**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:

Результат виконання програми:



Текст програми:

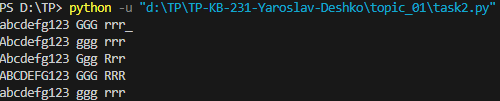
|  |
| --- |
| myStr = "abcdefg123"  print(myStr[::-1]) |

Тестування функцій

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

Хід виконання завдання:

Результат виконання програми:



Текст програми:

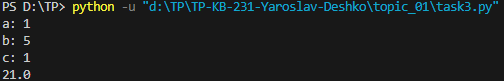
|  |
| --- |
| myStr = "abcdefg123 GGG rrr  "  print(myStr.strip() + "\_")  print(myStr.capitalize())  print(myStr.title())  print(myStr.upper())  print(myStr.lower()) |

Функція пошуку дискримінанта.

Написати функцію пошуку дискримінанту квадратного рівняння.

Хід виконання завдання:

Результат виконання програми:



Текст програми:

|  |
| --- |
| def discriminant(a: float, b: float, c: float) -> float:      return pow(b, 2) - 4\*a\*c  a = float(input("a: "))  b = float(input("b: "))  c = float(input("c: "))  print(discriminant(a, b, c)) |

Звіт до Теми №2

Умовний перехід

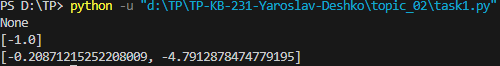
Під час виконання практичного завдання до Теми №2 було надано варіанти рішення до наступних задач:

Пошук коренів квадратного рівняння.

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

Хід виконання завдання:

Результат виконання програми:



Текст програми:

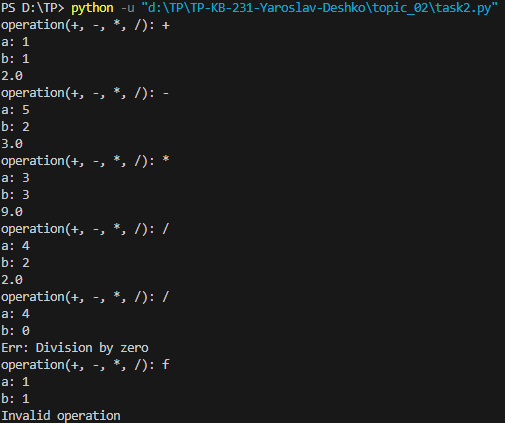
|  |
| --- |
| from math import sqrt  def discriminant(a: float, b: float, c: float) -> float:      return pow(b, 2) - 4\*a\*c  def findRoots(a: float, b: float, c: float):    d = discriminant(a, b, c)    if d == 0:       return [-(b/2\*a)]    elif d > 0:       return [((-b)+sqrt(d))/ 2\*a,               ((-b)-sqrt(d))/ 2\*a ]    return None  def test():    print(findRoots(1, 2, 2)) # D < 0    print(findRoots(1, 2, 1)) # D == 0    print(findRoots(1, 5, 1)) # D > 0      test() |

Калькулятор на **if else.**

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

Хід виконання завдання:

Результат виконання програми:



Текст програми:

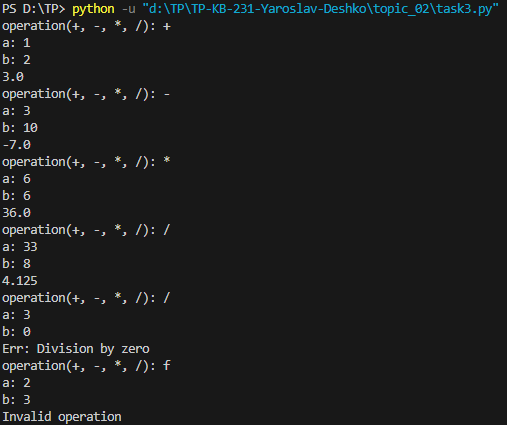
|  |
| --- |
| def calculator():      opr = input("operation(+, -, \*, /): ")      a, b = enterNum()      if opr == "+":         return add(a, b)      elif opr == "-":         return sub(a, b)      elif opr == "\*":         return mul(a, b)      elif opr == "/":         return div(a, b)      else: return "Invalid operation"  def enterNum():      a = input("a: ")      b = input("b: ")      return float(a), float(b)  def add(a, b):      return a+b  def sub(a, b):      return a-b  def mul(a, b):      return a\*b  def div(a, b):      if b == 0 : return "Err: Division by zero"      return a/b  while True:      print(calculator()) |

Калькулятор на **match.**

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

Хід виконання завдання:

Результат виконання програми:



Текст програми:

|  |
| --- |
| def calculator():      opr = input("operation(+, -, \*, /): ")      a, b = enterNum()      match opr:          case "+":              return add(a, b)          case "-":              return sub(a, b)          case "\*":              return mul(a, b)          case "/":              return div(a, b)          case \_:              return "Invalid operation"  def enterNum():      a = input("a: ")      b = input("b: ")      return float(a), float(b)  def add(a, b):      return a+b  def sub(a, b):      return a-b  def mul(a, b):      return a\*b  def div(a, b):      if b == 0 : return "Err: Division by zero"      return a/b  while True:      print(calculator()) |

Звіт до Теми №3

Цикли

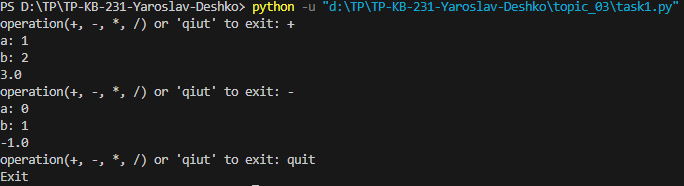
Під час виконання практичного завдання до Теми №3 було надано варіанти рішення до наступних задач:

Калькулятор з постійними запитами на введення.

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:

Результат виконання програми:



Текст програми:

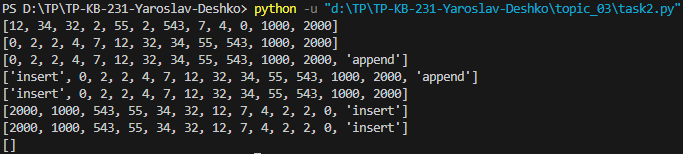
|  |
| --- |
| def calculator(opr: str):      a, b = enterNum()      match opr:          case "+":              return add(a, b)          case "-":              return sub(a, b)          case "\*":              return mul(a, b)          case "/":              return div(a, b)          case \_:              return "Invalid operation"      def enterNum():      a = input("a: ")      b = input("b: ")      return float(a), float(b)  def enterOpr():      opr = input("operation(+, -, \*, /) or 'qiut' to exit: ")      return opr  def add(a, b):      return a+b  def sub(a, b):      return a-b  def mul(a, b):      return a\*b  def div(a, b):      if b == 0 : return "Err: Division by zero"      return a/b  while True:      opr = enterOpr()      if opr!="quit":          print(calculator(opr))      else:          print("Exit")          break |

Тестування функцій списків.

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

Хід виконання завдання:

Результат виконання програми:



Тест програми:

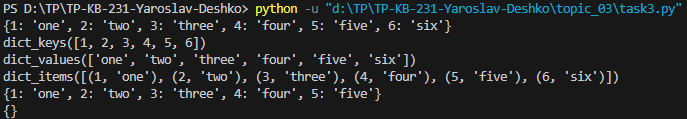
|  |
| --- |
| myList = [12,34,32,2,55,2,543,7,4,0]  myList.extend([1000, 2000])  print(myList)  myList.sort()  print(myList)  myList.append("append")  print(myList)  myList.insert(0, "insert")  print(myList)  myList.remove("append")  print(myList)  myList.reverse()  print(myList)  newList = myList.copy()  print(newList)  myList.clear()  print(myList) |

Тестування функцій словників.

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

Хід виконання завдання:

Результат виконання програми:



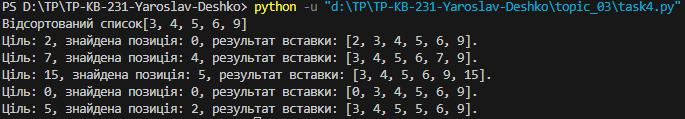
Текст програми:

|  |
| --- |
| myDict = {      1 : "one",      2: "two",      3: "three",      4: "four",      5: "five",  }  myDict.update({6:"six"})  print(myDict)  print(myDict.keys())  print(myDict.values())  print(myDict.items())  del myDict[6]  print(myDict)  myDict.clear()  print(myDict) |

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

Хід виконання завдання:

Результат виконання програми:



Текст програми:

|  |
| --- |
| def findInsertPosition(sortedList: list, target: int) -> int:      left = 0      right = len(sortedList)      while left < right:          mid = (left+right)//2          if sortedList[mid] < target:              left = mid+1          else: right = mid      return left  def insertNum(sortedList: list, num: int)-> str:      workList= sortedList.copy()      index = findInsertPosition(workList, num)      workList.insert(index, num)      return f"Ціль: {num}, знайдена позиція: {index}, результат вставки: {workList}."  sortedList = [3, 4, 5, 6, 9]  print(f"Відсортований список{sortedList}")  print(insertNum(sortedList, 2))  print(insertNum(sortedList, 7))  print(insertNum(sortedList, 15))  print(insertNum(sortedList, 0))  print(insertNum(sortedList, 5)) |

Звіт до Теми №4

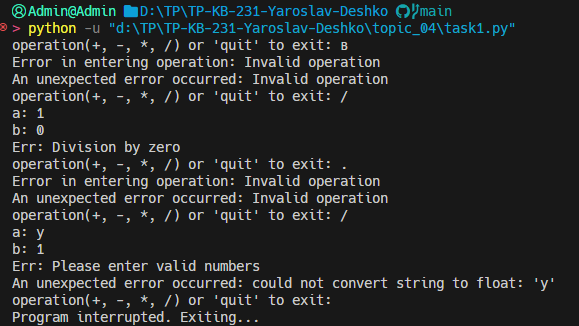
Виняткові ситуації

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.
2. Розширити функцію ділення обробкою виняткової ситуації ділення но нуль
3. Ознайомитись зі списком виняткових ситуацій за посиланням <https://docs.python.org/3/library/exceptions.html>

Хід виконання завдання:

Результат виконання програми:



Текст програми:

|  |
| --- |
| def calculator(opr: str):      a, b = enterNum()      match opr:          case "+":              return add(a, b)          case "-":              return sub(a, b)          case "\*":              return mul(a, b)          case "/":              return div(a, b)  def enterNum():      try:          a = input("a: ")          b = input("b: ")          return float(a), float(b)      except ValueError:          print("Err: Please enter valid numbers")          raise  def enterOpr():      try:          opr = input("operation(+, -, \*, /) or 'quit' to exit: ").strip().lower()          valid\_operations = ['+', '-', '\*', '/', 'quit']          if opr not in valid\_operations:              raise ValueError("Invalid operation")            return opr        except Exception as e:          print(f"Error in entering operation: {e}")          raise    def add(a, b):      return a+b  def sub(a, b):      return a-b  def mul(a, b):      return a\*b  def div(a, b):      try:          return a / b      except ZeroDivisionError:          return "Err: Division by zero"    while True:      try:          opr = enterOpr()          if opr == "quit":              print("Exit")              break            result = calculator(opr)          print(result)        except KeyboardInterrupt:          print("\nProgram interrupted. Exiting...")          break      except Exception as e:          print(f"An unexpected error occurred: {e}") |

Звіт до Теми №5

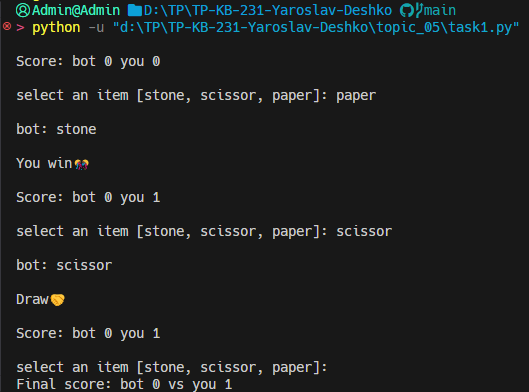
Бібліотеки

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 визначити переможця.

Хід виконання завдання:

Результат виконання програми:



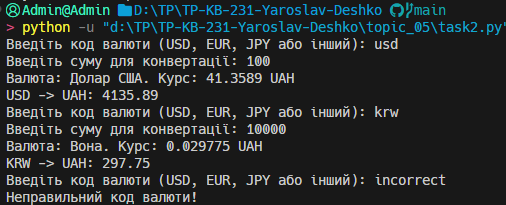
Код програми:

|  |
| --- |
| import random  objects = {      "stone": "scissor",      "scissor" : "paper",      "paper":"stone"  }  def playersChoice(objects:list)->list:      bot = random.choice(list(objects.keys()))      try:          player = input("select an item [stone, scissor, paper]: ")          if player not in ["stone", "scissor", "paper"]:              raise ValueError("Incorrect input")        except Exception as e :          print(f"Error in entering operation: {e}")          return playersChoice(objects)      print(f"\nbot: {bot}\n")      return bot, player  def game(playersChoice:list, objects: list) -> list:      if (playersChoice[0] == playersChoice[1]):          print("Draw🤝")          return [0, 0]      elif(objects[playersChoice[0]] != playersChoice[1]):          print("You win🎊")          return [0, 1]      else:          print("You lose💢")          return [1, 0]  score = [0, 0]  while True:      try:          print(f"\nScore: bot {score[0]} you {score[1]}\n")          res = game(playersChoice(objects), objects)          if(res[0]==0 and res[1]==0):              continue          elif(res[0]==1):              score[0] = score[0] + 1          else:              score[1] = score[1] + 1      except KeyboardInterrupt:          print(f"\nFinal score: bot {score[0]} vs you {score[1]}")          break |

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

Хід виконання завдання:

Результат виконання програми:



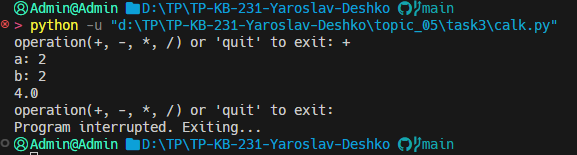
Код програми:

|  |
| --- |
| import requests  BASE\_URL = "https://bank.gov.ua/NBUStatService/v1/statdirectory/exchange?valcode="  while True:      cc = input("Введіть код валюти (USD, EUR, JPY або інший): ")      r = requests.get(BASE\_URL + cc + "&json")      if r.json() == []:          print("Неправильний код валюти!")      else:          m = float(input("Введіть суму для конвертації: "))          print(f"Валюта: {r.json()[0]["txt"]}. Курс: {r.json()[0]["rate"]} UAH")          print(cc.upper() + " -> " + "UAH: " + f"{(m \* float(r.json()[0]['rate'])):.2f}") |

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. Програму калькулятор реалізувати в файлі calc.py, до якого підключають файл functions.py та operations.py.

Хід виконання завдання:

Результат виконання програми:



Код програми:

|  |
| --- |
| #calk.py  from operations import enterNum, enterOpr  from functions import add, sub, mul, div  def calculator(opr: str):      a, b = enterNum()      match opr:          case "+":              return add(a, b)          case "-":              return sub(a, b)          case "\*":              return mul(a, b)          case "/":              return div(a, b)  while True:      try:          opr = enterOpr()          if opr == "quit":              print("Exit")              break            result = calculator(opr)          print(result)        except KeyboardInterrupt:          print("\nProgram interrupted. Exiting...")          break      except Exception as e:          print(f"An unexpected error occurred: {e}")  #functions.py  def add(a, b):      return a+b  def sub(a, b):      return a-b  def mul(a, b):      return a\*b  def div(a, b):      try:          return a / b      except ZeroDivisionError:          return "Err: Division by zero"  #operations.py  def enterNum():      try:          a = input("a: ")          b = input("b: ")          return float(a), float(b)      except ValueError:          print("Err: Please enter valid numbers")          raise  def enterOpr():      try:          opr = input("operation(+, -, \*, /) or 'quit' to exit: ").strip().lower()          valid\_operations = ['+', '-', '\*', '/', 'quit']          if opr not in valid\_operations:              raise ValueError("Invalid operation")            return opr        except Exception as e:          print(f"Error in entering operation: {e}")          raise |

Звіт до Теми №6

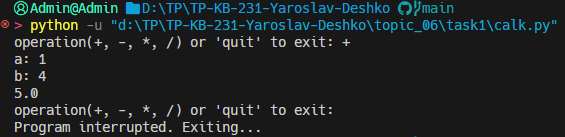
Робота з файлами

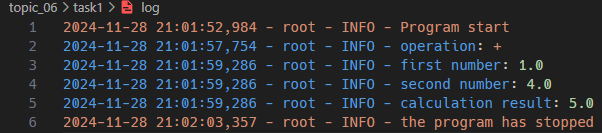
Під час виконання практичного завдання до Теми №6 було надано варіанти рішення до наступних задач:

1) 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:

Результат виконання програми:





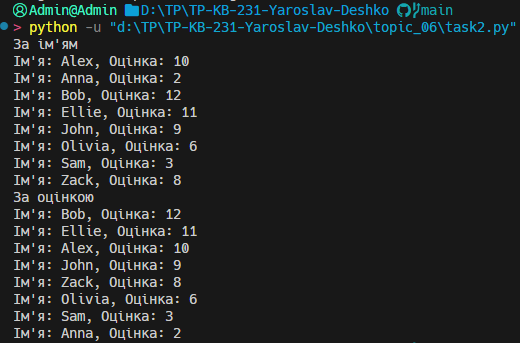
Код програми:

|  |
| --- |
| #calk.py  from operations import enterNum, enterOpr  from functions import add, sub, mul, div  import logging  logging.basicConfig(      filename="topic\_06/task1/log",      filemode="a",      format="%(asctime)s - %(name)s - %(levelname)s - %(message)s",      level=logging.INFO  )  logger = logging.getLogger()  def calculator(opr: str):      a, b = enterNum()      logger.info(f"first number: {a}")      logger.info(f"second number: {b}")      match opr:          case "+":              return add(a, b)          case "-":              return sub(a, b)          case "\*":              return mul(a, b)          case "/":              return div(a, b)  logger.info("Program start")  while True:      try:          opr = enterOpr()          logger.info(f"operation: {opr}")          if opr == "quit":              print("Exit")              logger.info(f"the program has stopped")              break            result = calculator(opr)          print(result)          logger.info(f"calculation result: {result}")        except KeyboardInterrupt:          print("\nProgram interrupted. Exiting...")          logger.info(f"the program has stopped")          break      except Exception as e:          print(f"An unexpected error occurred: {e}")          logger.info(f"An unexpected error occurred: {e}")  #functions.py  def add(a, b):      return a+b  def sub(a, b):      return a-b  def mul(a, b):      return a\*b  def div(a, b):      try:          return a / b      except ZeroDivisionError:          return "Err: Division by zero"  #operations.py  def enterNum():      try:          a = input("a: ")          b = input("b: ")          return float(a), float(b)      except ValueError:          print("Err: Please enter valid numbers")          raise  def enterOpr():      try:          opr = input("operation(+, -, \*, /) or 'quit' to exit: ").strip().lower()          valid\_operations = ['+', '-', '\*', '/', 'quit']          if opr not in valid\_operations:              raise ValueError("Invalid operation")            return opr        except Exception as e:          print(f"Error in entering operation: {e}")          raise |

2) 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:

Результат виконання програми:



Код програми:

|  |
| --- |
| workList = [      {"name": "Sam", "mark": 3},      {"name": "John", "mark": 9},      {"name": "Bob", "mark": 12},      {"name": "Alex", "mark": 10},      {"name": "Anna", "mark": 2},      {"name": "Zack", "mark": 8},      {"name": "Ellie", "mark": 11},      {"name": "Olivia", "mark": 6},  ]  def fprint(key: str, List:list ,reverse=False):      sortedList = sorted(List, key= lambda x:x[key], reverse=reverse)      for student in sortedList:          print(f"Ім'я: {student['name']}, Оцінка: {student['mark']}")  print("За ім'ям")  fprint("name", workList)  print("За оцінкою")  fprint("mark", workList, reverse=True) |

Звіт до Теми №7

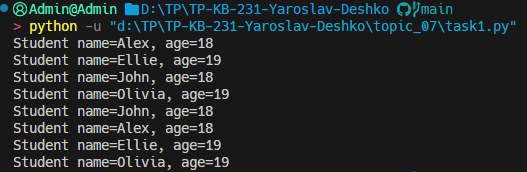
ООП

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

1. Ознайомитись з документацією що описує можливості використання класів у мові Python <https://docs.python.org/3/tutorial/classes.html>
2. Ознайомитись з існуючими за замовченням методами класу по типу \_\_init\_\_(self) \_\_str\_\_(self)\_\_ та надати приклади використання.
3. 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:

Результат виконання програми:



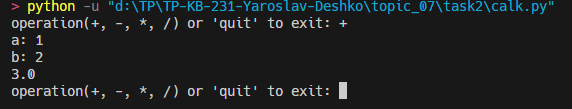
Код програми:

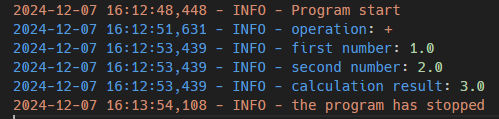
|  |
| --- |
| class Student:      def \_\_init\_\_(self, name, age):          self.name = name          self.age = age      def \_\_repr\_\_(self):          return f"Student name={self.name}, age={self.age}"  students = [      Student("John", 18),      Student("Ellie", 19),      Student("Alex", 18),      Student("Olivia", 19)  ]  sorted\_students = sorted(students, key= lambda student : student.name)  for student in sorted\_students:      print(student)  sorted\_students = sorted(students, key= lambda student : student.age)  for student in sorted\_students:      print(student) |

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

Хід виконання завдання:

Результат роботи програми:





Код програми:

|  |
| --- |
| from operations import Operations  from functions import Calc  import logging  calc = Calc  operations = Operations  logging.basicConfig(      filename="topic\_07/task2/log",      filemode="a",      format="%(asctime)s - %(levelname)s - %(message)s",      level=logging.INFO  )  logger = logging.getLogger()  def calculator(opr: str):      a, b = operations.enterNum()      logger.info(f"first number: {a}")      logger.info(f"second number: {b}")      match opr:          case "+":              return calc.add(a, b)          case "-":              return calc.sub(a, b)          case "\*":              return calc.mul(a, b)          case "/":              return calc.div(a, b)  logger.info("Program start")  while True:      try:          opr = operations.enterOpr()          logger.info(f"operation: {opr}")          if opr == "quit":              print("Exit")              logger.info(f"the program has stopped")              break            result = calculator(opr)          print(result)          logger.info(f"calculation result: {result}")        except KeyboardInterrupt:          print("\nProgram interrupted. Exiting...")          logger.info(f"the program has stopped")          break      except Exception as e:          print(f"An unexpected error occurred: {e}")          logger.error(f"An unexpected error occurred: {e}")  class Calc:      def add(a, b):          return a + b        def sub(a, b):          return a-b      def mul(a, b):          return a\*b        def div(a, b):          try:              return a / b          except ZeroDivisionError:              return "Err: Division by zero"  class Operations:      def enterNum():          try:              a = input("a: ")              b = input("b: ")              return float(a), float(b)          except ValueError:              print("Err: Please enter valid numbers")              raise      def enterOpr():          try:              opr = input("operation(+, -, \*, /) or 'quit' to exit: ").strip().lower()              valid\_operations = ['+', '-', '\*', '/', 'quit']              if opr not in valid\_operations:                  raise ValueError("Invalid operation")                return opr            except Exception as e:              print(f"Error in entering operation: {e}")              raise |

Посилання на github:

https://github.com/Kaspo02/TP-KB-231-Yaroslav-Deshko

Знімок екрану з посилання на github:

