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

Звіт до Теми №1

Функції та змінні

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

**Перетворення рядка**

Необхідно рядок, що має вигляд "abcdefg123" перетворити наступним чином "321gfedcba", вважаючи сталою довжину рядку в 10 символів.

Хід виконання завдання:

Користувач вводить рядок, який зберігається у змінній original\_string. Функція reverse\_string перевертає рядок за допомогою зрізу [::-1] і повертає результат, який зберігається у змінній reversed\_string. Далі перевернутий рядок виводиться на екран.

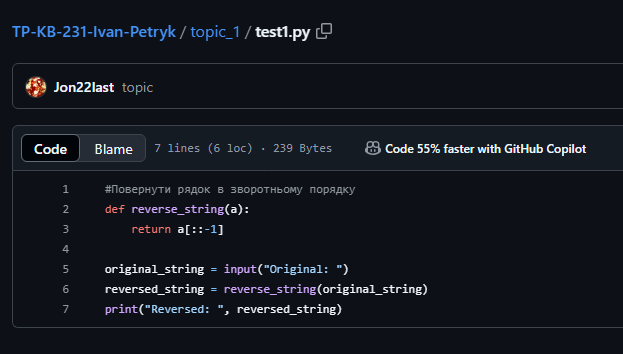
Текст програми:

|  |
| --- |
| #Повернути рядок в зворотньому порядку  def reverse\_string(a):  return a[::-1]  original\_string = input("Original: ")  reversed\_string = reverse\_string(original\_string)  print("Reversed: ", reversed\_string) |

Посилання на github:

https://github.com/Jon22last/TP-KB-231-Ivan-Petryk/blob/main/topic\_1/test1.py

Знімок екрану з посилання на github:



**Протестувати базові функції для рядків**

Протестувати базові функції для роботи з рядками.

Хід виконання завдання:

Код демонструє використання різних методів роботи з рядком: визначення його довжини, видалення зайвих пробілів, зміна регістру літер, перевірка початку та кінця рядка, заміна слів, а також розбиття рядка на слова. Результати кожної операції виводяться на екран.

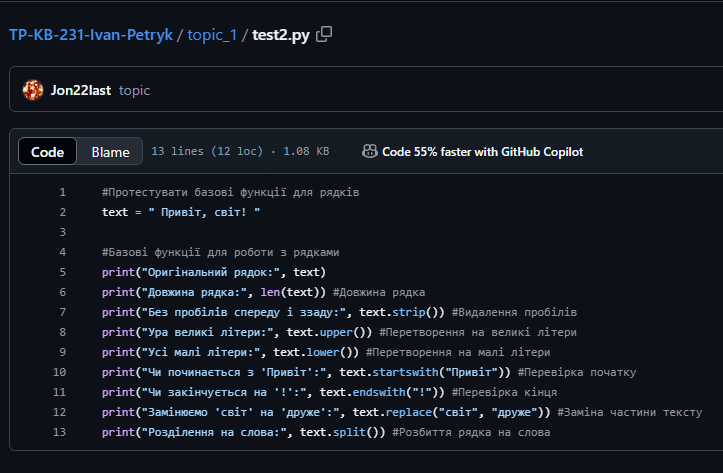
Текст програми:

|  |
| --- |
| #Протестувати базові функції для рядків  text = " Привіт, світ! "  #Базові функції для роботи з рядками  print("Оригінальний рядок:", text)  print("Довжина рядка:", len(text)) #Довжина рядка  print("Без пробілів спереду і ззаду:", text.strip()) #Видалення пробілів  print("Ура великі літери:", text.upper()) #Перетворення на великі літери  print("Усі малі літери:", text.lower()) #Перетворення на малі літери  print("Чи починається з 'Привіт':", text.startswith("Привіт")) #Перевірка початку  print("Чи закінчується на '!':", text.endswith("!")) #Перевірка кінця  print("Замінюємо 'світ' на 'друже':", text.replace("світ", "друже")) #Заміна частини тексту  print("Розділення на слова:", text.split()) #Розбиття рядка на слова |

Посилання на github:

https://github.com/Jon22last/TP-KB-231-Ivan-Petryk/blob/main/topic\_1/test2.py

Знімок екрану з посилання на github:



**Написати функцію пошуку Дескримінанту**

Обчислити дискримінант квадратного рівняння.

Хід виконання завдання:

Користувач вводить коефіцієнти ( a ), ( b ) і ( c ). На основі цих значень обчислюється дискримінант ( D ) за формулою, після чого результат виводиться на екран.

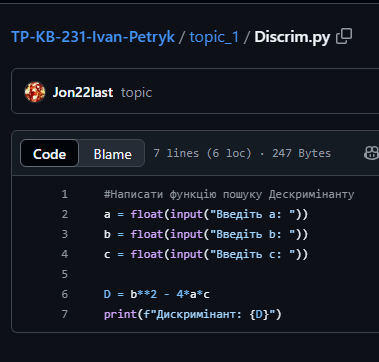
Текст програми:

|  |
| --- |
| #Написати функцію пошуку Дескримінанту  a = float(input("Введіть a: "))  b = float(input("Введіть b: "))  c = float(input("Введіть c: "))  D = b\*\*2 - 4\*a\*c  print(f"Дискримінант: {D}") |

Посилання на github:

https://github.com/Jon22last/TP-KB-231-Ivan-Petryk/blob/main/topic\_1/Discrim.py

Знімок екрану з посилання на github:



Звіт до Теми №2

Умовні переходи

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

**Виконати пошук коренів квадратного рівняння враховуючи значення дискримінанту**

Знайти корені квадратного рівняння з урахуванням значення дискримінанта.

Хід виконання завдання:

Обчислюється дискримінант. Якщо (D > 0), знаходяться два корені, якщо (D = 0) — один корінь, якщо (D < 0) — коренів немає.

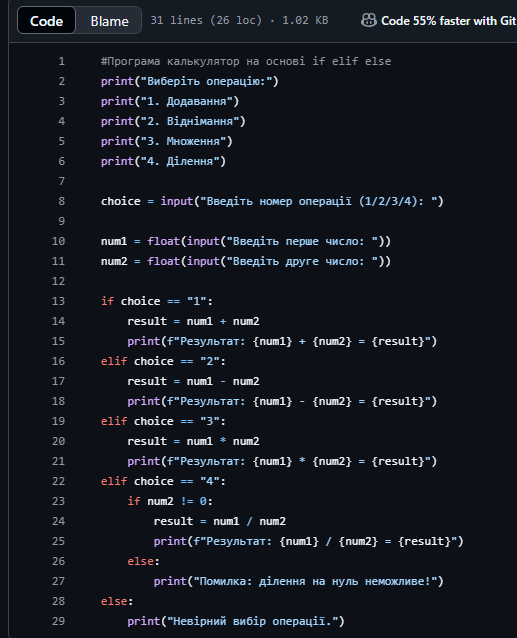
Текст програми:

|  |
| --- |
| #Виконати пошук коренів квадратного рівняння враховуючи значення дискримінанту  a = float(input("Введіть a: "))  b = float(input("Введіть b: "))  c = float(input("Введіть c: "))  D = b\*\*2-4\*a\*c  print(f"Дискримінант: {D}")  if D > 0:  x1 = (-b + D\*\*0.5) / (2\*a)  x2 = (-b - D\*\*0.5) / (2\*a)  print(f"Рівняння має два корені: x1 = {x1}, x2 = {x2}")  elif D== 0:  x = -b / (2\*a)  print(f"Рівняння має один корінь: x = {x}")  else:  print(f"Рівняння не має дійсних коренів") |

Посилання на github:

https://github.com/Jon22last/TP-KB-231-Ivan-Petryk/blob/main/topic\_2/test1.py

Знімок екрану з посилання на github:



**Програма калькулятор на основі if elif else**

Реалізувати простий калькулятор для виконання базових арифметичних операцій.

Хід виконання завдання:

Користувач обирає операцію (додавання, віднімання, множення або ділення) та вводить два числа. Залежно від вибору обчислюється результат. Якщо операція ділення і друге число дорівнює нулю, виводиться повідомлення про помилку.

Текст програми:

|  |
| --- |
| #Програма калькулятор на основі if elif else  print("Виберіть операцію:")  print("1. Додавання")  print("2. Віднімання")  print("3. Множення")  print("4. Ділення")  choice = input("Введіть номер операції (1/2/3/4): ")  num1 = float(input("Введіть перше число: "))  num2 = float(input("Введіть друге число: "))  if choice == "1":  result = num1 + num2  print(f"Результат: {num1} + {num2} = {result}")  elif choice == "2":  result = num1 - num2  print(f"Результат: {num1} - {num2} = {result}")  elif choice == "3":  result = num1 \* num2  print(f"Результат: {num1} \* {num2} = {result}")  elif choice == "4":  if num2 != 0:  result = num1 / num2  print(f"Результат: {num1} / {num2} = {result}")  else:  print("Помилка: ділення на нуль неможливе!")  else:  print("Невірний вибір операції.") |

Посилання на github:

https://github.com/Jon22last/TP-KB-231-Ivan-Petryk/blob/main/topic\_2/calculator1.py

Знімок екрану з посилання на github:



**Та сама програма калькулятор на основі оператора match**

Реалізувати калькулятор для базових арифметичних операцій за допомогою оператора match.

Хід виконання завдання:

Користувач обирає операцію (додавання, віднімання, множення або ділення) та вводить два числа. За допомогою match виконується відповідна операція. Якщо вибір некоректний або ділення на нуль, виводиться відповідне повідомлення.

Текст програми:

|  |
| --- |
| #Та сама програма калькулятор на основі оператора match  print("Виберіть операцію:")  print("1. Додавання")  print("2. Віднімання")  print("3. Множення")  print("4. Ділення")  choice = input("Введіть номер операції (1/2/3/4): ")  num1 = float(input("Введіть перше число: "))  num2 = float(input("Введіть друге число: "))  match choice:  case "1":  result = num1 + num2  print(f"Результат: {num1} + {num2} = {result}")  case "2":  result = num1 - num2  print(f"Результат: {num1} - {num2} = {result}")  case "3":  result = num1 \* num2  print(f"Результат: {num1} \* {num2} = {result}")  case "4":  if num2 != 0:  result = num1 / num2  print(f"Результат: {num1} / {num2} = {result}")  else:  print("Помилка: ділення на нуль неможливе!")  case \_:  print("Невірний вибір операції.") |

Посилання на github:

https://github.com/Jon22last/TP-KB-231-Ivan-Petryk/blob/main/topic\_2/calculator2.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:

**Написати програму калькулятор з постійними запитами на введення нових даних та операцій.**

Реалізувати калькулятор, який постійно запитує нові дані та операції до моменту виходу.

Хід виконання завдання:

Програма працює у нескінченному циклі while True. Користувач обирає операцію, вводить числа, і відповідний результат обчислюється та виводиться. Якщо обрано пункт "5", програма завершується, а при некоректному виборі операції виводиться повідомлення про помилку.

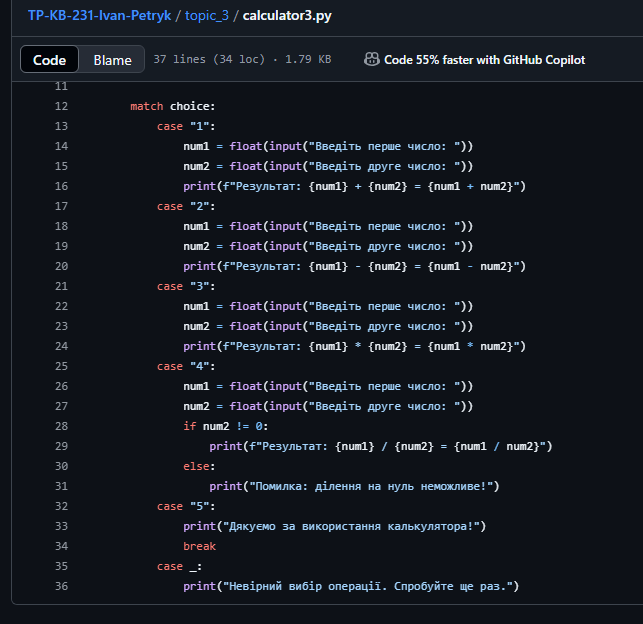
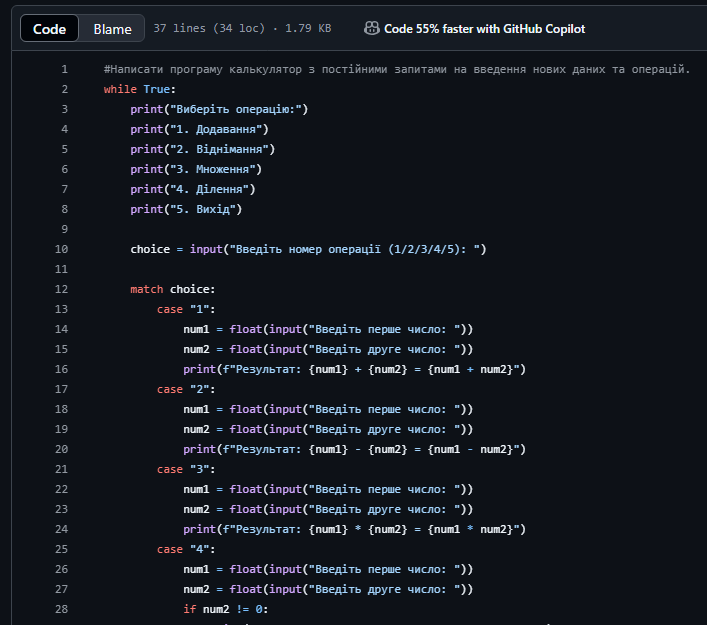
Текст програми:

|  |
| --- |
| #Написати програму калькулятор з постійними запитами на введення нових даних та операцій.  while True:  print("Виберіть операцію:")  print("1. Додавання")  print("2. Віднімання")  print("3. Множення")  print("4. Ділення")  print("5. Вихід")  choice = input("Введіть номер операції (1/2/3/4/5): ")  match choice:  case "1":  num1 = float(input("Введіть перше число: "))  num2 = float(input("Введіть друге число: "))  print(f"Результат: {num1} + {num2} = {num1 + num2}")  case "2":  num1 = float(input("Введіть перше число: "))  num2 = float(input("Введіть друге число: "))  print(f"Результат: {num1} - {num2} = {num1 - num2}")  case "3":  num1 = float(input("Введіть перше число: "))  num2 = float(input("Введіть друге число: "))  print(f"Результат: {num1} \* {num2} = {num1 \* num2}")  case "4":  num1 = float(input("Введіть перше число: "))  num2 = float(input("Введіть друге число: "))  if num2 != 0:  print(f"Результат: {num1} / {num2} = {num1 / num2}")  else:  print("Помилка: ділення на нуль неможливе!")  case "5":  print("Дякуємо за використання калькулятора!")  break  case \_:  print("Невірний вибір операції. Спробуйте ще раз.") |

Посилання на github:

https://github.com/Jon22last/TP-KB-231-Ivan-Petryk/blob/main/topic\_3/calculator3.py

Знімок екрану з посилання на github:



**Написати програму тестування функцій словників**

Протестувати базові операції зі словниками (додавання, оновлення, видалення, перевірка наявності, отримання ключів і значень).

Хід виконання завдання:

Програма створює словник з тваринами, виводить його, додає новий елемент, оновлює існуючий, видаляє ключ, перевіряє наявність ключів, а також отримує списки ключів і значень. Усі дії супроводжуються виведенням результатів.

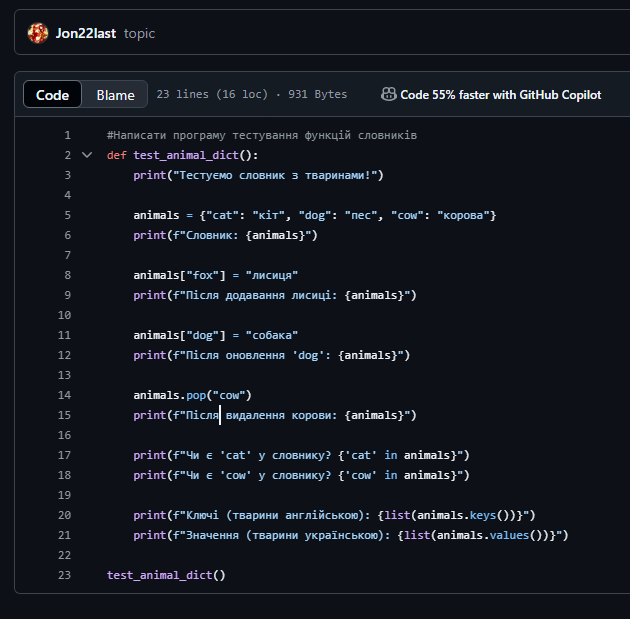
Текст програми:

|  |
| --- |
| #Написати програму тестування функцій словників  def test\_animal\_dict():  print("Тестуємо словник з тваринами!")  animals = {"cat": "кіт", "dog": "пес", "cow": "корова"}  print(f"Словник: {animals}")  animals["fox"] = "лисиця"  print(f"Після додавання лисиці: {animals}")  animals["dog"] = "собака"  print(f"Після оновлення 'dog': {animals}")  animals.pop("cow")  print(f"Після видалення корови: {animals}")  print(f"Чи є 'cat' y словнику? {'cat' in animals}")  print(f"Чи є 'cow' y словнику? {'cow' in animals}")  print(f"Ключі (тварини англійською): {list(animals.keys())}")  print(f"Значення (тварини українською): {list(animals.values())}")  test\_animal\_dict() |

Посилання на github:

https://github.com/Jon22last/TP-KB-231-Ivan-Petryk/blob/main/topic\_3/lexicon.py

Знімок екрану з посилання на github:



**Написати програму тестування функцій списків**

Протестувати основні функції роботи зі списками.

Хід виконання завдання:

Програма створює список і виконує такі операції: додавання елемента, вставка у певну позицію, видалення за значенням та індексом, перевірка наявності елемента, сортування, реверс, копіювання, та очищення списку. Усі результати операцій виводяться на екран.

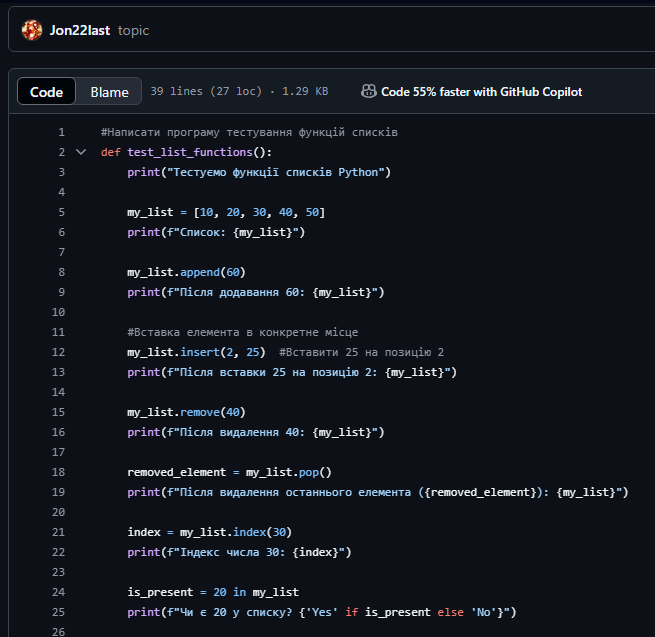
Текст програми:

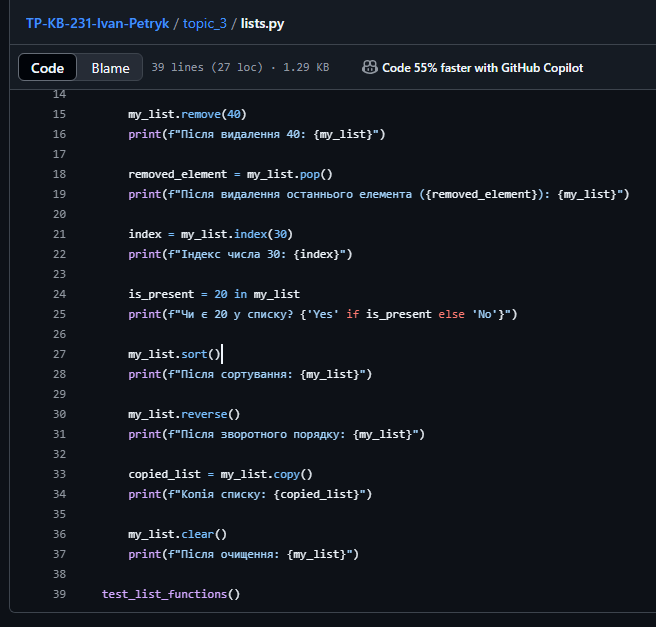
|  |
| --- |
| #Написати програму тестування функцій списків  def test\_list\_functions():  print("Тестуємо функції списків Python")  my\_list = [10, 20, 30, 40, 50]  print(f"Список: {my\_list}")  my\_list.append(60)  print(f"Після додавання 60: {my\_list}")  #Вставка елемента в конкретне місце  my\_list.insert(2, 25) #Вставити 25 на позицію 2  print(f"Після вставки 25 на позицію 2: {my\_list}")  my\_list.remove(40)  print(f"Після видалення 40: {my\_list}")  removed\_element = my\_list.pop()  print(f"Після видалення останнього елемента ({removed\_element}): {my\_list}")  index = my\_list.index(30)  print(f"Індекс числа 30: {index}")  is\_present = 20 in my\_list  print(f"Чи є 20 y списку? {'Yes' if is\_present else 'No'}")  my\_list.sort()  print(f"Після сортування: {my\_list}")  my\_list.reverse()  print(f"Після зворотного порядку: {my\_list}")  copied\_list = my\_list.copy()  print(f"Копія списку: {copied\_list}")  my\_list.clear()  print(f"Після очищення: {my\_list}")  test\_list\_functions() |

Посилання на github:

https://github.com/Jon22last/TP-KB-231-Ivan-Petryk/blob/main/topic\_3/lists.py

Знімок екрану з посилання на github:





**Написати функцію пошуку позиції для вставки нового елементу у відсортований список.**

Знайти позицію для вставки нового елемента у відсортований список, щоб зберегти порядок.

Хід виконання завдання:

Функція проходить по списку та повертає індекс першого елемента, який більший або рівний новому. Якщо новий елемент більший за всі, повертається довжина списку (вставка в кінець). У прикладі елемент 8 вставляється перед 9, тому позиція — 4.

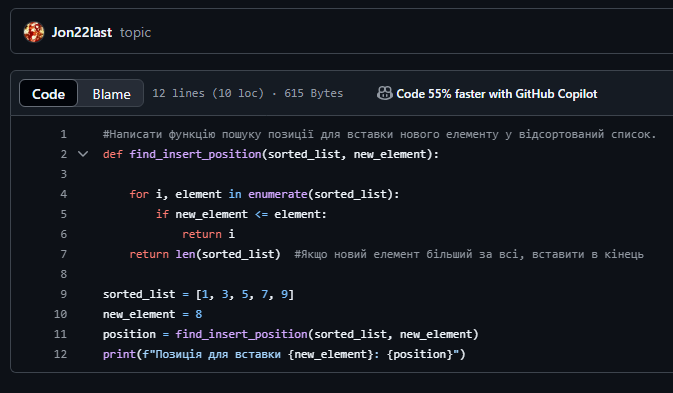
Текст програми:

|  |
| --- |
| #Написати функцію пошуку позиції для вставки нового елементу у відсортований список.  def find\_insert\_position(sorted\_list, new\_element):  for i, element in enumerate(sorted\_list):  if new\_element <= element:  return i  return len(sorted\_list) #Якщо новий елемент більший за всі, вставити в кінець  sorted\_list = [1, 3, 5, 7, 9]  new\_element = 8  position = find\_insert\_position(sorted\_list, new\_element)  print(f"Позиція для вставки {new\_element}: {position}") |

Посилання на github:

https://github.com/Jon22last/TP-KB-231-Ivan-Petryk/blob/main/topic\_3/sort\_list.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:

**Розширити програму калькулятор функцією запитів від користувача, що обробляє виняткові ситуації і розширити функцію ділення обробкою виняткової ситуації ділення но нуль**

Розширити калькулятор для обробки винятків (некоректного вводу та ділення на нуль) та додати можливість виходу користувача.

Хід виконання завдання:

Програма працює у циклі, приймає два числа та оператор. Обробляє помилки вводу (ValueError) і ділення на нуль (ZeroDivisionError). Користувач може вийти, ввівши 'q'. Результат обчислень або повідомлення про помилки виводяться на екран.

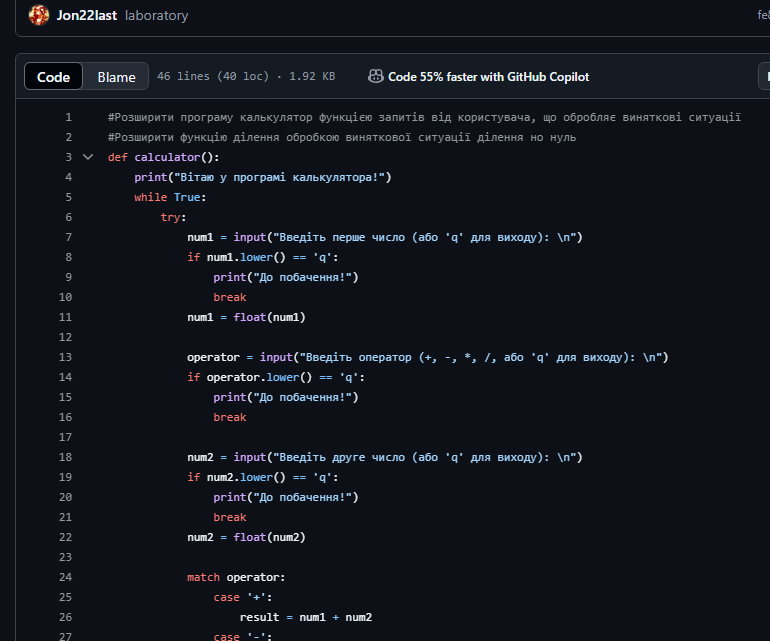
Текст програми:

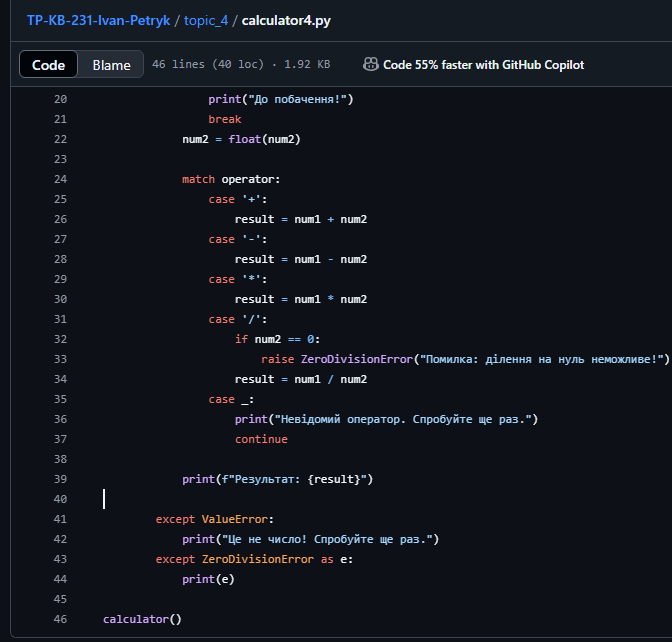
|  |
| --- |
| #Розширити програму калькулятор функцією запитів від користувача, що обробляє виняткові ситуації  #Розширити функцію ділення обробкою виняткової ситуації ділення но нуль  def calculator():  print("Вітаю у програмі калькулятора!")  while True:  try:  num1 = input("Введіть перше число (або 'q' для виходу): \n")  if num1.lower() == 'q':  print("До побачення!")  break  num1 = float(num1)  operator = input("Введіть оператор (+, -, \*, /, або 'q' для виходу): \n")  if operator.lower() == 'q':  print("До побачення!")  break  num2 = input("Введіть друге число (або 'q' для виходу): \n")  if num2.lower() == 'q':  print("До побачення!")  break  num2 = float(num2)  match operator:  case '+':  result = num1 + num2  case '-':  result = num1 - num2  case '\*':  result = num1 \* num2  case '/':  if num2 == 0:  raise ZeroDivisionError("Помилка: ділення на нуль неможливе!")  result = num1 / num2  case \_:  print("Невідомий оператор. Спробуйте ще раз.")  continue  print(f"Результат: {result}")  except ValueError:  print("Це не число! Спробуйте ще раз.")  except ZeroDivisionError as e:  print(e)  calculator() |

Посилання на github:

https://github.com/Jon22last/TP-KB-231-Ivan-Petryk/blob/main/topic\_4/calculator4.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:

**Гра Камінь Ножиці Папер**

Реалізувати гру "Камінь, Ножиці, Папір" проти комп'ютера.

Хід виконання завдання:

Гравець і комп'ютер роблять вибір. Результат (перемога, поразка чи нічия) визначається за правилами гри. Гра триває, доки гравець не введе 'q' для виходу.

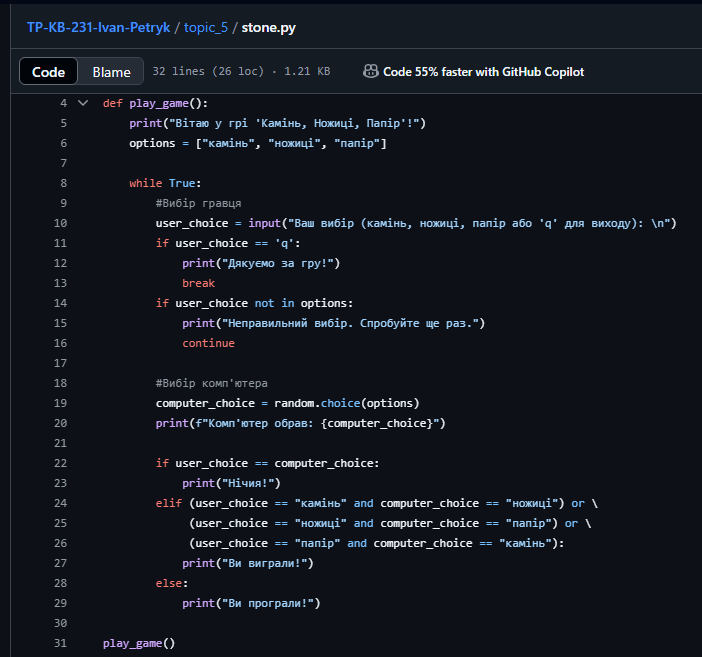
Текст програми:

|  |
| --- |
| #Гра Камінь Ножиці Папер  import random  def play\_game():  print("Вітаю у грі 'Камінь, Ножиці, Папір'!")  options = ["камінь", "ножиці", "папір"]  while True:  #Вибір гравця  user\_choice = input("Ваш вибір (камінь, ножиці, папір або 'q' для виходу): \n")  if user\_choice == 'q':  print("Дякуємо за гру!")  break  if user\_choice not in options:  print("Неправильний вибір. Спробуйте ще раз.")  continue  #Вибір комп'ютера  computer\_choice = random.choice(options)  print(f"Комп'ютер обрав: {computer\_choice}")  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("Ви програли!")  play\_game() |

Посилання на github:

https://github.com/Jon22last/TP-KB-231-Ivan-Petryk/blob/main/topic\_5/stone.py

Знімок екрану з посилання на github:



**Конвертор валют**

Реалізувати конвертор валют із актуальними курсами через API НБУ.

Хід виконання завдання:

Програма отримує курси валют, користувач вводить суму, вихідну та цільову валюти. Виконується конвертація, поки користувач не завершить роботу. Обробляються помилки API та некоректні введення.

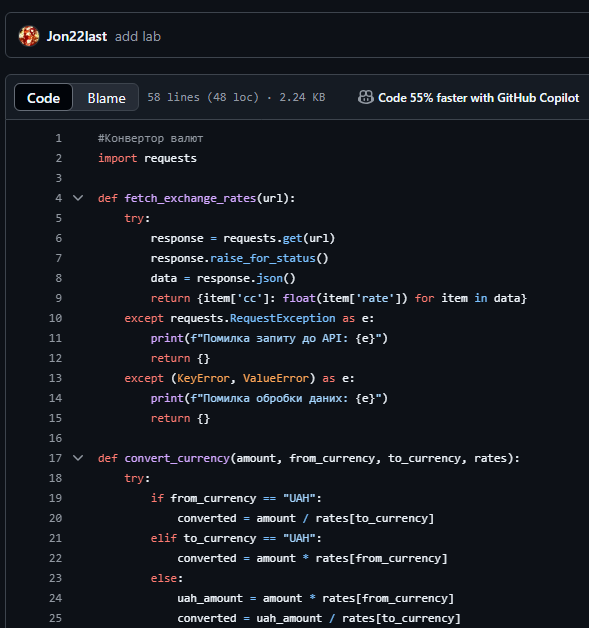
Текст програми:

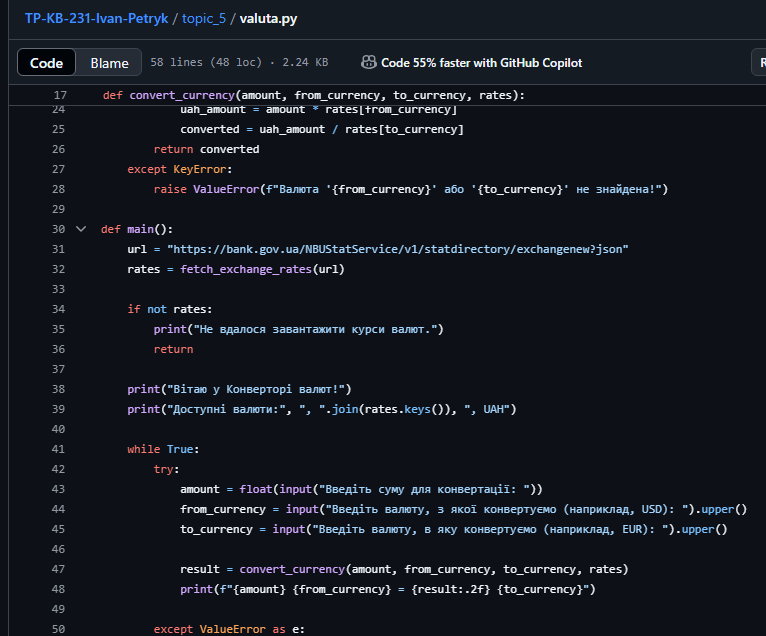
|  |
| --- |
| #Конвертор валют  import requests  def fetch\_exchange\_rates(url):  try:  response = requests.get(url)  response.raise\_for\_status()  data = response.json()  return {item['cc']: float(item['rate']) for item in data}  except requests.RequestException as e:  print(f"Помилка запиту до API: {e}")  return {}  except (KeyError, ValueError) as e:  print(f"Помилка обробки даних: {e}")  return {}  def convert\_currency(amount, from\_currency, to\_currency, rates):  try:  if from\_currency == "UAH":  converted = amount / rates[to\_currency]  elif to\_currency == "UAH":  converted = amount \* rates[from\_currency]  else:  uah\_amount = amount \* rates[from\_currency]  converted = uah\_amount / rates[to\_currency]  return converted  except KeyError:  raise ValueError(f"Валюта '{from\_currency}' або '{to\_currency}' не знайдена!")  def main():  url = "https://bank.gov.ua/NBUStatService/v1/statdirectory/exchangenew?json"  rates = fetch\_exchange\_rates(url)  if not rates:  print("Не вдалося завантажити курси валют.")  return  print("Вітаю у Конверторі валют!")  print("Доступні валюти:", ", ".join(rates.keys()), ", UAH")    while True:  try:  amount = float(input("Введіть суму для конвертації: "))  from\_currency = input("Введіть валюту, з якої конвертуємо (наприклад, USD): ").upper()  to\_currency = input("Введіть валюту, в яку конвертуємо (наприклад, EUR): ").upper()  result = convert\_currency(amount, from\_currency, to\_currency, rates)  print(f"{amount} {from\_currency} = {result:.2f} {to\_currency}")  except ValueError as e:  print(e)  cont = input("Бажаєте виконати ще одну конвертацію? (так/ні): ").lower()  if cont != "так":  print("Дякуємо за використання конвертора валют!")  break  main() |

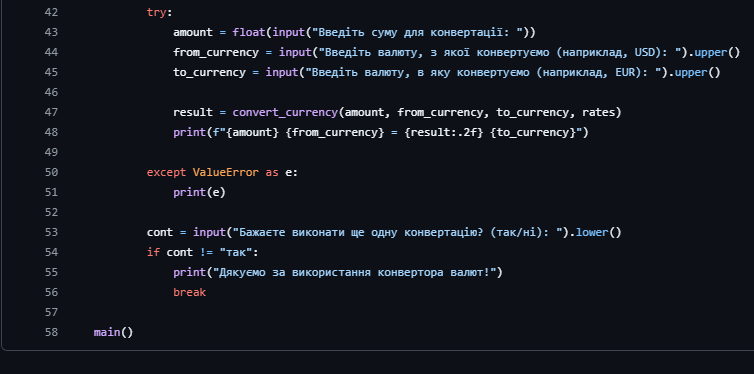
Посилання на github:

https://github.com/Jon22last/TP-KB-231-Ivan-Petryk/blob/main/topic\_5/valuta.py

Знімок екрану з посилання на github:







**Модулі для програми калькулятор**

Об'єднати арифметичні операції та перевірку вводу з модулів у калькулятор.

Хід виконання завдання:

Користувач вводить числа та оператор, які перевіряються імпортованими функціями. Виконується відповідна операція, і результат виводиться. Програма працює у циклі до введення 'ex', обробляючи помилки вводу.

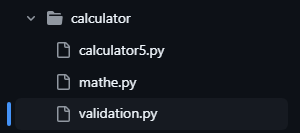
Текст програми:

|  |
| --- |
| #calculator5.py - об’єднує функціонал з інших модулів  from mathe import add, subtract, multiply, divide  from validation import validate\_number, validate\_operator  def main():  print("Вітаю у програмі 'Калькулятор'!")    while True:  try:  num1 = input("Введіть перше число (або 'ex' для виходу): ")  if num1.lower() == 'ex':  print("До побачення!")  break  num1 = validate\_number(num1)  operator = input("Введіть оператор (+, -, \*, /): ")  operator = validate\_operator(operator)  num2 = input("Введіть друге число: ")  num2 = validate\_number(num2)  match operator:  case '+':  result = add(num1, num2)  case '-':  result = subtract(num1, num2)  case '\*':  result = multiply(num1, num2)  case '/':  result = divide(num1, num2)  case \_:  raise ValueError("Невідомий оператор!")  print(f"Результат: {result}")  except ValueError as e:  print(f"Помилка: {e}")  main()  #math.py - математичні операції  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  #validation.py - перевіряє ввід користувача  def validate\_number(value):  try:  return float(value)  except ValueError:  raise ValueError("Це не число! Спробуйте ще раз.")  def validate\_operator(operator):  if operator in ['+', '-', '\*', '/']:  return operator  else:  raise ValueError("Невідомий оператор. Доступні: +, -, \*, /") |

Посилання на github:

https://github.com/Jon22last/TP-KB-231-Ivan-Petryk/tree/main/topic\_5/calculator

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:

**Використання lambda функцій для функції сортування**

Сортувати список словників за допомогою lambda за вибраним критерієм.

Хід виконання завдання:

Користувач вводить критерій (name, age, weight) або 'q' для виходу. Список сортується за допомогою lambda, результат виводиться. Некоректний критерій викликає помилку.

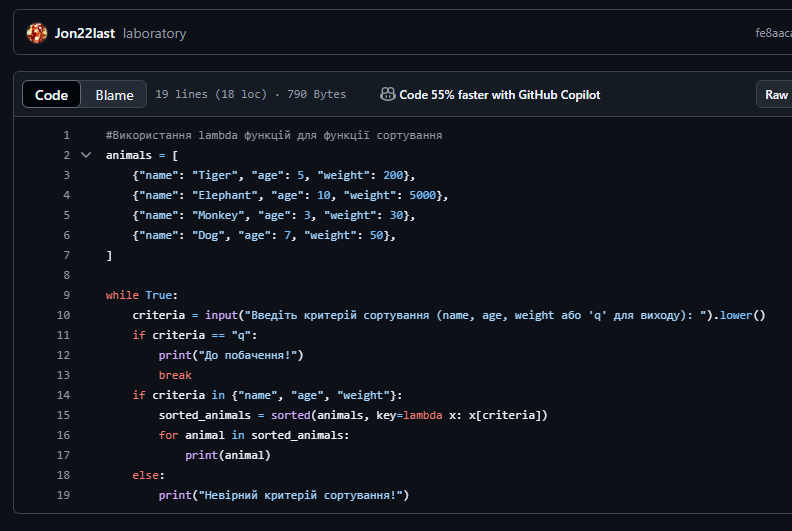
Текст програми:

|  |
| --- |
| #Використання lambda функцій для функції сортування  animals = [  {"name": "Tiger", "age": 5, "weight": 200},  {"name": "Elephant", "age": 10, "weight": 5000},  {"name": "Monkey", "age": 3, "weight": 30},  {"name": "Dog", "age": 7, "weight": 50},  ]  while True:  criteria = input("Введіть критерій сортування (name, age, weight або 'q' для виходу): ").lower()  if criteria == "q":  print("До побачення!")  break  if criteria in {"name", "age", "weight"}:  sorted\_animals = sorted(animals, key=lambda x: x[criteria])  for animal in sorted\_animals:  print(animal)  else:  print("Невірний критерій сортування!") |

Посилання на github:

https://github.com/Jon22last/TP-KB-231-Ivan-Petryk/blob/main/topic\_6/lambda.py

Знімок екрану з посилання на github:



**Логування всіх дій в застосунку Калькулятор**

Додати запис усіх операцій калькулятора у файл.

Хід виконання завдання:

Калькулятор виконує обчислення в циклі, записуючи кожну дію (операнди, оператор, результат) із позначкою часу у файл Calc\_Log.txt.

