**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 довільного вигляду перетворити таким чином, щоб він відтворився у зворотному напрямку.

Хід виконання завдання:

Я вирішив використати функціонал зрізу з кроком -1, щоб відтворити рядок не використовуючи циклів.

Текст програми:

|  |
| --- |
| input\_string = input("Write your line here:")  def reverse(input\_string):  return input\_string[::-1]  reversed\_string = reverse(input\_string)  print("Raw line:", input\_string)  print("Reversed line:", reversed\_string) |

**Виконати тестування функцій, що працюють з рядками**

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

Хід виконання завдання:

На прикладі одного й того самого рядка показав принцип роботи всіх функцій.

Текст програми:

|  |
| --- |
| string0 = " pYthoN is SimPLe!< "  print("strip() прибирає пробіли в рядку:", string0.strip())  print("strip() переводить перший символ рядка в верхній регістр:",string0.strip().capitalize()) # Прибрав пробіли для наглядності  print("strip() переводить кожну першу літеру слова в верхній регістр:", string0.title())  print("strip() переводить усі літери у верхній регістр:", string0.upper())  print("strip() переводить усі літери у нижній регістр:", string0.lower()) |

**Дискримінант**

Необхідно написати функцію пошуку дискримінанту квадратного рівняння.

Хід виконання завдання:

Використовуючи дві функції – одну для збору даних, іншу – для їх опрацювання дізнаємося дискримінант за заданими коефіцієнтами.

Текст програми:

|  |
| --- |
| def get\_coefficients():  print("Enter your coefficients (ax^2 + bx + c):")  a = float(input('Enter your a: '))  b = float(input('Enter your b: '))  c = float(input('Enter your c: '))  return a, b, c  def discriminant(a, b, c):  D = b\*\*2 - 4\*a\*c  return D  a, b, c = get\_coefficients()  print("The discriminant is:", discriminant(a, b, c)) |

Звіт до Теми №2

Умовні переходи

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

**Коренів квадратного рівняння**

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

Хід виконання завдання:

Використовуючи функцію розрахування дискримінанту з минулої теми, додав функцію обрахування коренів та перевірку дійсності вводу.

Текст програми:

|  |
| --- |
| import math  def get\_coefficients(): #input func  print("Enter your coefficients (ax^2 + bx + c):")  a = float(input('Enter your a: '))  b = float(input('Enter your b: '))  c = float(input('Enter your c: '))  return a, b, c  def discriminant(a, b, c): #disc calc func  return b\*\*2 - 4\*a\*c  def find\_roots(a, b, c): #root calc func  D = discriminant(a, b, c)  if D > 0:  x1 = (-b + math.sqrt(D)) / (2 \* a)  x2 = (-b - math.sqrt(D)) / (2 \* a)  return f"Two real roots: x1 = {x1}, x2 = {x2}"  elif D == 0:  x = -b / (2 \* a)  return f"One real root: x = {x}"  else:  return "No real roots exist."  a, b, c = get\_coefficients()  if a == 0: #is possible check  print("This is not a quadratic equation (a cannot be 0).")  else:  roots = find\_roots(a, b, c)  print("The roots are:", roots) |

**Калькулятор на if else конструкції**

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

Хід виконання завдання:

Для початку створив всі необхідні функції для роботи операцій калькулятора, а саме додавання, віднімання, множення, ділення, остача від ділення, піднесення для степеня та цілочислене ділення.  
Наступна функція – перевірка вибору операції.

Якщо вибір коректний – перехід до введення числових значень з вбудованою перевіркою їх коректності за допомогою функції try:

Остання частина – виконання обчислення за допомогою конструкцій if else

Текст програми:

|  |
| --- |
| # Operation func's  def add(a, b):  return a + b  def subtract(a, b):  return a - b  def multiply(a, b):  return a \* b  def divide(a, b):  if b != 0:  return a / b  else:  return "Division by zero is not allowed!"  def modulus(a, b):  return a % b  def exponent(a, b):  return a \*\* b  def floor\_division(a, b):  if b != 0:  return a // b  else:  return "Division by zero is not allowed!"  # Main function  def calculator():  print("Select an operation:")  print("1. Addition (+)")  print("2. Subtraction (-)")  print("3. Multiplication (\*)")  print("4. Division (/)")  print("5. Modulus (%)")  print("6. Exponentiation (\*\*)")  print("7. Floor Division (//)")  choice = input("Enter the number of the operation (1-7): ")  # Validate the choice of operation  if choice in ['1', '2', '3', '4', '5', '6', '7']:  try:  num1 = float(input("Enter the first number: "))  num2 = float(input("Enter the second number: "))  except ValueError:  print("Please enter valid numerical values!")  return    if choice == '1':  print(f"Result: {add(num1, num2)}")  elif choice == '2':  print(f"Result: {subtract(num1, num2)}")  elif choice == '3':  print(f"Result: {multiply(num1, num2)}")  elif choice == '4':  print(f"Result: {divide(num1, num2)}")  elif choice == '5':  print(f"Result: {modulus(num1, num2)}")  elif choice == '6':  print(f"Result: {exponent(num1, num2)}")  elif choice == '7':  print(f"Result: {floor\_division(num1, num2)}")  else:  print("Invalid operation choice!")  # Start  calculator() |

**Калькулятор на match конструкції**

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

Хід виконання завдання:

Використовуючи скелет калькулятора з минулого завдання, замінив конструкцію if else на конструкцію match

Текст програми:

|  |
| --- |
| # Operation func's  def add(a, b):  return a + b  def subtract(a, b):  return a - b  def multiply(a, b):  return a \* b  def divide(a, b):  if b != 0:  return a / b  else:  return "Division by zero is not allowed!"  def modulus(a, b):  return a % b  def exponent(a, b):  return a \*\* b  def floor\_division(a, b):  if b != 0:  return a // b  else:  return "Division by zero is not allowed!"  # Main function  def calculator():  print("Select an operation:")  print("1. Addition (+)")  print("2. Subtraction (-)")  print("3. Multiplication (\*)")  print("4. Division (/)")  print("5. Modulus (%)")  print("6. Exponentiation (\*\*)")  print("7. Floor Division (//)")  choice = input("Enter the number of the operation (1-7): ")  # Validate the choice of operation  if choice in ['1', '2', '3', '4', '5', '6', '7']:  try:  num1 = float(input("Enter the first number: "))  num2 = float(input("Enter the second number: "))  except ValueError:  print("Please enter valid numerical values!")  return    match choice:  case '1':  print(f"Result: {add(num1, num2)}")  case '2':  print(f"Result: {subtract(num1, num2)}")  case '3':  print(f"Result: {multiply(num1, num2)}")  case '4':  print(f"Result: {divide(num1, num2)}")  case '5':  print(f"Result: {modulus(num1, num2)}")  case '6':  print(f"Result: {exponent(num1, num2)}")  case '7':  print(f"Result: {floor\_division(num1, num2)}")  case \_:  print("Invalid operation choice!")  # Start  calculator() |

Звіт до Теми №3

Цикли

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

**Калькулятор з постійними запитами**

Написати програму калькулятор з постійними запитами на введення нових даних та операцій. За основу взяти програму калькулятор з попередньої теми. Реалізувати механізм завершення програми після отримання відповідної команди..

Хід виконання завдання:

Використовуючи код калькулятора з минулої теми я трохи модифікував його, використовуючи нескінченний цикл «while True:» та одну додаткову перевірку на введення команди exit, що припиняє роботу калькулятора.

Текст програми:

|  |
| --- |
| # 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:  return a / b  else:  return "Division by zero is not allowed!"  def modulus(a, b):  return a % b  def exponent(a, b):  return a \*\* b  def floor\_division(a, b):  if b != 0:  return a // b  else:  return "Division by zero is not allowed!"  # Головна функція  def calculator():  print("Welcome to the calculator!")  print("Type 'exit' at any time to quit.")    while True:  print("\nSelect an operation:")  print("1. Addition (+)")  print("2. Subtraction (-)")  print("3. Multiplication (\*)")  print("4. Division (/)")  print("5. Modulus (%)")  print("6. Exponentiation (\*\*)")  print("7. Floor Division (//)")    choice = input("Enter the number of the operation (1-7) or 'exit' to quit: ").strip().lower()    if choice == 'exit':  print("Goodbye!")  break    # Перевірка вибору операції  if choice in ['1', '2', '3', '4', '5', '6', '7']:  try:  num1 = float(input("Enter the first number: "))  num2 = float(input("Enter the second number: "))  except ValueError:  print("Invalid input! Please enter numerical values.")  continue    match choice:  case '1':  print(f"Result: {add(num1, num2)}")  case '2':  print(f"Result: {subtract(num1, num2)}")  case '3':  print(f"Result: {multiply(num1, num2)}")  case '4':  print(f"Result: {divide(num1, num2)}")  case '5':  print(f"Result: {modulus(num1, num2)}")  case '6':  print(f"Result: {exponent(num1, num2)}")  case '7':  print(f"Result: {floor\_division(num1, num2)}")  else:  print("Invalid choice! Please select a valid operation.")  # Запуск  calculator() |

**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:

Написав невеликий ланцю зміни одного й того самого списку з виведенням результату після кожної зміни.

Текст програми:

|  |
| --- |
| list0 = [5, 3, 8, 1, 2]  print("Початковий список:", list0)  list0.extend([6, 7, 9])  print("extend() додає всі елементи іншого списку до кінця:", list0)  list0.append(10)  print("append() додає один елемент до кінця:", list0)  list0.insert(2, 15)  print("insert(2, 15) вставляє 15 у позицію 2:", list0)  list0.remove(3)  print("remove(3) видаляє перше входження числа 3:", list0)  list0.clear()  print("clear() очищає список:", list0)  list0 = [5, 3, 8, 1, 2]  print("\nоновлений список для подальшого тестування:", list0)  list0.sort()  print("sort() сортує список за зростанням:", list0)  list0.reverse()  print("reverse() змінює порядок елементів на протилежний:", list0)  list\_copy = list0.copy()  print("copy() створює копію списку:", list\_copy) |

**Тестування функцій словників**

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

Хід виконання завдання:

Аналогічно до завдання 2, ланцюг перетворення одного й того самого словника.

Текст програми:

|  |
| --- |
| dict0 = {"a": 1, "b": 2, "c": 3}  print("Початковий словник:", dict0)  dict0.update({"d": 4, "a": 10})  print("update({'d': 4, 'a': 10}) додає нові пари і оновлює існуючі ключі:", dict0)  del dict0["b"]  print("del dict0['b'] видаляє пару з ключем 'b':", dict0)  dict0.clear()  print("clear() очищає словник:", dict0)  dict0 = {"x": 5, "y": 10, "z": 15}  print("\nоновлений словник для подальшого тестування:", dict0)  print("keys() повертає всі ключі:", dict0.keys())  print("values() повертає всі значення:", dict0.values())  print("items() повертає всі пари ключ-значення:", dict0.items()) |

**Функція пошуку позиції для вставки**

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

Хід виконання завдання:

Використовуючи вбудований модуль bisect написав невеличку програму що виводить поточний список і запитує нове значення, після чого вставляє його.

Текст програми:

|  |
| --- |
| import bisect  sorted\_list = [1, 3, 5, 7, 9]  print("Поточний список:", sorted\_list)  new\_value = int(input("Введіть нове значення:"))  position = bisect.bisect\_left(sorted\_list, new\_value)  print("Список з новим елементом:", sorted\_list[:position] + [new\_value] + sorted\_list[position:]) |

Звіт до Теми №4

Виняткові ситуації

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

**Модифікація програми калькулятор**

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

Розширити функцію ділення обробкою виняткової ситуації ділення но нуль

Хід виконання завдання:

Оскільки я з самого початку реалізував подібні перевірки, тож я розширив програму іншим функціоналом. Окрім вибору операції за порядковим номером, я також додав можливість вибрати її за відповідним символом.

Текст програми:

|  |
| --- |
| # Operation func's  def add(a, b):  return a + b  def subtract(a, b):  return a - b  def multiply(a, b):  return a \* b  def divide(a, b):  if b != 0:  return a / b  else:  return "Division by zero is not allowed!"  def modulus(a, b):  return a % b  def exponent(a, b):  return a \*\* b  def floor\_division(a, b):  if b != 0:  return a // b  else:  return "Division by zero is not allowed!"  # Main function  def calculator():  print("Welcome to the calculator!")  print("Type 'exit' at any time to quit.")    while True:  print("\nSelect an operation:")  print("1. Addition (+)")  print("2. Subtraction (-)")  print("3. Multiplication (\*)")  print("4. Division (/)")  print("5. Modulus (%)")  print("6. Exponentiation (\*\*)")  print("7. Floor Division (//)")    choice = input("Enter the number of the operation (1-7) or 'exit' to quit: ").strip().lower()    if choice == 'exit':  print("Goodbye!")  break    # Validate the choice of operation  if choice in ['1', '2', '3', '4', '5', '6', '7', '+','-','\*','/','%','\*\*','//',]:  try:  num1 = float(input("Enter the first number: "))  num2 = float(input("Enter the second number: "))  except ValueError:  print("Invalid input! Please enter numerical or symbolical values.")  continue    match choice:  case '1' | '+':  print(f"Result: {add(num1, num2)}")  case '2' | '-':  print(f"Result: {subtract(num1, num2)}")  case '3' | '\*':  print(f"Result: {multiply(num1, num2)}")  case '4' | '/':  print(f"Result: {divide(num1, num2)}")  case '5' | '%':  print(f"Result: {modulus(num1, num2)}")  case '6' | '\*\*':  print(f"Result: {exponent(num1, num2)}")  case '7' | '//':  print(f"Result: {floor\_division(num1, num2)}")  else:  print("Invalid choice! Please select a valid operation.")  # Start  calculator() |

Звіт до Теми №5

Бібліотеки

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

**Гра з комп’ютером**

Необхідно реалізувати таку програму: Програма виконує запит від користувача на введення одного із значень ["stone", "scissor", "paper"]. Наступним кроком, використовуючи модуль random, програма у випадковому порядку вибирає одне із значень ["stone", "scissor", "paper"]. В залежності від умови, що камінь перемагає ножиці, ножиці перемагають папір, а папір перемагає камінь визначити переможця.

Хід виконання завдання:

Створив масив опцій для вибору (камінь, папір та ножиці). Написав функцію для вибору варіанту гравця, та перевірку на правильність вводу. Потім функція вибору варіанту комп’ютером, та перевірку переможця.

Текст програми:

|  |
| --- |
| import random  options = ["stone", "scissor", "paper"]  user\_choice = input("Enter your choice (stone, scissor, paper): ").strip().lower()  if user\_choice not in options:  print("Invalid choice. Please choose from 'stone', 'scissor', or 'paper'.")  else:  computer\_choice = random.choice(options)  print(f"Computer chose: {computer\_choice}")    if user\_choice == computer\_choice:  print("It's a draw!")  elif (user\_choice == "stone" and computer\_choice == "scissor") or \  (user\_choice == "scissor" and computer\_choice == "paper") or \  (user\_choice == "paper" and computer\_choice == "stone"):  print("You win!")  else:  print("Computer wins!") |

**Конвертування іноземної валюти**

Необхідно написати програму. конвертування іноземної валюти в українську гривню. Для отримання актуальних курсів валют необхідно використовувати API НБУ та модуль, що надає можливість виконувати запити до сторонніх сервісів requests. Достатня умова роботи – можливість конвертації для трьох іноземних валют EUR, USD, PLN. Користувачу надається можливість введення кількості та типу валюти, результат роботи програми – конвертоване значення в українських гривнях.

Хід виконання завдання:

Використовуючи модулі urllib.requerst та json (модуль requests чомусь не працював) написав код для отримання поточного курсу валют, та його конвертацію

Текст програми:

|  |
| --- |
| import urllib.request  import json  def get\_exchange\_rate(currency\_code):  url = f"https://bank.gov.ua/NBUStatService/v1/statdirectory/exchange?valcode={currency\_code}&json"  try:  with urllib.request.urlopen(url) as response:  data = response.read()  rates = json.loads(data)  if rates:  return rates[0]["rate"]  else:  print(f"Currency {currency\_code} not found in NBU data.")  return None  except Exception as e:  print(f"Error fetching exchange rate: {e}")  return None  def currency\_converter():  supported\_currencies = {"EUR": "Євро", "USD": "Долар США", "PLN": "Польський злотий"}  print("Supported currencies: EUR, USD, PLN")  currency = input("Enter the currency code (EUR, USD, PLN): ").strip().upper()  if currency not in supported\_currencies:  print("Unsupported currency. Please choose from EUR, USD, PLN.")  return  try:  amount = float(input(f"Enter the amount in {currency}: "))  except ValueError:  print("Invalid amount. Please enter a numeric value.")  return  rate = get\_exchange\_rate(currency)  if rate:  converted\_amount = round(amount \* rate, 2)  print(f"{amount} {currency} = {converted\_amount} UAH")  else:  print("Conversion failed due to issues with fetching the exchange rate.")  if \_\_name\_\_ == "\_\_main\_\_":  currency\_converter() |

**Використання модулів для програми калькулятор**

Необхідно реалізувати використання модулів для програми калькулятор. Функції додавання, віднімання, множення та ділення перенести в файл functions.py. Функції запиту на введення даних для операцій та самих операцій перемістити в файл operations.py. Програму калькулятор реалізувати в файлі calc.py, до якого підключають файл functions.py та operations.py.

Хід виконання завдання:

Розділив вже існуючу програму на три різні файли, реалізувавши імпорт результатів виконання необхідних функцій між файлами.

Текст програми:

|  |
| --- |
| **calc.py** from operations import get\_numbers, perform\_operation  def calculator():  print("Welcome to the calculator!")  print("Type 'exit' at any time to quit.")    while True:  print("\nSelect an operation:")  print("1. Addition (+)")  print("2. Subtraction (-)")  print("3. Multiplication (\*)")  print("4. Division (/)")  print("5. Modulus (%)")  print("6. Exponentiation (\*\*)")  print("7. Floor Division (//)")    choice = input("Enter the number of the operation (1-7) or 'exit' to quit: ").strip().lower()    if choice == 'exit':  print("Goodbye!")  break    if choice in ['1', '2', '3', '4', '5', '6', '7', '+', '-', '\*', '/', '%', '\*\*', '//']:  num1, num2 = get\_numbers()  if num1 is None or num2 is None:  continue    result = perform\_operation(choice, num1, num2)  print(f"Result: {result}")  else:  print("Invalid choice! Please select a valid operation.")  # Start the calculator  if \_\_name\_\_ == "\_\_main\_\_":  calculator()  **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 a / b  else:  return "Division by zero is not allowed!"  def modulus(a, b):  return a % b  def exponent(a, b):  return a \*\* b  def floor\_division(a, b):  if b != 0:  return a // b  else:  return "Division by zero is not allowed!"  **operations.py**  from functions import add, subtract, multiply, divide, modulus, exponent, floor\_division  def get\_numbers():  try:  num1 = float(input("Enter the first number: "))  num2 = float(input("Enter the second number: "))  return num1, num2  except ValueError:  print("Invalid input! Please enter numerical values.")  return None, None  def perform\_operation(choice, num1, num2):  match choice:  case '1' | '+':  return add(num1, num2)  case '2' | '-':  return subtract(num1, num2)  case '3' | '\*':  return multiply(num1, num2)  case '4' | '/':  return divide(num1, num2)  case '5' | '%':  return modulus(num1, num2)  case '6' | '\*\*':  return exponent(num1, num2)  case '7' | '//':  return floor\_division(num1, num2)  case \_:  return "Invalid choice!" |

Звіт до Теми №6

Робота з файлами

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

**Механізм логування**

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

Хід виконання завдання:

Модифікував програму калькулятора функціями логування. Нижче наведені зміни в файлах:

Текст програми:

|  |
| --- |
| **calc.py**  **import logging**  from operations import get\_numbers, perform\_operation  **logging.basicConfig(**  **filename="calculator.log",**  **level=logging.INFO,**  **format="%(asctime)s - %(levelname)s - %(message)s",**  **)**  def calculator():  print("Welcome to the calculator!")  print("Type 'exit' at any time to quit.")    while True:  print("\nSelect an operation:")  print("1. Addition (+)")  print("2. Subtraction (-)")  print("3. Multiplication (\*)")  print("4. Division (/)")  print("5. Modulus (%)")  print("6. Exponentiation (\*\*)")  print("7. Floor Division (//)")    choice = input("Enter the number of the operation (1-7) or 'exit' to quit: ").strip().lower()    if choice == 'exit':  **logging.info("Користувач завершив роботу.")**  print("Goodbye!")  break    if choice in ['1', '2', '3', '4', '5', '6', '7', '+', '-', '\*', '/', '%', '\*\*', '//']:  num1, num2 = get\_numbers()  if num1 is None or num2 is None:  continue    result = perform\_operation(choice, num1, num2)  print(f"Result: {result}")  else:  print("Invalid choice! Please select a valid operation.")  # Start the calculator  if \_\_name\_\_ == "\_\_main\_\_":  **logging.info("Запуск програми калькулятор.")**  calculator()  **logging.info("Програма завершена.")**  **operations.py**  **import logging**  from functions import add, subtract, multiply, divide, modulus, exponent, floor\_division  def get\_numbers():  try:  num1 = float(input("Enter the first number: "))  num2 = float(input("Enter the second number: "))  return num1, num2  except ValueError:  **logging.warning("Некоректне введення даних користувачем.")**  print("Invalid input! Please enter numerical values.")  return None, None  def perform\_operation(choice, num1, num2):  result = None  operation = ""  match choice:  case '1' | '+':  result = add(num1, num2)  operation = "Addition"  case '2' | '-':  result = subtract(num1, num2)  operation = "Subtraction"  case '3' | '\*':  result = multiply(num1, num2)  operation = "Multiplication"  case '4' | '/':  result = divide(num1, num2)  operation = "Division"  case '5' | '%':  result = modulus(num1, num2)  operation = "Modulus"  case '6' | '\*\*':  result = exponent(num1, num2)  operation = "Exponentiation"  case '7' | '//':  result = floor\_division(num1, num2)  operation = "Floor Division"  case \_:  **logging.warning("Користувач обрав некоректну операцію.")**  return "Invalid choice!"      if result is not None:  **logging.info(f"Операція: {operation}, Введені дані: {num1}, {num2}, Результат: {result}")**    return result |

**Сортування списку**

Необхідно маючи не відсортований список, елементами якого є словники з двома параметрами (ім’я та оцінка) виконати сортування списку, використовуючи стандартну функцію sorted(). Другим параметром для функції sorted() має бути lambda функція, що повертає ім’я або оцінку із елемента словника.

Хід виконання завдання:

Створив умовний список студентів та їх оцінок. Реалізував два сортування, використовуючи лямбда функцію, за іменем та за оцінками в порядку спадання.

Текст програми:

|  |
| --- |
| students = [  {'name': 'Kyrylo', 'score': 85},  {'name': 'Dmytro', 'score': 92},  {'name': 'Maxim', 'score': 78},  {'name': 'Sofia', 'score': 88}  ]  sorted\_by\_name = sorted(students, key=lambda x: x['name'])  print("Сортування за ім'ям:", sorted\_by\_name)  sorted\_by\_score = sorted(students, key=lambda x: x['score'], reverse=True)  print("Сортування за оцінкою:", sorted\_by\_score) |

Звіт до Теми №7

Об’єктно-орієнтоване програмування

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

**Методики класу**

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

Хід виконання завдання:

Створив найпростіші класи Person та Unit, та задав їм атрибути. Для першого класу не задавав метод str, та порівняв їх відображення.

Текст програми:

|  |
| --- |
| class Person:  def \_\_init\_\_(self, name, age):  self.name = name  self.age = age  person1 = Person("Alice", 30)  print(person1.name) #Outcome "Alice"  print(person1) #Outcome "<\_\_main\_\_.Person object at 0x000002AE9123E060>"  class Unit:  def \_\_init\_\_(self, cost, damage):  self.cost = cost  self.damage = damage    def \_\_str\_\_(self):  return f"Unit (Ціна={self.cost}, шкода={self.damage})"    warrior = Unit(100, 5)  print(warrior) #Outcome: "Unit (Ціна=100, шкода=5)" |

**Клас Student**

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:

Оголосив клас Student, вніс в нього дані про студентів, а потім відсортував студентів використовуючи лямбда функцію для отримання віку.

Текст програми:

|  |
| --- |
| # Class init  class Student:  def \_\_init\_\_(self, name, age):  self.name = name  self.age = age  def \_\_str\_\_(self):  return f"Student(name='{self.name}', age={self.age})"  # List of students  students = [  Student("Anna", 20),  Student("Bohdan", 22),  Student("Kateryna", 19),  Student("Dmytro", 21),  ]  # Sorting by age  sorted\_students = sorted(students, key=lambda student: student.age)  # Output  for student in sorted\_students:  print(student) |

**Калькулятор**

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

Хід виконання завдання:

Список основних змін:

* Основна програма очищена від функцій, залишено лише перевірку виконання основної програми та імпорт функцій.
* Створено файл calculator.py, куди перенесені функції виводу інформаційних повідомлень про список операцій, та запит на вибір операції.
* Файл operations.py збережено структуру, але функції об’єднано в клас.

Текст програми:

|  |
| --- |
| **calc.py** from calculator import Calculator  if \_\_name\_\_ == "\_\_main\_\_":  calc = Calculator()  calc.run()  **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 a / b  else:  return "Division by zero is not allowed!"  def modulus(a, b):  return a % b  def exponent(a, b):  return a \*\* b  def floor\_division(a, b):  if b != 0:  return a // b  else:  return "Division by zero is not allowed!"  **operations.py**  import logging  from functions import add, subtract, multiply, divide, modulus, exponent, floor\_division  class OperationManager:  def get\_numbers(self):  try:  num1 = float(input("Enter the first number: "))  num2 = float(input("Enter the second number: "))  return num1, num2  except ValueError:  logging.warning("Некоректне введення даних користувачем.")  print("Invalid input! Please enter numerical values.")  return None, None  def process\_choice(self, choice):  num1, num2 = self.get\_numbers()  if num1 is None or num2 is None:  return    operations = {  '1': ('Addition', add),  '+': ('Addition', add),  '2': ('Subtraction', subtract),  '-': ('Subtraction', subtract),  '3': ('Multiplication', multiply),  '\*': ('Multiplication', multiply),  '4': ('Division', divide),  '/': ('Division', divide),  '5': ('Modulus', modulus),  '%': ('Modulus', modulus),  '6': ('Exponentiation', exponent),  '\*\*': ('Exponentiation', exponent),  '7': ('Floor Division', floor\_division),  '//': ('Floor Division', floor\_division),  }    if choice in operations:  operation\_name, func = operations[choice]  result = func(num1, num2)  logging.info(f"Операція: {operation\_name}, Введені дані: {num1}, {num2}, Результат: {result}")  print(f"Result: {result}")  else:  logging.warning("Користувач обрав некоректну операцію.")  print("Invalid choice!")  **calculator.py**  import logging  from operations import OperationManager  logging.basicConfig(  filename="calculator.log",  level=logging.INFO,  format="%(asctime)s - %(levelname)s - %(message)s",  )  class Calculator:  def \_\_init\_\_(self):  self.operations = OperationManager()  logging.info("Калькулятор ініціалізовано.")  def display\_menu(self):  print("\nSelect an operation:")  print("1. Addition (+)")  print("2. Subtraction (-)")  print("3. Multiplication (\*)")  print("4. Division (/)")  print("5. Modulus (%)")  print("6. Exponentiation (\*\*)")  print("7. Floor Division (//)")  print("Type 'exit' to quit.")  def run(self):  print("Welcome to the calculator!")  while True:  self.display\_menu()  choice = input("Enter your choice: ").strip().lower()  if choice == 'exit':  logging.info("Користувач завершив роботу.")  print("Goodbye!")  break  self.operations.process\_choice(choice) |

Посилання на github:

https://github.com/MrMostliker/TP-KB-231-Dmytro-Potapenko

Знімок екрану з посилання на github:

