Міністерство освіти і науки України

Національний університет «Чернігівська політехніка»

Навчально-науковий інститут електронних та інформаційних технологій

Кафедра кібербезпеки та математичного моделювання

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

виконав

студент групи КБ-231

Гладченко Олег Геннадійович

перевірив

Дюба І.М

2024 р

Звіт до Теми №1

Функції та змінні

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

**Обернути рядок в зворотному порядку.**

Хід виконання роботи:

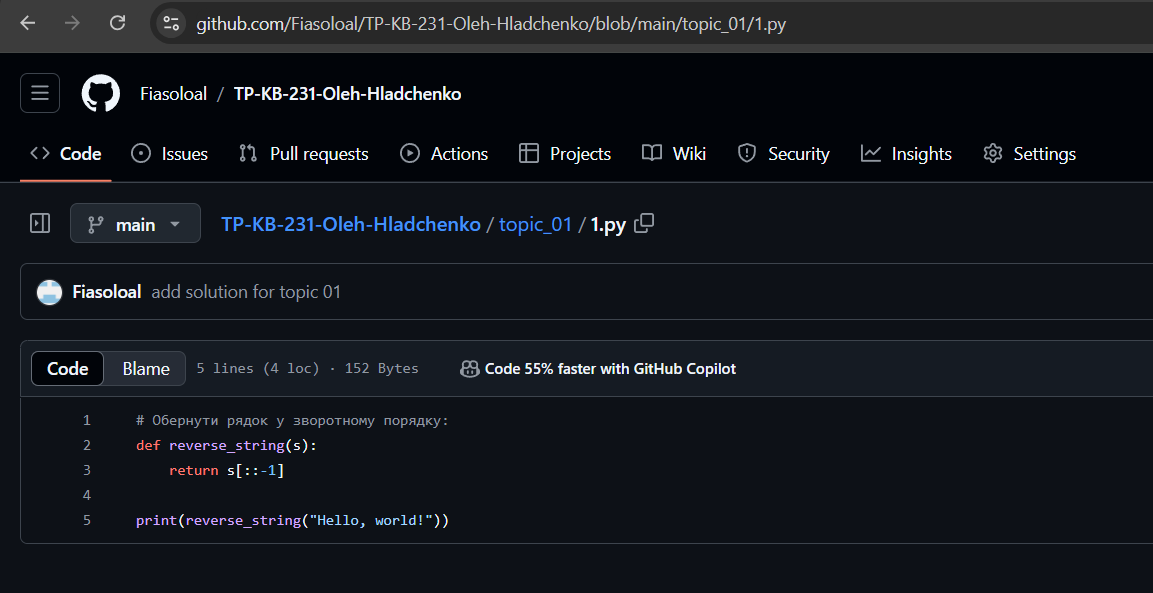
Для реалізації функції, яка обертає рядок, використовується зріз (s[::-1]). Цей метод дозволяє швидко отримати новий рядок, що містить символи оригінального рядка в зворотному порядку.

Текст програми :

|  |
| --- |
| # Обернути рядок у зворотному порядку:  def reverse\_string(s):      return s[::-1]  print(reverse\_string("Hello, world!")) |

Посилання на github: <https://github.com/Fiasoloal/TP-KB-231-Oleh-Hladchenko/blob/main/topic_01/1.py>

Знімок екрану з посилання на github:

****

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

Хід виконання роботи:

Було проведено тестування стандартних функцій Python для обробки рядків:

strip() видаляє зайві пробіли з початку і кінця рядка.

capitalize() перетворює першу літеру рядка у велику.

title() перетворює кожне слово в рядку на "титульний" формат (з великої літери).

upper() переводить всі літери рядка у верхній регістр.

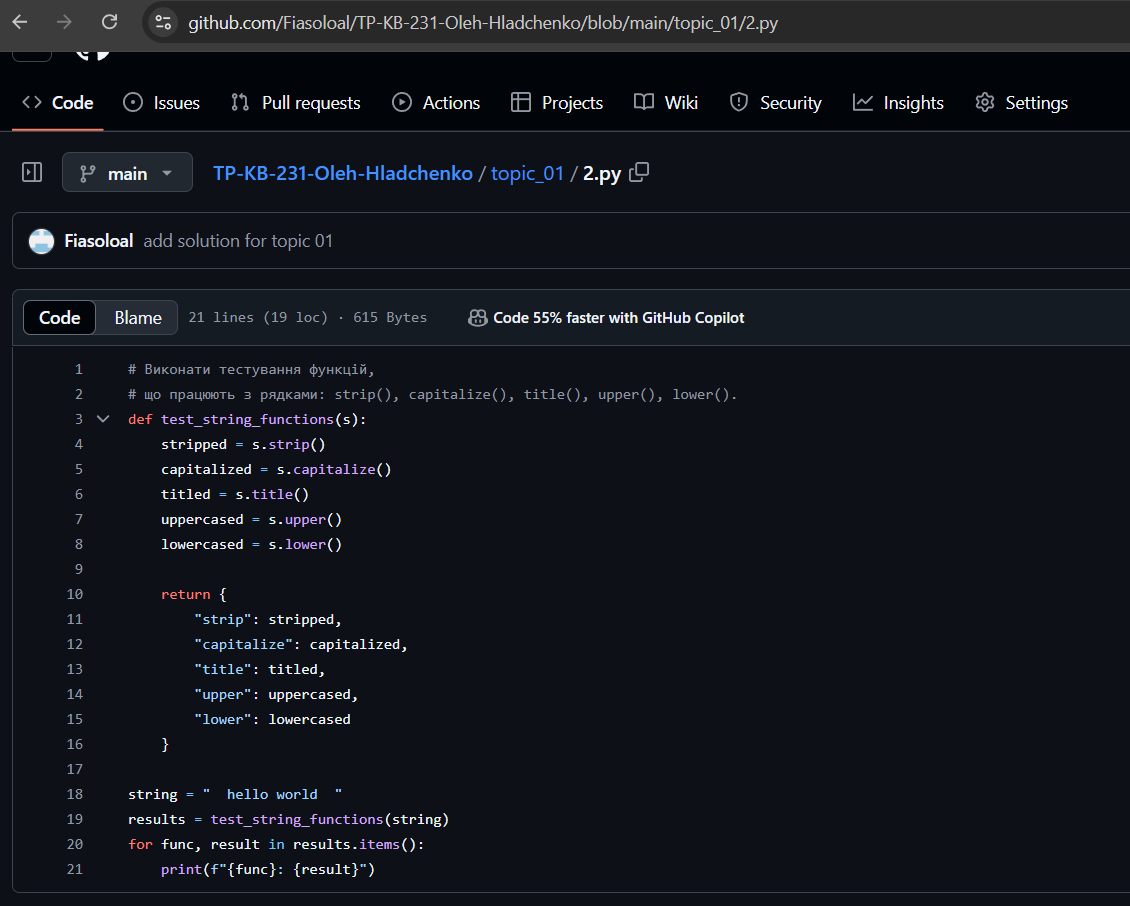
lower() переводить всі літери у нижній регістр.

Текст програми:

|  |
| --- |
| # Виконати тестування функцій,  # що працюють з рядками: strip(), capitalize(), title(), upper(), lower().  def test\_string\_functions(s):      stripped = s.strip()      capitalized = s.capitalize()      titled = s.title()      uppercased = s.upper()      lowercased = s.lower()      return {          "strip": stripped,          "capitalize": capitalized,          "title": titled,          "upper": uppercased,          "lower": lowercased      }  string = "  hello world  "  results = test\_string\_functions(string)  for func, result in results.items():      print(f"{func}: {result}") |

Посилання на github: <https://github.com/Fiasoloal/TP-KB-231-Oleh-Hladchenko/blob/main/topic_01/2.py>

Знімок екрану з посилання на github:



**Написати функцію пошуку дискримінанту квадратного рівняння.**

Хід виконання роботи:

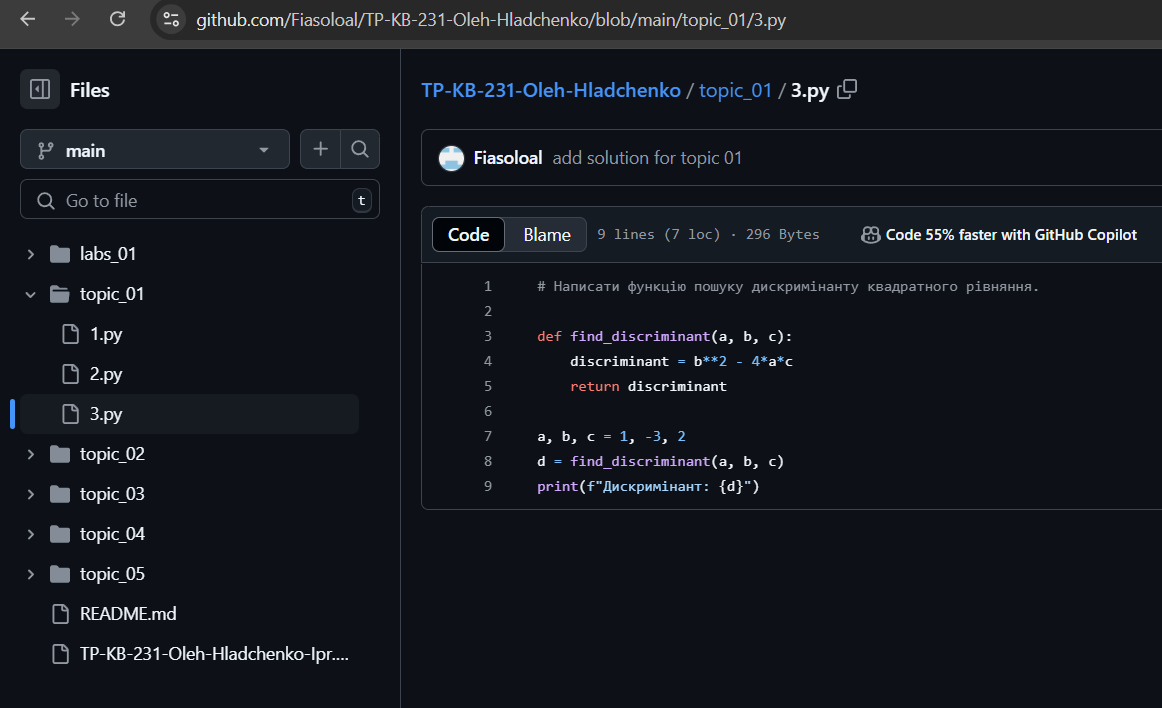
Функція **find\_discriminant(a, b, c)** реалізована згідно з формулою для знаходження дискримінанту: **D = b^2 - 4ac** . При тестуванні на коефіцієнтах a = 1, b = -3, c = 2 дискримінант виводиться коректно і дорівнює 1.

Текст програми:

|  |
| --- |
| # Написати функцію пошуку дискримінанту квадратного рівняння.  def find\_discriminant(a, b, c):      discriminant = b\*\*2 - 4\*a\*c      return discriminant  a, b, c = 1, -3, 2  d = find\_discriminant(a, b, c)  print(f"Дискримінант: {d}") |

Посилання на github: <https://github.com/Fiasoloal/TP-KB-231-Oleh-Hladchenko/blob/main/topic_01/3.py>

Знімок екрану з посилання на github:

****

Звіт до Теми №2

Функції та змінні

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 роботи:

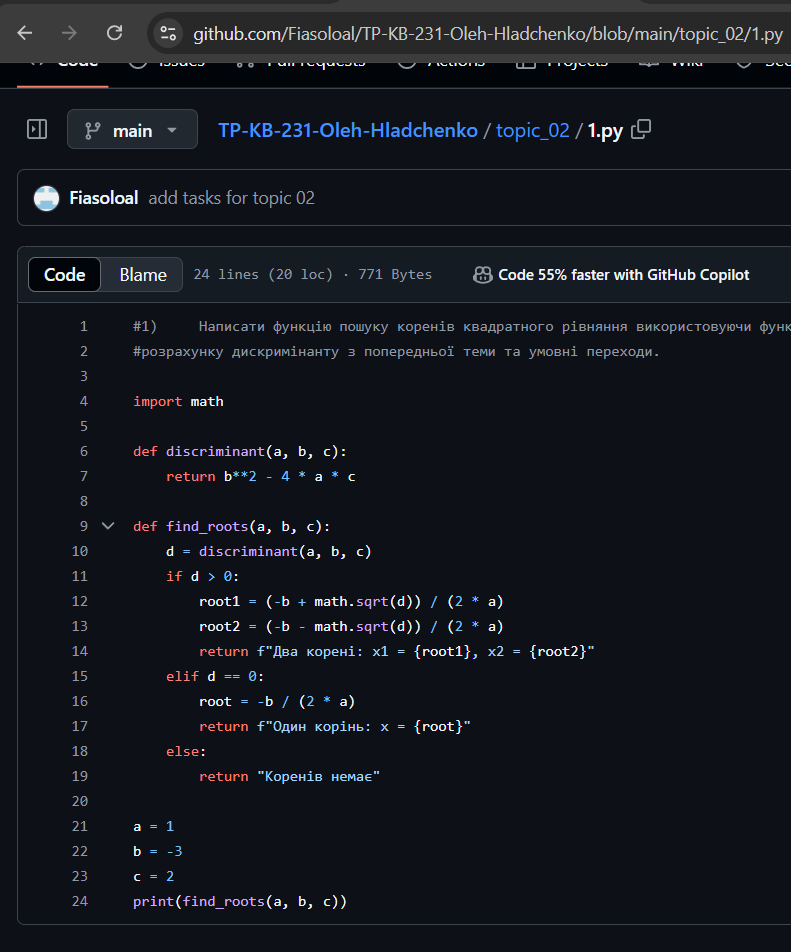
Квадратне рівняння: **ax^2 + bx + c = 0**. Щоб знайти корені цього рівняння, нам потрібно спочатку обчислити дискримінант **D = b^2 - 4ac**. Якщо дискримінант більше 0, є два корені. Якщо дискримінант дорівнює 0, є один корінь. Якщо дискримінант менше 0, коренів немає.

Текст програми:

|  |
| --- |
| #1) Написати функцію пошуку коренів квадратного рівняння використовуючи функцію  #розрахунку дискримінанту з попередньої теми та умовні переходи.  import math  def discriminant(a, b, c):      return b\*\*2 - 4 \* a \* c  def find\_roots(a, b, c):      d = discriminant(a, b, c)      if d > 0:          root1 = (-b + math.sqrt(d)) / (2 \* a)          root2 = (-b - math.sqrt(d)) / (2 \* a)          return f"Два корені: x1 = {root1}, x2 = {root2}"      elif d == 0:          root = -b / (2 \* a)          return f"Один корінь: x = {root}"      else:          return "Коренів немає"  a = 1  b = -3  c = 2  print(find\_roots(a, b, c)) |

Посилання на github: <https://github.com/Fiasoloal/TP-KB-231-Oleh-Hladchenko/blob/main/topic_02/1.py>

Знімок екрану з посилання на github:



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

Хід виконання роботи:

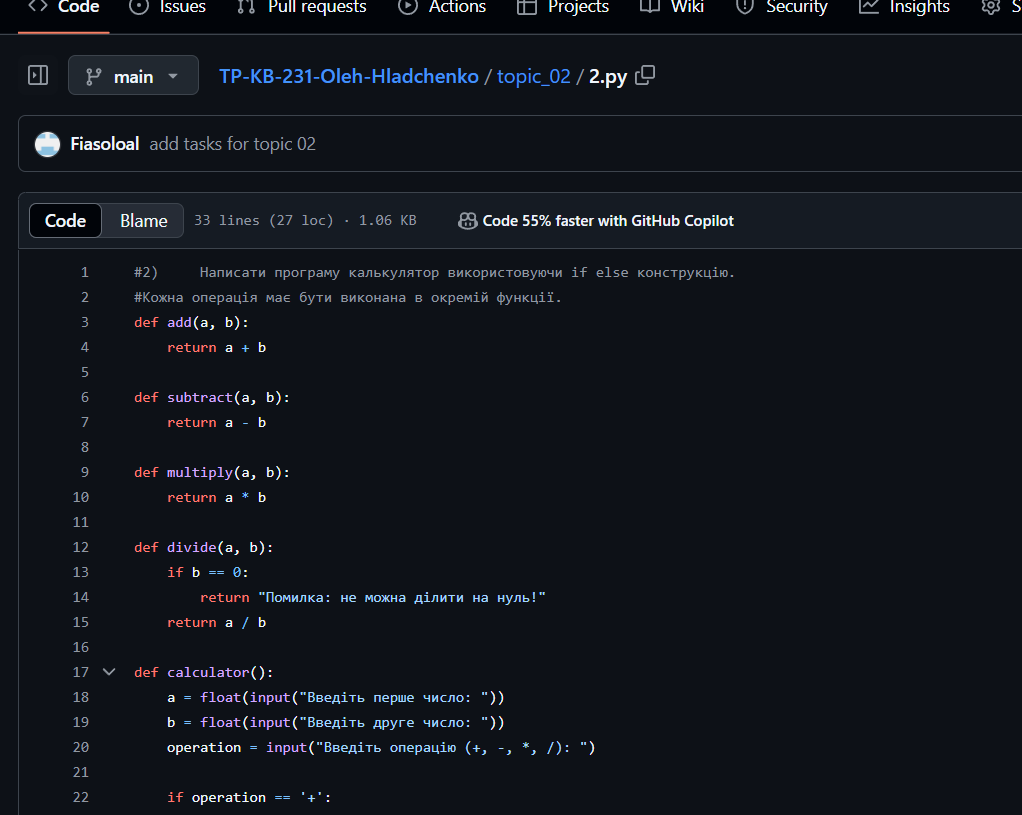
Калькулятор з використанням if-else. Ця програма дозволяє додавати, віднімати, множити і ділити два числа. Використовуючи умови для перевірки, яку операцію користувач хоче виконати.

Текст програми:

|  |
| --- |
| #2) Написати програму калькулятор використовуючи if else конструкцію.  #Кожна операція має бути виконана в окремій функції.  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 "Помилка: не можна ділити на нуль!"      return a / b  def calculator():      a = float(input("Введіть перше число: "))      b = float(input("Введіть друге число: "))      operation = input("Введіть операцію (+, -, \*, /): ")      if operation == '+':          print(f"Результат: {add(a, b)}")      elif operation == '-':          print(f"Результат: {subtract(a, b)}")      elif operation == '\*':          print(f"Результат: {multiply(a, b)}")      elif operation == '/':          print(f"Результат: {divide(a, b)}")      else:          print("Помилка: невідома операція")  calculator() |

Посилання на github: <https://github.com/Fiasoloal/TP-KB-231-Oleh-Hladchenko/blob/main/topic_02/2.py>

Знімок екрану з посилання на github:



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

Хід виконання роботи:

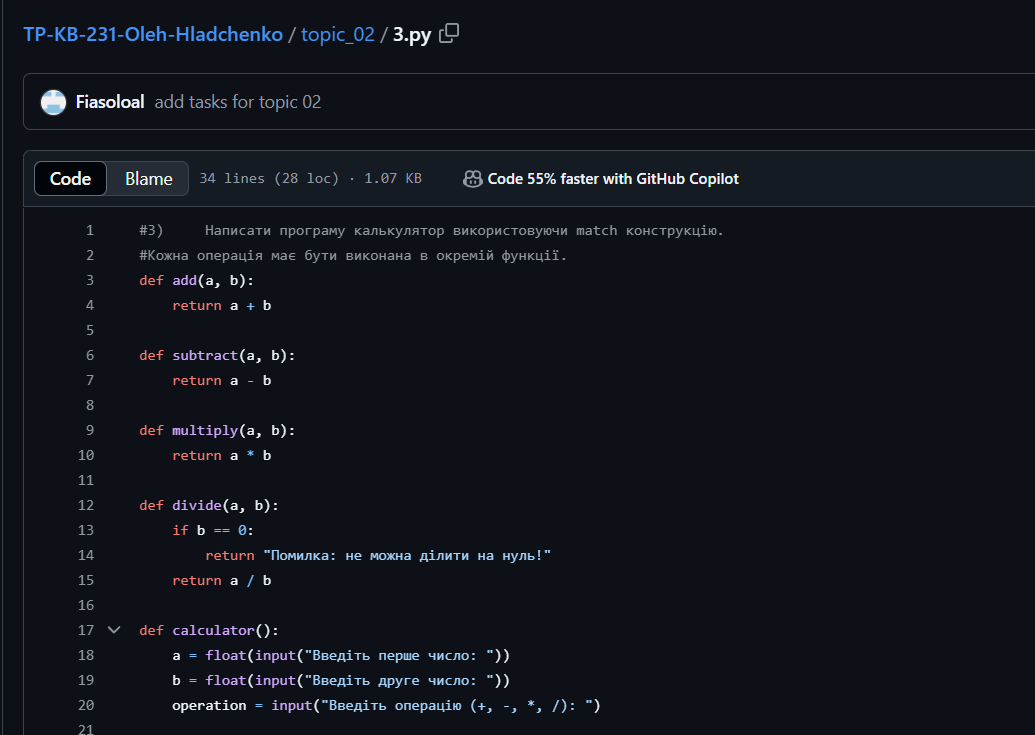
Калькулятор з використанням match. Ця програма дозволяє додавати, віднімати, множити і ділити два числа. Використовуючи операцію яка викликає окрему функцію для обчислень.

Текст програми:

|  |
| --- |
| #3) Написати програму калькулятор використовуючи match конструкцію.  #Кожна операція має бути виконана в окремій функції.  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 "Помилка: не можна ділити на нуль!"      return a / b  def calculator():      a = float(input("Введіть перше число: "))      b = float(input("Введіть друге число: "))      operation = input("Введіть операцію (+, -, \*, /): ")      match operation:          case '+':              print(f"Результат: {add(a, b)}")          case '-':              print(f"Результат: {subtract(a, b)}")          case '\*':              print(f"Результат: {multiply(a, b)}")          case '/':              print(f"Результат: {divide(a, b)}")          case \_:              print("Помилка: невідома операція")  calculator() |

Посилання на github: <https://github.com/Fiasoloal/TP-KB-231-Oleh-Hladchenko/blob/main/topic_02/3.py>

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:

Хід виконання роботи:

**Програма калькулятор з постійними запитами**

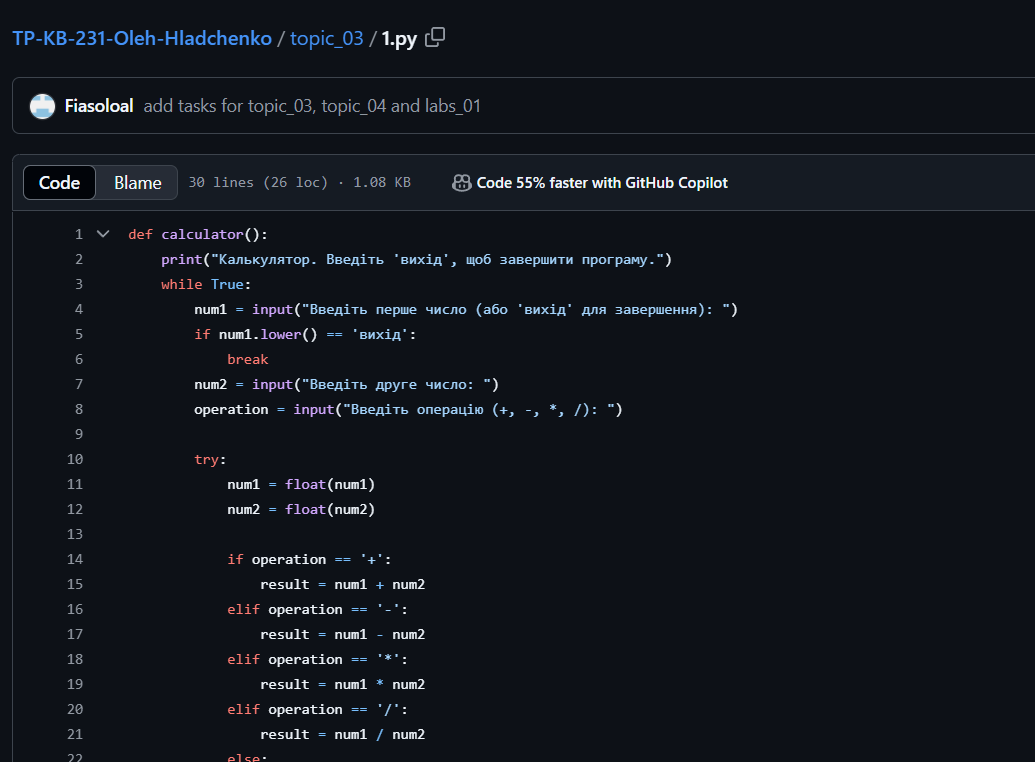
Програма калькулятора, що працює в нескінченному циклі та виконує основні арифметичні операції: додавання, віднімання, множення та ділення. Користувач може ввести дві змінні та вибрати операцію для виконання. В програмі також реалізовано механізм завершення роботи при введенні ключового слова "вихід".

Текст програми :

|  |
| --- |
| #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.  def calculator():      print("Калькулятор. Введіть 'вихід', щоб завершити програму.")      while True:          num1 = input("Введіть перше число (або 'вихід' для завершення): ")          if num1.lower() == 'вихід':              break          num2 = input("Введіть друге число: ")          operation = input("Введіть операцію (+, -, \*, /): ")          try:              num1 = float(num1)              num2 = float(num2)              if operation == '+':                  result = num1 + num2              elif operation == '-':                  result = num1 - num2              elif operation == '\*':                  result = num1 \* num2              elif operation == '/':                  result = num1 / num2              else:                  print("Неправильна операція.")                  continue              print(f"Результат: {result}")          except ValueError:              print("Будь ласка, введіть правильні числа.")  calculator() |

Посилання на github: <https://github.com/Fiasoloal/TP-KB-231-Oleh-Hladchenko/blob/main/topic_03/1.py>

Знімок екрану з посилання на github:



**Програма тестування функцій списків**

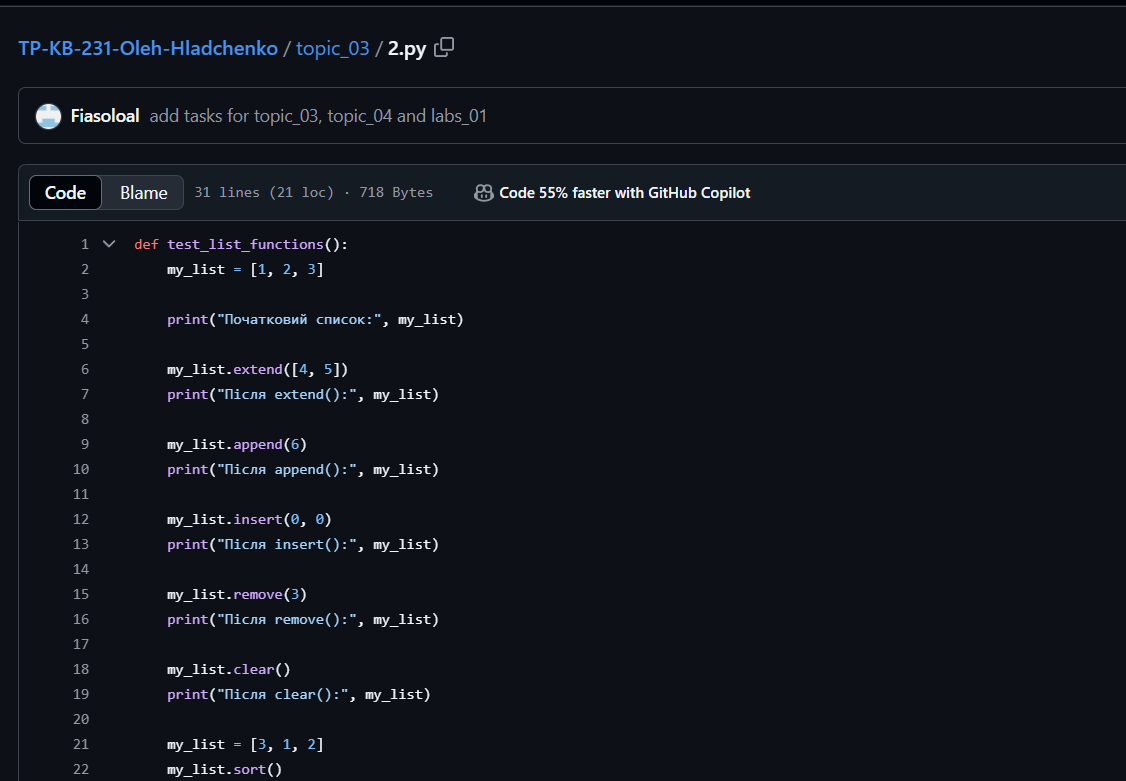
Ця програма перевіряє різні методи роботи зі списками, такі як extend(), append(), insert(), remove(), clear(), sort(), reverse(), copy(). Кожен метод застосовується до списку, а результати виводяться на екран.

Текст програми:

|  |
| --- |
| #Написати програму тестування  #функцій списків таких як: extend(), append(), insert(id, val), remove(val), clear(), sort(), reverse(), copy()  def test\_list\_functions():      my\_list = [1, 2, 3]      print("Початковий список:", my\_list)      my\_list.extend([4, 5])      print("Після extend():", my\_list)      my\_list.append(6)      print("Після append():", my\_list)      my\_list.insert(0, 0)      print("Після insert():", my\_list)      my\_list.remove(3)      print("Після remove():", my\_list)      my\_list.clear()      print("Після clear():", my\_list)      my\_list = [3, 1, 2]      my\_list.sort()      print("Після sort():", my\_list)      my\_list.reverse()      print("Після reverse():", my\_list)      copied\_list = my\_list.copy()      print("Скопійований список:", copied\_list)  test\_list\_functions() |

Посилання на github: <https://github.com/Fiasoloal/TP-KB-231-Oleh-Hladchenko/blob/main/topic_03/2.py>

Знімок екрану з посилання на github:



**Програма тестування функцій словників**

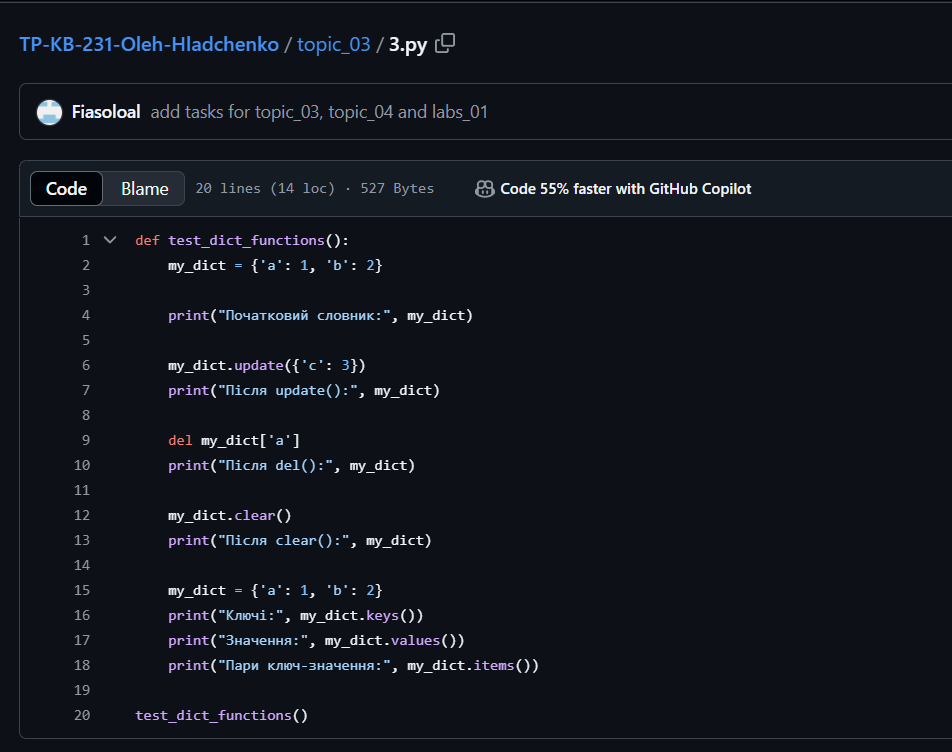
Програма тестує методи для роботи зі словниками, включаючи update(), del, clear(), keys(), values(), та items(). Використовується базовий словник, і кожна функція викликається окремо з виведенням результатів.

Текст програми:

|  |
| --- |
| #Написати програму тестування функцій словників таких як: update(), del(), clear(), keys(), values(), items()  def test\_dict\_functions():      my\_dict = {'a': 1, 'b': 2}      print("Початковий словник:", my\_dict)      my\_dict.update({'c': 3})      print("Після update():", my\_dict)      del my\_dict['a']      print("Після del():", my\_dict)      my\_dict.clear()      print("Після clear():", my\_dict)      my\_dict = {'a': 1, 'b': 2}      print("Ключі:", my\_dict.keys())      print("Значення:", my\_dict.values())      print("Пари ключ-значення:", my\_dict.items())  test\_dict\_functions() |

Посилання на github: <https://github.com/Fiasoloal/TP-KB-231-Oleh-Hladchenko/blob/main/topic_03/3.py>

Знімок екрану з посилання на github:



**Пошук позиції для вставки елементу в відсортований список**

Функція приймає відсортований список та новий елемент і знаходить позицію, куди слід вставити цей елемент для збереження сортування.

Текст програми:

|  |
| --- |
| #Маючи відсортований список, написати функцію пошуку позиції для вставки нового елементу в список.  def find\_insert\_position(sorted\_list, new\_element):      for index, element in enumerate(sorted\_list):          if new\_element < element:              return index      return len(sorted\_list)  sorted\_list = [1, 3, 5, 7]  new\_element = 4  position = find\_insert\_position(sorted\_list, new\_element)  print(f"Позиція для вставки {new\_element} в {sorted\_list}: {position}") |

Посилання на github: <https://github.com/Fiasoloal/TP-KB-231-Oleh-Hladchenko/blob/main/topic_03/4.py>

Знімок екрану з посилання на github:



Звіт до Теми №4

[Виняткові ситуації](https://eln.stu.cn.ua/course/section.php?id=98204)

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

Хід виконання роботи:

**Попередні умови**: реалізована програма калькулятор, що використовує метод нескінченного введення даних для обробки. Всі дії (додавання, віднімання, множення, ділення) реалізовані як окремі функції та використовуються у відповідних місцях.

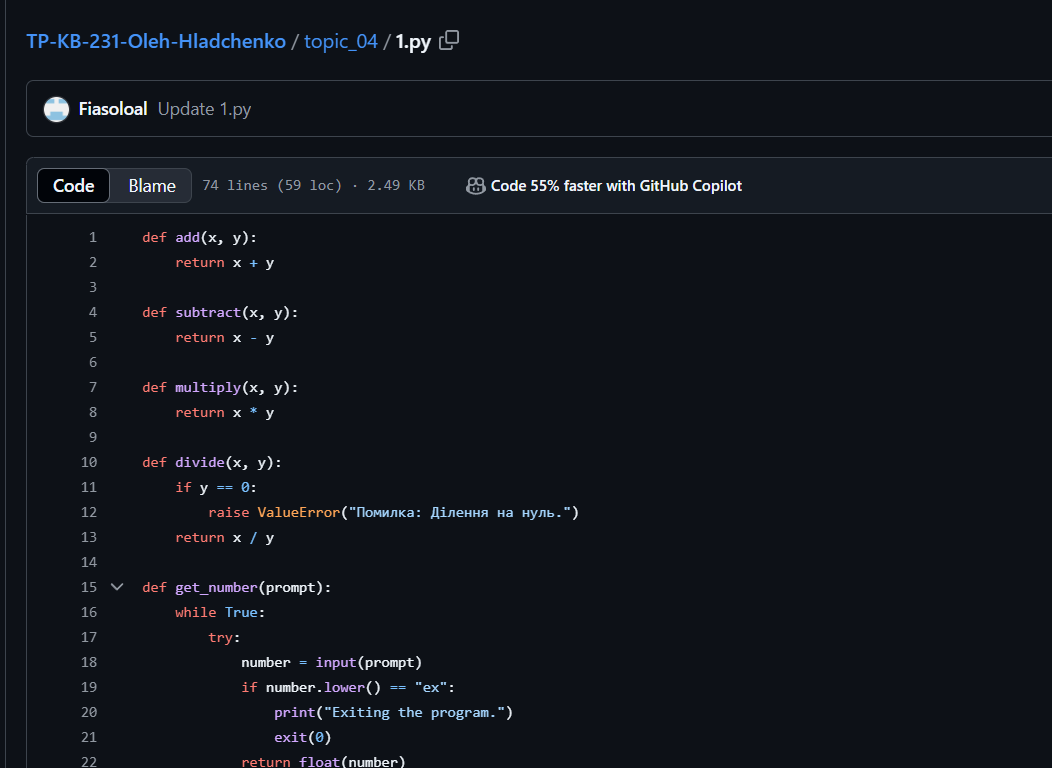
1. Розширити програму калькулятор функцією запитів даних для виконання операцій від користувача, що обробляє виняткові ситуації.
2. Розширити функцію ділення обробкою виняткової ситуації ділення но нуль

Текст програми :

|  |
| --- |
| def add(x, y):  return x + y  def subtract(x, y):  return x - y  def multiply(x, y):  return x \* y  def divide(x, y):  if y == 0:  raise ValueError("Помилка: Ділення на нуль.")  return x / y  def get\_number(prompt):  while True:  try:  number = input(prompt)  if number.lower() == "ex":  print("Exiting the program.")  exit(0)  return float(number)  except ValueError:  print("Помилка: Введіть правильне число.")  def calculator():  print("Калькулятор. Введіть 'ex' для завершення програми.")    result = get\_number("Введіть перше число: ")  while True:  print("\nОберіть операцію:\n"  "'+' — додавання, '-' — віднімання, '\*' — множення, '/' — ділення\n"  "'!-' — віднімання (з перестановкою), '!/' — ділення (з перестановкою)\n"  "'^' — піднесення до степеня, '!^' — піднесення до степеня (з перестановкою)\n"  "'ex' — для виходу з програми")  operation = input("Введіть операцію: ").lower()    if operation == 'ex':  print(f"Ваш поточний результат = {result}")  print("Exiting the program.")  exit(0)  num = get\_number("Введіть наступне число: ")  try:  if operation == '+':  result = add(result, num)  elif operation == '-':  result = subtract(result, num)  elif operation == '!-':  result = subtract(num, result)  elif operation == '\*':  result = multiply(result, num)  elif operation == '/':  result = divide(result, num)  elif operation == '!/':  result = divide(num, result)  elif operation == '^':  result \*\*= num  elif operation == '!^':  result = num \*\* result  else:  print("Помилка: Неправильна операція.")  continue  print(f"\nВаш поточний результат = {result}")  except ValueError as e:  print(e)  calculator() |

Посилання на github: <https://github.com/Fiasoloal/TP-KB-231-Oleh-Hladchenko/blob/main/topic_04/1.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:

**Написання гри**

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 роботи:

Текст програми :

|  |
| --- |
| import random  # Можливі варіанти  choices = ["камінь", "ножиці", "папір"]  # Запит вводу від користувача  user\_choice = input("Введіть ваш вибір (камінь, ножиці, папір): ").strip().lower()  # Перевіряємо коректність вводу  if user\_choice not in choices:  print("Неправильний вибір. Виберіть один із варіантів: камінь, ножиці, папір.")  else:  # Випадковий вибір комп'ютера  computer\_choice = random.choice(choices)  print(f"Комп'ютер обрав: {computer\_choice}")  # Визначаємо переможця  if user\_choice == computer\_choice:  print("Нічия!")  elif (user\_choice == "камінь" and computer\_choice == "ножиці") or \  (user\_choice == "ножиці" and computer\_choice == "папір") or \  (user\_choice == "папір" and computer\_choice == "камінь"):  print("Ви виграли!")  else:  print("Переміг комп'ютер!") |

Посилання на github: <https://github.com/Fiasoloal/TP-KB-231-Oleh-Hladchenko/blob/main/topic_05/1.py>

Знімок екрану з посилання на github:

**Написання програми**

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 роботи:

Текст програми :

|  |
| --- |
| import requests  def get\_exchange\_rate(currency):  url = "https://bank.gov.ua/NBUStatService/v1/statdirectory/exchange?json"  response = requests.get(url)  data = response.json()  for item in data:  if item['cc'] == currency:  return item['rate']  return None  def convert\_currency():  currency = input("Введіть тип валюти (EUR, USD, PLN): ").upper()  amount = float(input("Введіть кількість: "))  rate = get\_exchange\_rate(currency)  if rate:  uah = amount \* rate  print(f"{amount} {currency} = {uah:.2f} UAH")  else:  print("Невідома валюта або помилка отримання курсу.")  convert\_currency() |

Посилання на github: <https://github.com/Fiasoloal/TP-KB-231-Oleh-Hladchenko/blob/main/topic_05/2.py>

Знімок екрану з посилання на github:

**Удосконалення**

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 роботи:

Текст програми :

Calc.py

|  |
| --- |
| # calc.py  from operations import enter, actions  from functions import sum, sub, rsub, mult, div, rdiv, deg, rdeg  def calculation():  a = enter()  while True:  b = enter()  act = actions(a)  match act:  case "+":  a = sum(a, b)  case "-":  a = sub(a, b)  case "!-":  a = rsub(a, b)  case "\*":  a = mult(a, b)  case "/":  a = div(a, b)  case "!/":  a = rdiv(a, b)  case "^":  a = deg(a, b)  case "!^":  a = rdeg(a, b)  print("\nПоточний результат =", a)  # Запуск калькулятора  calculation() |

Function.py

|  |
| --- |
| # functions.py  def sum(a, b):  return a + b  def sub(a, b):  return a - b  def rsub(a, b):  return b - a  def mult(a, b):  return a \* b  def div(a, b):  if b == 0:  return "Cannot divide by 0. Please enter a different number."  return a / b  def rdiv(a, b):  if a == 0:  return "Cannot divide by 0. Result is 0. Try another action."  return b / a  def deg(a, b):  return a \*\* b  def rdeg(a, b):  return b \*\* a |

Operation.py

|  |
| --- |
| # operations.py  def enter():  while True:  a = input("Для виходу з програми введіть 'ex' \nВведіть число: ")  if a.lower() == "ex":  exit(0)  try:  return float(a)  except ValueError:  print("Неправильний ввід. Будь ласка, введіть правильне число.")  def actions(a):  print("\nКоманди:\n'+' — додати, '-' — відняти, '!-' — відняти навпаки, '\*' — множити, '/' — ділити, '!/' — ділити навпаки, '^' — піднести до степеня, '!^' — піднести до степеня навпаки\nВведіть 'ex' для виходу.")  action = input("Виберіть дію: ")  if action == "ex":  print("\nВаш поточний результат:", a)  exit(0)  elif action in ["+", "-", "!-", "\*", "/", "!/", "^", "!^"]:  return action  else:  print("Неправильна дія. Спробуйте ще раз.")  return actions(a) |

Посилання на github: <https://github.com/Fiasoloal/TP-KB-231-Oleh-Hladchenko/blob/main/topic_05/3.py>

Знімок екрану з посилання на github:

Звіт до Теми №6

[Робота з файлами](https://eln.stu.cn.ua/course/section.php?id=98206)

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

**Удосконалення коду**

Попередні умови: Реалізована програма калькулятор в файлі calc.py, до якого підключають файл functions.py та operations.py.

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

Хід виконання роботи:

Текст програми :

|  |
| --- |
| logging.basicConfig(      filename="calculator.log",      level=logging.INFO,      format="%(asctime)s - %(message)s"  )  def log\_operation(a, b, operation, result):      logging.info(f"Число 1: {a}, Число 2: {b}, Операція: {operation}, Результат: {result}") |

Посилання на github: <https://github.com/Fiasoloal/TP-KB-231-Oleh-Hladchenko/blob/main/topic_06/1.py>

Знімок екрану з посилання на github:

**Сортування списків**

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 роботи:

Текст програми :

|  |
| --- |
| students = [      {"ім'я": "Олексій", "оцінка": 85},      {"ім'я": "Марія", "оцінка": 92},      {"ім'я": "Іван", "оцінка": 78},      {"ім'я": "Анна", "оцінка": 88}  ]  # Сортування за іменем  sortedname = sorted(students, key=lambda x: x["ім'я"])  print("Сортування за іменем:")  print(sortedname)  # Сортування за оцінкою  sortedscore = sorted(students, key=lambda x: x["оцінка"])  print("Сортування за оцінкою:")  print(sortedscore) |

Посилання на github: <https://github.com/Fiasoloal/TP-KB-231-Oleh-Hladchenko/blob/main/topic_06/2.py>

Знімок екрану з посилання на github:

Звіт до Теми №7

[Об'єктно-орієнтоване програмування](https://eln.stu.cn.ua/course/section.php?id=98207)

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

**Ознайомитись з деякими особливостями класів**

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

Хід виконання роботи:

Текст програми :

|  |
| --- |
| class User:      def \_\_init\_\_(self, name, age, occupation, login, password):          self.name = name          self.age = age          self.occupation = occupation          self.login = login          self.password = password      def \_\_str\_\_(self):          return f"Ім'я: {self.name}\nВік: {self.age}\nПрофесія: {self.occupation} \nЛогін: {self.login}\nПароль: {self.password}"      def \_\_repr\_\_(self):          return f"User(name='{self.name}', age={self.age}, occupation='{self.occupation}', login='{self.login}', password='{self.password}')"  u = User("Vii", 19, "Student", "Vii$234", "12345")  print(u)  print(repr(u)) |

Посилання на github: <https://github.com/Fiasoloal/TP-KB-231-Oleh-Hladchenko/blob/main/topic_07/2.py>

Знімок екрану з посилання на github:

**Розробка класів**

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 роботи:

Текст програми :

|  |
| --- |
| class Student:      def \_\_init\_\_(self, name, age):          self.name = name          self.age = age      def \_\_str\_\_(self):          return f"{self.name}, {self.age} років"  students = [      Student("Vit", 21),      Student("Di", 19),      Student("Jonh", 22),      Student("Frid", 20)  ]  # Сортуємо список студентів за віком, використовуючи lambda функцію  sorted\_students = sorted(students, key=lambda student: student.age)  print("\nСортування за віком:")  for student in sorted\_students:      print(student)  # Сортуємо список студентів за іменем, використовуючи lambda функцію  sorted\_by\_name = sorted(students, key=lambda student: student.name)  print("\nСортування за іменем:")  for student in sorted\_by\_name:      print(student) |

Посилання на github: <https://github.com/Fiasoloal/TP-KB-231-Oleh-Hladchenko/blob/main/topic_07/3.py>

Знімок екрану з посилання на github:

**Удосконалення коду**

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

Хід виконання роботи:

Текст програми :

Calculator

|  |
| --- |
| from custom\_logger import Logger  from functions import Operations  from operations import InputHandler  class Calculator:      def \_\_init\_\_(self):          self.logger = Logger()          self.operations = Operations()          self.input\_handler = InputHandler()      def log\_operation(self, a, b, operation, result):          self.logger.log\_operation(a, b, operation, result)      def calculation(self):          a = self.input\_handler.enter()          while True:              b = self.input\_handler.enter()              act = self.input\_handler.actions(a)              match act:                  case "+":                      result = self.operations.sum(a, b)                  case "-":                      result = self.operations.sub(a, b)                  case "!-":                      result = self.operations.rsub(a, b)                  case "\*":                      result = self.operations.mult(a, b)                  case "/":                      result = self.operations.div(a, b)                  case "!/":                      result = self.operations.rdiv(a, b)                  case "^":                      result = self.operations.deg(a, b)                  case "!^":                      result = self.operations.rdeg(a, b)              self.log\_operation(a, b, act, result)              a = result              print("\nПоточний результат =", a)  if \_\_name\_\_ == "\_\_main\_\_":      calc = Calculator()      calc.calculation() |

Custom\_logger

|  |
| --- |
| import logging  class Logger:      def \_\_init\_\_(self, filename="calculator.log"):          logging.basicConfig(              filename=filename,              level=logging.INFO,              format="%(asctime)s - %(message)s"          )      def log(self, a, b, operation, result):          logging.info(f"Число 1: {a}, Число 2: {b}, Операція: {operation}, Результат: {result}") |

Functions

|  |
| --- |
| class Operations:      @staticmethod      def sum(a, b):          return a + b      @staticmethod      def sub(a, b):          return a - b      @staticmethod      def rsub(a, b):          return b - a      @staticmethod      def mult(a, b):          return a \* b      @staticmethod      def div(a, b):          if b == 0:              return "Cannot divide by 0. Please enter a different number."          return a / b      @staticmethod      def rdiv(a, b):          if a == 0:              return "Cannot divide by 0. Result is 0. Try another action."          return b / a      @staticmethod      def deg(a, b):          return a \*\* b      @staticmethod      def rdeg(a, b):          return b \*\* a |

Operations

|  |
| --- |
| class InputHandler:      @staticmethod      def enter():          while True:              a = input("Для виходу з програми введіть 'ex' \nВведіть число: ")              if a.lower() == "ex":                  exit(0)              try:                  return float(a)              except ValueError:                  print("Неправильний ввід. Будь ласка, введіть правильне число.")      @staticmethod      def actions(a):          print("\nКоманди:\n'+' — додати, '-' — відняти, '!-' — відняти навпаки, '\*' — множити, '/' — ділити, '!/' — ділити навпаки, '^' — піднести до степеня, '!^' — піднести до степеня навпаки\nВведіть 'ex' для виходу.")          action = input("Виберіть дію: ")          if action == "ex":              print("\nВаш поточний результат:", a)              exit(0)          elif action in ["+", "-", "!-", "\*", "/", "!/", "^", "!^"]:              return action          else:              print("Неправильна дія. Спробуйте ще раз.")              return InputHandler.actions(a) |

Посилання на github: <https://github.com/Fiasoloal/TP-KB-231-Oleh-Hladchenko/blob/main/topic_07/4.py>

Знімок екрану з посилання на github: