коми.

Якщо валюта не знайдена, виводиться повідомлення: "Exchange rate for currency {currency} not found".

Текст програми:

import requests

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

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

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

eurCurr = 0

for elem in r.json():

if elem['cc'] == currency:

eurCurr = elem['rate']

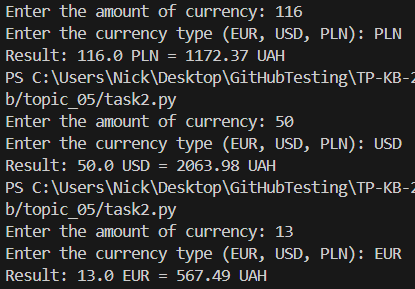
print(f"Result: {value} {currency} = {eurCurr\*value:.2f} UAH")

break

else:

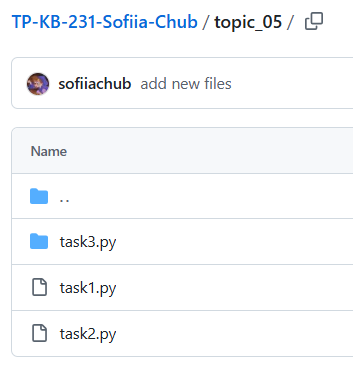
print(f"Exchange rate for currency {currency} not found")

Запущений код:



Посилання на github: <https://github.com/sofiiachub/TP-KB-231-Sofiia-Chub/blob/main/topic_05/task2.py>

Знімок екрану з посилання на github:



1. 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.

Хід виконання завдання:

1. Файл calc.py

Функції, що імпортуються:

* add, subtract, multiply, divide — це математичні операції, що імпортуються з functions.py.
* myinput, operation — це функції для введення даних і обробки вибору операції, що імпортуються з operations.py.

Функція main:

Викликає функцію myinput() для введення двох чисел і операції. Потім передає ці значення разом з операціями до функції operation, щоб виконати обрану операцію. Виводить результат.

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

Переконується, що функція main() виконується тільки тоді, коли цей файл запускається безпосередньо, а не коли він імпортується як модуль в інший файл.

1. Файл operations.py

Функція myinput:

* Запитує у користувача два числа і вибір операції (одну з чотирьох: "+", "-", "\*", "/").
* Повертає ці значення, щоб передати їх далі в програму.

Функція operation:

* Отримує два числа, операцію та математичні функції (додавання, віднімання тощо).
* В залежності від вибору операції (плюс, мінус, множення, ділення), викликає відповідну функцію з functions.py.
* Якщо вибір операції неправильний, повертається повідомлення про помилку.

3) Файл functions.py

Функції:

* add(a, b): Додає два числа.
* subtract(a, b): Віднімає друге число від першого.
* multiply(a, b): Множить два числа.
* divide(a, b): Діє на два числа, при цьому якщо друге число — це 0, то повертається повідомлення про помилку "Error. Division by zero".

Програма працює так:

Користувач запускає програму (calc.py), яка викликає функцію main().

Функція main() викликає функцію myinput() з operations.py, де користувач вводить два числа і вибір операції. Потім значення передаються до функції operation() (також з operations.py), яка вибирає правильну математичну операцію на основі введеної користувачем операції і викликає відповідну функцію з functions.py. Результат обчислень виводиться на екран.

Текст програми:

Calc.py

from functions import add, subtract, multiply, divide

from operations import myinput, operation

def main():

num1, num2, operation1 = myinput()

result = operation(num1, num2, operation1, add, subtract, multiply, divide)

print("Result:", result)

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

main()

Operations.py

def myinput():

print("Enter two numbers: ")

num1 = float(input("First number: "))

num2 = float(input("Second number: "))

operation = input("Select an operation (+, -, \*, /): ")

return num1, num2, operation

def operation(num1, num2, operation, add, subtract, multiply, divide):

match operation:

case "+":

return add(num1, num2)

case "-":

return subtract(num1, num2)

case "\*":

return multiply(num1, num2)

case "/":

return divide(num1, num2)

case \_:

return "Incorrect operations"

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:

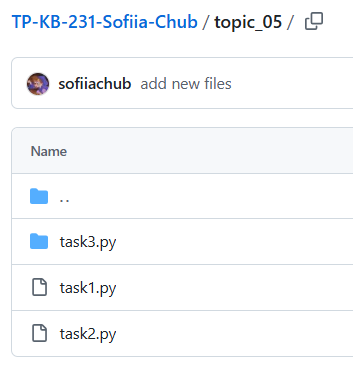
return "Error. Division by zero"

return a / b

Запущений код:

Посилання на github: <https://github.com/sofiiachub/TP-KB-231-Sofiia-Chub/tree/main/topic_05/task3.py>

Знімок екрану з посилання на github:



Звіт до Теми №6

Робота з файлами

Під час виконання практичного завдання до Теми №6 було надано варіанти рішення до наступних задач, попередня умовова яких програма калькулятор в файлі calc.py, до якого підключають файл functions.py та operations.py:

1. Маючи не відсортований список студентів, елементами якого є словники з двома параметрами (група та електронна пошта) виконати сортування списку, використовуючи функцію lambda функція, що повертає ім’я.

Хід виконання завдання:

1. Спочатку створюється список, що містить інформацію про студентів. Кожен студент — це кортеж із трьох елементів: ім'я, група, email.
2. Список студентів потрібно відсортувати. Оскільки ми хочемо спочатку сортувати за групами, а потім за іменами, програма використовує функцію sorted(). Ключ сортування — це функція, яка визначає порядок, за яким студенти будуть впорядковані. Вона бере два елементи: група та ім'я студента. Таким чином, студенти спочатку будуть відсортовані по групах, а в межах кожної групи – за алфавітним порядком їхніх імен.
3. Програма використовує змінну groupBig для того, щоб відслідковувати, яку групу ми вже вивели. Для кожного студента в відсортованому списку: Якщо група студента не збігається з попередньо виведеною (змінною groupBig), то виводиться заголовок цієї групи (наприклад, "Група CS-1"). Змінна groupBig оновлюється на поточну групу. Якщо група вже виведена, то заголовок групи не виводиться знову, і програма переходить до виведення студентів цієї групи.
4. Після сортування виводиться заголовок “Відсортований список студентів, після програма виводить імена та email-адреси студентів цієї групи.
5. Програма працює так: студенти спочатку сортуються за групою, а потім за іменем. Це дозволяє вивести їх групами та по алфавітному порядку в межах груп. Коли програма виводить студентів, вона перевіряє, чи змінилася група і в разі зміни групи виводить назву нової групи. Програма ітерує через список відсортованих студентів, перевіряючи кожного студента і виводячи його ім'я та email, при необхідності додаючи заголовок для нової групи.

Текст програми:

students = [

("Oil", "CS-1", "[oil@gmail.com](mailto:oil@gmail.com)"),

("Bob", "CS-2", "[bob@gmail.com](mailto:bob@gmail.com)"),

("Marta", "CS-2", "[marta@gmail.com](mailto:marta@gmail.com)"),

("Dima", "CS-2", "[dima@gmail.com](mailto:dima@gmail.com)"),

("Eve", "CS-1", "[eve@gmail.com](mailto:eve@gmail.com)"),

("Zoe", "CS-2", "[zoe@gmail.com](mailto:zoe@gmail.com)"),

("Josh", "CS-1", "[josh@gmail.com](mailto:josh@gmail.com)"),

]

sortStudents = sorted(students, key=lambda x: (x[1], x[0]))

print("Відсортований список студентів:")

groupBig = None

for student in sortStudents:

name, group, gmail = student

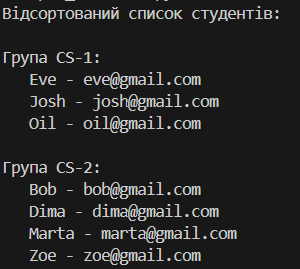
if group != groupBig:

print(f"\nГрупа {group}:")

groupBig = group

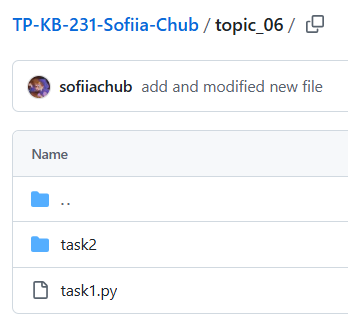
print(f" {name} - {gmail}")

Запущений код:



Посилання на github: <https://github.com/sofiiachub/TP-KB-231-Sofiia-Chub/tree/main/topic_06/task2>

Знімок екрану з посилання на github:



1. Розробити механізм логування дій, що виконує програма калькулятор. Забезпечити зберігання інформації про введені данні, виконану операцію та результат виконання операції над даними, а також вивід помилок, що сталися при помилковому введені.

Хід виконання завдання:

1. Файл calc1.py

Функції, що імпортуються:

* add, subtract, multiply, divide — це математичні операції, що імпортуються з functions.py.
* myinput, operation, recordHistory — функції для введення даних, виконання операцій і запису історії, що імпортуються з файлу operations1.py.

Запуск програми: Програма починається з запиту у користувача, чи готовий він розпочати (Yes/No). Якщо відповідь "Yes", програма переходить до циклу, де користувач вводить два числа і вибирає математичну операцію.

Введення даних: Викликається функція myinput(), яка запитує у користувача два числа та обрану операцію (одну з чотирьох: +, -, \*, /).

Перевірка коректності вводу: Якщо введено правильні числа та коректну операцію, результат обчислення передається в функцію operation(), яка виконує відповідну операцію (додавання, віднімання, множення, ділення). Результат виводиться на екран.

Запис історії: Якщо операція виконана правильно, історія дії з результатом записується у файл fileActions. Якщо введено некоректні дані, записується в файл fileMistakes.

Продовження роботи: Після кожного виконання операції програма запитує, чи хоче користувач продовжити роботу (Yes/No). Якщо відповідь "No", програма завершується.

1. Файл operations1.py

Функції:

* myinput(): Запитує у користувача два числа та операцію для обчислень. Повертає введені значення або None, якщо сталася помилка вводу (наприклад, введено нечислові значення).
* operation(): Виконує обчислення на основі введених чисел і операції:

Вибирає одну з математичних операцій (+, -, \*, /).

Якщо операція неправильна або ввід некоректний, повертається повідомлення про помилку.

* recordHistory(): Записує історію виконаних операцій у файли:

Якщо операція виконана успішно, записується в fileActions.

Якщо сталася помилка (наприклад, ділення на нуль), записується в fileMistakes.

3) Файл functions1.py

Функції:

* add(a, b): Додає два числа.
* subtract(a, b): Віднімає друге число від першого.
* multiply(a, b): Множить два числа.
* divide(a, b): Діє на два числа, при цьому якщо друге число — це 0, то повертається повідомлення про помилку "Error. Division by zero".

Програма працює так:

Користувач запускає програму (сalc1.py), де запитується, чи хоче він розпочати. У разі позитивної відповіді, функція myinput() запитує у користувача числа та операцію. Введені дані передаються у функцію operation(), яка вибирає відповідну математичну операцію з файлу functions1.py і виконує її.

Якщо все в порядку, результат виводиться на екран, а сама операція записується в файл fileActions. У разі помилки (неправильний ввід чи операція), програма записує помилку у файл fileMistakes. Після виконання операції користувача запитують, чи хоче він продовжити роботу, і так повторюється, поки користувач не вибере "No".

Текст програми:

Calc1.py:

from functions1 import addd, subtract, multiply, divide

from operations1 import myinput, operation, recordHistory

start = input("This is a calculator. Will we start? (Yes/No): ")

while start.lower() == "yes":

num1, num2, operation1 = myinput()

if num1 is not None and num2 is not None and operation1 in ['+', '-', '\*', '/']:

result = operation(num1, num2, operation1, addd, subtract, multiply, divide)

print("Result:", result)

recordHistory(num1, num2, operation1, result, True)

else:

print("Invalid input, please try again.")

recordHistory(num1, num2, operation1, "Invalid input", False)

start = input("Will we continue the work? (Yes/No): ")

Operation1.py:

def myinput():

print("Enter two numbers: ")

try:

num1 = float(input("First number: "))

num2 = float(input("Second number: "))

operation = input("Select an operation (+, -, \*, /): ")

return num1, num2, operation

except ValueError:

print("Error: Invalid input. Please enter numbers.")

return None, None, None

def operation(num1, num2, operation, add, subtract, multiply, divide):

if num1 is None or num2 is None:

return "Invalid input, cannot perform operation."

match operation:

case "+":

return add(num1, num2)

case "-":

return subtract(num1, num2)

case "\*":

return multiply(num1, num2)

case "/":

try:

return divide(num1, num2)

except ZeroDivisionError:

return "Error: Division by zero"

case \_:

return "Incorrect operation. Please use +, -, \*, or /."

def recordHistory(num1, num2, operation1, result, trueDate):

if trueDate:

with open("fileActions", "a") as file:

file.write(f"\tYour Action:\n")

file.write(f"{num1} {operation1} {num2} = {result}\n")

else:

with open("fileMistakes", "a") as file:

file.write(f"Invalid input or operation. num1: {num1}, num2: {num2}, operation: {operation1}\n")

Functions1.py:

def addd(a, b):

return a + b

def subtract(a, b):

return a - b

def multiply(a, b):

return a \* b

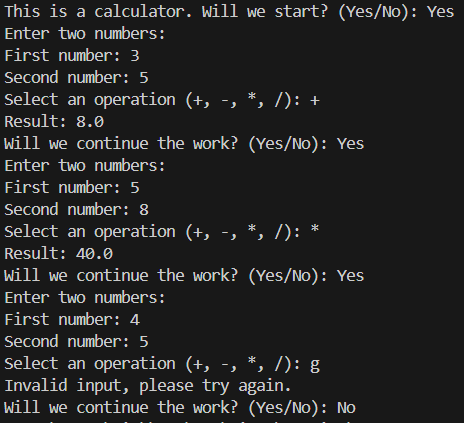
def divide(a, b):

if b == 0:

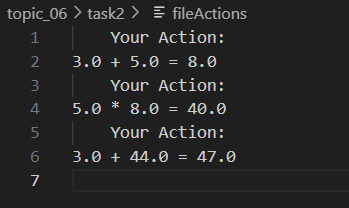
return "Error: Division by zero"

return a / b

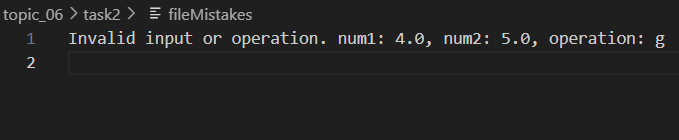
Запущений код:



FileActions:

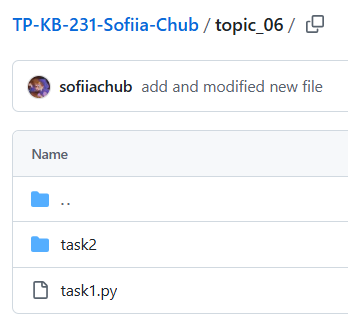


FileMistakes:



Посилання на github: <https://github.com/sofiiachub/TP-KB-231-Sofiia-Chub/blob/main/topic_06/task1.py>

Знімок екрану з посилання на github:



Звіт до Теми №7

Об’єктно-орієнтоване програмування

Під час виконання практичного завдання до Теми №7 було надано варіанти рішення до наступних задач, попередня умовова яких програма калькулятор в файлі calc.py, до якого підключають файл functions.py та operations.py:

1. 1) Ознайомитись з документацією що описує можливості використання класів у мові Python <https://docs.python.org/3/tutorial/classes.html>

З документацією була ознайомлена.

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

Метод \_\_init\_\_(self): Метод \_\_init\_\_ є конструктором класу. Він автоматично викликається під час створення нового об'єкта цього класу. Використовується для ініціалізації атрибутів нового об'єкта і виконання будь-яких необхідних дій, таких як підготовка стану об'єкта.

|  |
| --- |
| class Person:  def \_\_init\_\_(self, name, age):  self.name = name # Ініціалізація атрибута name  self.age = age # Ініціалізація атрибута age    # Створення об'єкта класу Person  person1 = Person("Alice", 30)    # Доступ до атрибутів об'єкта  print(person1.name) # Виведе: Alice  print(person1.age) # Виведе: 30 |

Метод \_\_str\_\_(self): Метод \_\_str\_\_ відповідає за текстове представлення об'єкта. Він викликається, коли ви намагаєтеся вивести об'єкт за допомогою функції print() або при використанні функції str() на об'єкті. Це дозволяє створити зручний, людський формат представлення об'єкта, що може бути корисно для відлагодження, логування або просто для виведення коректної інформації.

|  |
| --- |
| class Person:  def \_\_init\_\_(self, name, age):  self.name = name  self.age = age    def \_\_str\_\_(self):  return f"Person(name={self.name}, age={self.age})"    # Створення об'єкта класу Person  person1 = Person("Alice", 30)    # Виведення об'єкта на екран  print(person1) # Виведе: Person(name=Alice, age=30) |

1. 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:

Цей код демонструє створення і використання класу Student, який має два атрибути: ім'я та вік студента. Конструктор класу \_\_init\_\_ ініціалізує ці атрибути при створенні нового об'єкта. Для того, щоб зручно вивести інформацію про студента, визначений метод \_\_str\_\_, який повертає рядок у форматі "Name: {name}, age: {age}". Далі створюється список об'єктів класу Student, в якому зберігаються різні студенти з різними іменами та віками. Для сортування студентів за іменем використовується стандартна функція sorted(), яка приймає список студентів і лямбда-функцію як ключ для сортування (в даному випадку за іменем кожного студента). Лямбда-функція lambda student: student.name витягує ім'я студента, і саме за ним відбувається сортування. Після сортування за алфавітним порядком, використовується цикл for, щоб пройти по відсортованому списку і вивести кожного студента за допомогою методу \_\_str\_\_, який автоматично викликається при використанні print(). Завдяки цьому кожен студент виводиться у зручному вигляді, що містить його ім'я та вік.

Текст програми:

class Student:

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

self.name = name

self.age = age

def \_\_str\_\_(self):

return f"Name: {self.name}, age: {self.age})"

students = [

Student("Tim", 20),

Student("David", 13),

Student("Vlada", 11),

Student("Olena", 54)

]

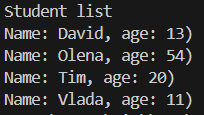
sorted\_students = sorted(students, key=lambda student: student.name)

print("Student list")

for student in sorted\_students:

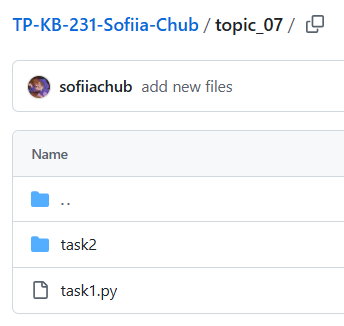
print(student)

Запущений код:



Посилання на github: <https://github.com/sofiiachub/TP-KB-231-Sofiia-Chub/blob/main/topic_07/task1.py>

Знімок екрану з посилання на github:



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

Хід виконання завдання:

Calc2.py:

У файлі Calc2.py реалізується клас Calculator, який здійснює основну роботу калькулятора. При ініціалізації класу в конструкторі \_\_init\_\_ створюється файл для зберігання історії обчислень під ім'ям "fileActions2". Клас містить два основні методи:

* addHistory: Цей метод додає запис про виконану операцію у файл історії. Він записує рядок, що містить перше число, операцію, друге число та результат обчислення.
* start: Цей метод запускає цикл, в якому калькулятор постійно запитує у користувача два числа та операцію. Зібрані дані передаються до функції myoperation, що виконує відповідну операцію (додавання, віднімання, множення, ділення). Якщо введені коректні значення, результат виводиться на екран і записується в історію. Програма запитує, чи хоче користувач продовжити обчислення, і якщо ні, завершується з повідомленням про те, що історія записана.

Operations2.py:

Цей файл містить дві функції:

* myinput: Запитує користувача ввести два числа і вибрати математичну операцію. Якщо введені значення не є числовими, функція обробляє помилку за допомогою try-except і повертає None для чисел та операції.
* myoperation: Приймає два числа, операцію і математичні функції (додавання, віднімання, множення, ділення). В залежності від введеної операції (якщо це один з символів +, -, \*, /), викликається відповідна функція з functions2.py. Якщо операція невірна або сталося ділення на нуль, повертається повідомлення про помилку.

Functions2.py:

У файлі functions2.py знаходяться функції для виконання математичних операцій:

* pluss: Додає два числа.
* subtract: Віднімає друге число від першого.
* multiply: Множить два числа.
* divide: Діє на два числа, і якщо друге число дорівнює нулю, повертає помилку про ділення на нуль.

Програма працює наступним чином:

При запуску програми створюється об'єкт класу Calculator і викликається метод start(), який ініціює цикл взаємодії з користувачем. У циклі користувач вводить два числа та обирає операцію. Введені значення передаються до функції myoperation, яка виконує відповідну математичну операцію. Результат обчислення виводиться на екран, і одночасно записується в файл fileActions2. Програма запитує, чи хоче користувач продовжити. Якщо відповідь "Ні", цикл завершується і виводиться повідомлення про успішне збереження історії в файл.

Текст програми:

Calc2.py:

from functions2 import pluss, subtract, multiply, divide

from operations2 import myinput, myoperation

class Calculator:

def \_\_init\_\_(self):

self.file = "fileActions2"

def addHistory(self, num1, num2, operationn, result):

with open(self.file, "a") as file:

file.write(f"{num1} {operationn} {num2} = {result}\n")

def start(self):

while True:

num1, num2, operationn = myinput()

if num1 is not None and num2 is not None:

result = myoperation(num1, num2, operationn, pluss, subtract, multiply, divide)

print(f"Result: {result}")

self.addHistory(num1, num2, operationn, result)

else:

print("Invalid input, please try again.")

continueWork = input("Do you want to continue? (Yes/No): ").lower()

if continueWork != "yes":

print("History is recorded")

break

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

calc = Calculator()

calc.start()

Operations2.py:

def myinput():

try:

num1 = float(input("First number: "))

num2 = float(input("Second number: "))

operationn = input("Select an operation (+, -, \*, /): ")

return num1, num2, operationn

except ValueError:

print("Error: Invalid input. Please enter valid numbers.")

return None, None, None

def myoperation(num1, num2, operationn, pluss, subtract, multiply, divide):

match operationn:

case "+":

return pluss(num1, num2)

case "-":

return subtract(num1, num2)

case "\*":

return multiply(num1, num2)

case "/":

try:

return divide(num1, num2)

except ZeroDivisionError:

return "Error: Division by zero"

case \_:

return "Incorrect operation. Please use +, -, \*, or /."

Functions2.py:

def pluss(a, b):

return a + b

def subtract(a, b):

return a - b

def multiply(a, b):

return a \* b

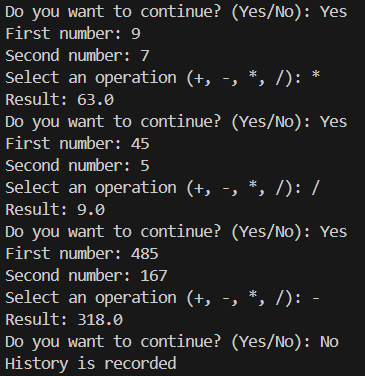
def divide(a, b):

if b == 0:

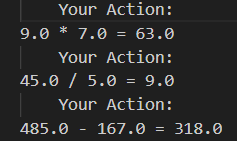
return "Error: Division by zero"

return a / b

Запущений код:

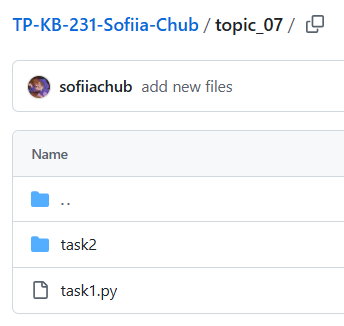


FileActions:



Посилання на github: <https://github.com/sofiiachub/TP-KB-231-Sofiia-Chub/tree/main/topic_07/task2>

Знімок екрану з посилання на github:



**Висновки:** Я пройшла та засвоїла матеріал до курсу технології програмування на мові Python. Вдалось опанувати та використати на практиці основні функції, цикли та умовні переходи. Розглянути та використати бібліотеки, класи та модулі у своїх роботах. Було розроблено програму калькулятор, який вдосконалювався з етапами вивчення мови Python.