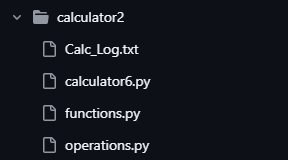
Текст програми:

|  |
| --- |
| #Логування всіх дій в застосунку Калькулятор  from functions import add, sub, mult, div, rdiv, deg  from operations import enter, actions  from datetime import datetime  import os  file\_path = os.path.join(os.path.dirname(\_\_file\_\_), "Calc\_Log.txt")  def calculation(a):  while True:  a\_old = a  b = enter()  act = actions(a)  match act:  case "+":  a = add(a, b)  case "-":  a = sub(a, b)  case "!-":  a = sub(b, a)  case "\*":  a = mult(a, b)  case "/":  a = div(a, b)  case "!/":  a = rdiv(a, b)  case "^":  a = deg(a, b)  case "!^":  a = deg(b, a)  print("\nВаш поточний результат = ", a)  with open(file\_path, "a") as file:  log\_time = datetime.now().strftime("%Y-%m-%d %H:%M:%S")  if not act:  pass  elif act in ["!-", "!/", "!^"]:  file.write(f"{log\_time} | {b} {act[1]} {a\_old} = {a}\n")  else:  file.write(f"{log\_time} | {a\_old} {act} {b} = {a}\n")  # Запуск функції калькулятора  calculation(enter())  #functions.py  def add(a, b):  return a + b  def sub(a, b):  return a - b  def mult(a, b):  return a \* b  def div(a, b):  if b == 0:  return "Ділення на нуль не допускається!"  return a / b  def rdiv(a, b):  if a == 0:  return "Не можна ділити на нуль!"  return b / a  def deg(a, b):  return a \*\* b  #operations.py  def enter():  while True:  a = input("Щоб вийти з програми, виведіть 'ex'. Введіть число: ")  if a.lower() == "ex":  exit(0)  try:  return float(a)  except ValueError:  print("Невірний вхід. Введіть дійсний номер.")  def actions(a):  print("\nДоступні дії:")  print("+ — додавання, - — віднімання, !- — віднімання з операндами, поміняними місцями")  print("\* — множення, / — ділення, !/ — ділення з операндами, поміняними місцями")  print("^ — піднесення до ступеня, !^ — зворотне піднесення до ступеня (нестандартна операція)")  print("або «ex», щоб вийти з програми")  action = input("Виберіть дію: ").strip()  if action.lower() == "ex":  print(f"Ваш поточний результат = {a}")  exit(0)  elif action in ["+", "-", "!-", "\*", "/", "!/", "^", "!^"]:  return action  else:  print("Недійсна дія. Спробуйте ще раз.")  return actions(a) |

Посилання на github:

https://github.com/Jon22last/TP-KB-231-Ivan-Petryk/tree/main/topic\_6/calculator2

Знімок екрану з посилання на github:



Звіт до Теми №7

ООП

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

**Список обєктів класу, сортування з lambda**

Сортувати список об’єктів класу Student за віком.

Хід виконання завдання:

Список об'єктів сортується за допомогою lambda, яка використовує поле age. Виводиться відсортований список.

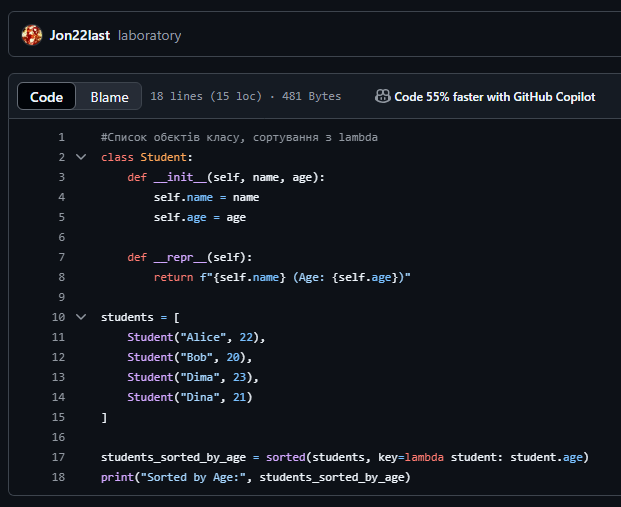
Текст програми:

|  |
| --- |
| #Список обєктів класу, сортування з lambda  class Student:  def \_\_init\_\_(self, name, age):  self.name = name  self.age = age  def \_\_repr\_\_(self):  return f"{self.name} (Age: {self.age})"  students = [  Student("Alice", 22),  Student("Bob", 20),  Student("Dima", 23),  Student("Dina", 21)  ]  students\_sorted\_by\_age = sorted(students, key=lambda student: student.age)  print("Sorted by Age:", students\_sorted\_by\_age) |

Посилання на github:

https://github.com/Jon22last/TP-KB-231-Ivan-Petryk/blob/main/topic\_7/lambda1.py

Знімок екрану з посилання на github:



**Calc with OOP**

Реалізувати калькулятор із логуванням операцій у файл.

Хід виконання завдання:

Користувач вводить числа та операцію, обчислюється результат і записується у файл Calc\_Log.txt із часом. Програма працює в циклі до команди виходу (ex).

Текст програми:

|  |
| --- |
| #Calc with OOP  import os  from datetime import datetime  from interface import get\_operation, get\_number  from operations import add, subtract, multiply, divide  file\_path = os.path.join(os.path.dirname(\_\_file\_\_), "Calc\_Log.txt")  def log\_operation(a, b, operation, result):  with open(file\_path, "a") as file:  log\_time = datetime.now().strftime("%Y-%m-%d %H:%M:%S")  file.write(f"{log\_time} | {a} {operation} {b} = {result}\n")  def calculator():  print("Ласкаво просимо до Калькулятора!")    while True:  operation = get\_operation()  if operation == "ex":  print("Дякуємо за використання калькулятора!")  break  if operation not in ["+", "-", "\*", "/"]:  print("Неправильна операція. Спробуйте ще раз.")  continue  a = get\_number("Введіть перше число: ")  b = get\_number("Введіть друге число: ")  try:  match operation:  case "+":  result = add(a, b)  case "-":  result = subtract(a, b)  case "\*":  result = multiply(a, b)  case "/":  result = divide(a, b)  print(f"Результат: {result}")  log\_operation(a, b, operation, result)  except ValueError as e:  print(f"Помилка: {e}")  #Запуск функції калькулятора  calculator()  #interface.py  def get\_operation():  """Запитує у користувача вибір операції."""  print("\nОберіть операцію: +, -, \*, / (або 'ex' для виходу)")  return input("Операція: ")  def get\_number(prompt):  """Запитує у користувача число."""  while True:  try:  return float(input(prompt))  except ValueError:  print("Це не число. Спробуйте ще раз.")  #operations.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("На нуль ділити не можна!")  return a / b |

Посилання на github:

https://github.com/Jon22last/TP-KB-231-Ivan-Petryk/tree/main/topic\_7/calculator3

Знімок екрану з посилання на github:

