**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:

Створив функцію що розвертає рядок.

Текст програми:

|  |
| --- |
| string = "abcdefg123"  def reverse(var):  var = var[::-1]  return var  reversedString = reverse(string)  print(reversedString) |

**Тестування методів для рядків**

Необхідно протестувати деякі методи для рдяків.

Хід виконання завдання:

Протестував такі методи:

Strip() – видаляє пробіли і табуляції спереду та позаду рядку.

Capitalize() – робить 1 букву рядку вищого регістру, інші нижчого.

Title() – всі 1 букви кожного слова в рядку робить вищого регістру.

Upper() – всі букви до верхнього регістру.

Lower() – всі букви до нижчого регістру.

Find() – шукає позицію символа в рядку.

Split() – розбиває рядок по заданому символу та записує частини в масив.

Count() – рахує кількість входжень символу до рядку.

Swapcase() – міняє місцями регістри літер.

Текст програми:

|  |
| --- |
| #strip() method  stringWithSpaces = " I am Denis "  print (stringWithSpaces.strip())  #capitalize() method  notCapitalizedString = "i am a student"  print (notCapitalizedString.capitalize())  #title() method  string = "Hello i am denis"  print (string.title())  #upper() and lower() methods  weirdString = "hElLo i aM dEnIs aNd i aM a sTuDeNt"  print (weirdString.upper())  print (weirdString.lower())  #some other  #find() method  print(string.find("i"))  #split() method  print(weirdString.upper().split(" "))  #count() method  print(string.count("l"))  #swapcase() method  print (weirdString.swapcase()) |

**Знаходження дискримінанта**

Необхідно створити функцію що знаходить дискримінант.

Хід виконання завдання:

Формула дискримінанта b\*\*2 – 4\*a\*c. Запрошуємо ввід 3-х коефіцієнтів від користувача та передаємо їх функції на обчислення.

Текст програми:

|  |
| --- |
| def dSearching(a, b, c):  res = b\*\*2-4\*a\*c  return res  a = float(input("Enter a: "))  b = float(input("Enter b: "))  c = float(input("Enter c: "))  D = dSearching(a, b, c)  print(D) |

Посилання на github:

<https://github.com/Aatheeriss/TP-KB-221-Denis-Sazonenko.git>

Звіт до Теми №2

Умовні переходи

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

**Пошук коренів квадратного рівняння**

Необхідно дописати умови перевірки дискримінанта та знайти корені рівняння.

Хід виконання завдання:

Створив умови перевірки чи дискримінант більший, дорівнює, або інше (тобто менший) за нуль. Створив функцію для знаходження коренів.

Код програми:

|  |
| --- |
| #calculating D  def dSearching(a, b, c):      res = b\*\*2 - 4\*a\*c      return res  #calculating D  def dSearching(a, b, c):      res = b\*\*2 - 4\*a\*c      return res  #variables input  a = float(input("Enter a: "))  b = float(input("Enter b: "))  c = float(input("Enter c: "))  #D output  D = dSearching(a, b, c)  print("\nD =", D)  #topic\_02  #roots calculating function  def calculatingRoots(b, D, a):      x1 = (-b + D\*\*0.5) / (2\*a)      x2 = (-b - D\*\*0.5) / (2\*a)      if x1 == x2:          return x1      else:          return x1, x2    #D conditions check and roots output  if D > 0:      print("\nEquation has 2 roots: ")      print("x =", calculatingRoots(b, D, a))  elif D == 0:      x = -b / (2\*a)      print("\nEquation has 1 root: ")      print("x =", calculatingRoots(b, D, a))  else:      print("\nEquation has no roots.") |

**Калькулятор з конструкціями if else**

Потрібно створити калькулятор з умовними переходами.

Хід виконання завдання:

Створив калькулятор з 4 діями. Ділення на 0 – виключення.

Код програми:

|  |
| --- |
| a = float(input("Enter the first number: "))  b = float(input("Enter the second number: "))  action = input("Enter action (+, -, \*, /): ").strip()  if action == "+":      print(a + b)  elif action == "-":      print(a - b)  elif action == "\*":      print(a \* b)  elif action == "/":      try:          result = a / b          print(result)      except ZeroDivisionError:          print("Division by zero is not allowed.")  else:      print("Invalid action. Please enter +, -, \*, or /.") |

**Калькулятор з конструкцією match case**

Потрібно створити калькулятор з конструкцією match case.

Хід виконання завдання:

Створив калькулятор з 4 діями. Ділення на 0 – виключення.

Код програми:

|  |
| --- |
| a = float(input("Enter the first number: "))  b = float(input("Enter the second number: "))  action = input("Enter action (+, -, \*, /): ").strip()  match action:      case "+":          print(a + b)      case "-":          print(a - b)      case "\*":          print(a \* b)      case "/":          try:              result = a / b              print(result)          except ZeroDivisionError:              print("Division by zero is not allowed.")      case \_:          print("Invalid action. Please enter +, -, \*, or /.") |

Звіт до Теми №3

Цикли

**Калькулятор:**

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:

Додав нескінченний цикл.

Код програми:

|  |
| --- |
| while True:  a = float(input("Enter the first number: "))  b = float(input("Enter the second number: "))  action = input("Enter action (+, -, \*, /): ").strip()  if action == "+":  print(a + b)  elif action == "-":  print(a - b)  elif action == "\*":  print(a \* b)  elif action == "/":  try:  result = a / b  print(result)  except ZeroDivisionError:  print("Division by zero is not allowed.")  else:  print("Invalid action. Please enter +, -, \*, or /.") |

**Тестування функцій**:

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

Хід виконання завдання:

Написав програму тестування даних функцій.

Код програми:

|  |
| --- |
| #1. extend()  def extendTesting():  list1 = [1, 2, 3, 4, 5]  list2 = [6, 7, 8, 9, 0]  list1.extend(list2)  print(list1)  return  #2. append()  def appendTesting():  list1 = ['Hello', 'I', 'am']  list1.append('Denis')  print(list1)  return  #3. insert(id, val)  def insertTesting():  list1 = ['Hello', 'I', 'Denis']  list1.insert(2, 'am')  print(list1)  return  #4. remove()  def removeTesting():  list1 = ['Hello', 'I', 'removeMEpLeAsE', 'am', 'Denis']  list1.remove('removeMEpLeAsE')    print(list1)  return  #5. clear()  def clearTesting():  list1 = [1, 2, 3, 4, 5]  list1.clear()  list1.append('list is empty but this message')  print(list1)  return  #6. sort()  def sortTesting():  list1 = [2, 5, 3, 1, 4]  list1.sort()  print(list1)  return  #7. reverse()  def reverseTesting():  list1 =[5, 4, 3, 2, 1]  list1.reverse()  print(list1)  return  #8. copy()  def copyTesting():  list1 = [1, 2, 3, 4, 5]  print(list1.copy())  return  def main():  a = input("Enter number 1-8:\n 1. extend()\n 2. append()\n 3. insert()\n 4. remove()\n 5. clear()\n 6. sort()\n 7. reverse()\n 8. copy()\n")  match a:  case "1":  extendTesting()  case "2":  appendTesting()  case "3":  insertTesting()  case "4":  removeTesting()  case "5":  clearTesting()  case "6":  sortTesting()  case "7":  reverseTesting()  case "8":  copyTesting()  case \_:  print("Try again!")    print("\n\n-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\n\n")    return  while True:  main() |

**Тестування функцій:**

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

Хід виконання завдання:

Написав програму тестування даних функцій.

Код програми:

|  |
| --- |
| laptop = {  "brand": "lenovo",  "series": "legion"  }  #1. update()  def updateTesting():  laptop.update({"serial number": "LL252433"})  print(laptop)  return  #2. del()  def delTesting():  del laptop["brand"]  print(laptop)  return  #3. clear()  def clearTesting():  laptop.clear()  print(laptop)  return  #4. keys()  def keysTesting():  laptop.keys()  laptop["brand"] = "asus"  print(laptop)  return  #5. values()  def valuesTesting():    laptop.values()  laptop["brand"] = "asus"  print(laptop)  return  #6. items()  def itemsTesting():  laptop.items()  laptop["brand"] = "asus"  print(laptop)  return  def main():  a = input("Enter number 1-6:\n 1. update()\n 2. del()\n 3. clear()\n 4. keys()\n 5. values()\n 6. items()\n")  match a:  case "1":  updateTesting()  case "2":  delTesting()  case "3":  clearTesting()  case "4":  keysTesting()  case "5":  valuesTesting()  case "6":  itemsTesting()  case \_:  print("Try again!")    print("\n\n-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\n\n")    return  while True:  main() |

**Функція пошуку позиціі**

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

Хід виконання завдання:

Створив функцію бінарного пошуку елементу в масиві, та позиціі для додавання.

Код програми:

|  |
| --- |
| def positionFind(list1, element):  list\_start = 0  list\_end = len(list1) - 1  while list\_start <= list\_end:  list\_mid = (list\_start + list\_end) // 2  if list1[list\_mid] == element:  print("Element is already in list. Position is: " + str(list\_mid))  return  elif list1[list\_mid] < element:  list\_start = list\_mid + 1  else:  list\_end = list\_mid - 1  print ("Position to add is: " + str(list\_start))  return  list1 = [1, 3, 5, 7, 9]  while True:  print(list1)  element = input("Enter the element to find position to add: ")  element = int(element)  position = positionFind(list1, element) |

Звіт до Теми №4

Виняткові ситуації

**Розширення калькулятора:**

**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

Хід виконання завдання:

Розклав дії програми по окремих функціях. В функції insertion реалізував захоплення винятку ValueError. В фукнції calculation – ZeroDivisionError.

Загалом по програмі – keyboardInterrupt exeption, який виводить повідомлення перед закінченням програми у випадку, якщо програма переривається за допомогою клавіатури.

Функція insertion буде запрошувати дані на введення до тих пір доки не отримає валідні дані для обробки. Функція calculation завершиться або виведенням підрахованого значення, або зайде в виняток ділення на нуль и закінчиться, програма почне роботу спочатку завдяки нескінченному циклу визову функції main, яка в свою чергу визиває попередні функції.

Код програми:

|  |
| --- |
| try:      #insertion function which repeats until getting valid values of required functions      def insertion():          while True:              try:                  a = float(input("Enter the first number: "))                  b = float(input("Enter the second number: "))                  action = input("Enter action (+, -, \*, /): ").strip()                  if action not in ('+', '-', '\*', '/'):                      print("Invalid action. Please enter +, -, \*, or /.")                      continue                  return a, b, action              except ValueError:                  print('Invalid input. Please enter valid numbers.')      #calculation function (catches division by zero exeption)      def calculation(a, b, action):          if action == "+":              print(a + b)          elif action == "-":              print(a - b)          elif action == "\*":              print(a \* b)          elif action == "/":              try:                  result = a / b                  print(result)              except ZeroDivisionError:                  print("Division by zero is not allowed.")      #main      def main():          a, b, action = insertion()          calculation(a, b, action)      while True:          main()  except KeyboardInterrupt:      print('\n\nBye!')      exit |

Посилання на github:

<https://github.com/Aatheeriss/TP-KB-221-Denis-Sazonenko.git>

Звіт до Теми №5

Бібліотеки

**Гра з комп’ютером:**

Створити програму, яка 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:

Імпортував бібліотеку, створив декілька функцій. Використав функцію random для вибору випадкоового значення.

Код програми:

|  |
| --- |
| import random  def user\_choice():  choice1 = input("Enter R for rock, S for scissors, P for paper: ").upper()  match choice1:  case "R":  choice1 = "rock"  case "S":  choice1 = "scissors"  case "P":  choice1 = "paper"  return choice1  options = ["rock", "scissors", "paper"]  def main():  computer\_choice = random.choice(options)  user = user\_choice()  if computer\_choice == user:  print("Draw!")  elif (computer\_choice == "rock" and user == "scissors") or (computer\_choice == "paper" and user == "rock") or (computer\_choice == "scissors" and user == "paper"):  print("Computer chosen: " + computer\_choice)  print("You lose!")  else:  print("Computer chosen: " + computer\_choice)  print("You win!")  while True:  trigger = input("Do you want to continue? [Y/N]?").upper()  if trigger == "Y":  main()  else:  break |

**Конвертер валют:**

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:

Імпортував деякі бібліотеки, використав API від НБУ. Використав GUI бібліотеку tkinter для покращення комфорту роботи з програмою.

Код програми:

|  |
| --- |
| import tkinter as tk  from tkinter import ttk  import requests  #request for response from NBU API  res = requests.get("https://bank.gov.ua/NBUStatService/v1/statdirectory/exchange?json")  def convert\_currency():  currency = currency\_var.get()  amount = float(entry\_amount.get())  for elem in res.json():  if elem["cc"] == currency:  rate = elem["rate"]  break    exchange\_rates = {"EUR": rate, "USD": rate, "PLN": rate}    if currency in exchange\_rates:  result = amount \* exchange\_rates[currency]  result\_label.config(text=f"Result: {result:.2f} {"UAH"}")  else:  result\_label.config(text="Choose currency")  root = tk.Tk()  root.title("Converter")  #styles  style = ttk.Style()  style.configure("TButton", padding=(10, 5, 10, 5))  style.configure("TLabel", padding=(5, 5, 5, 5))  #var for currency type choice  currency\_var = tk.StringVar()  currency\_var.set("EUR") #default value  #frame  frame = ttk.Frame(root, padding=10)  frame.grid(row=0, column=0)  currency\_label = ttk.Label(frame, text="Choose currency:")  currency\_label.grid(row=0, column=0, padx=5, pady=5)  #creates option menu to choose currency, returns chosen currency to "currency\_var"  currency\_option\_menu = ttk.Combobox(frame, textvariable=currency\_var, values=["EUR", "USD", "PLN"])  currency\_option\_menu.grid(row=0, column=1, padx=5, pady=5)  amount\_label = ttk.Label(frame, text="Enter value:")  amount\_label.grid(row=1, column=0, padx=5, pady=5)  #creates field to enter amount of value to calculate  entry\_amount = ttk.Entry(frame)  entry\_amount.grid(row=1, column=1, padx=5, pady=5)  #creates button to proceed the action  convert\_button = ttk.Button(frame, text="Convert", command=convert\_currency)  convert\_button.grid(row=2, column=0, columnspan=2, padx=5, pady=10)  result\_label = ttk.Label(frame, text="")  result\_label.grid(row=3, column=0, columnspan=2, padx=5, pady=5)  root.mainloop() |

Посилання на github:

<https://github.com/Aatheeriss/TP-KB-221-Denis-Sazonenko.git>

**Модулі для калькулятора**

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:

Переніс функції калькулятора в окрремі файли. Імпортував в основному файлі.

Код functions.py:

|  |
| --- |
| def addNums(a, b):      return a + b  def subtractNums(a, b):      return a - b  def multiplicateNums(a, b):      return a \* b  def divideNums(a, b):      try:          return a / b      except ZeroDivisionError:          print("Division by zero is not allowed!") |

Код operations.py:

|  |
| --- |
| def getOperationsData():      while True:              try:                  a = float(input("Enter the first number: "))                  b = float(input("Enter the second number: "))                  action = input("Enter action (+, -, \*, /): ").strip()                  if action not in ('+', '-', '\*', '/'):                      print("Invalid action. Please enter +, -, \*, or /.")                      continue                  return a, b, action              except ValueError:                  print('Invalid input. Please enter valid numbers.') |

Код calc.py:

|  |
| --- |
| import functions  import operations  while True:      continue\_check = (input('Do you want to continue?[Y/N]')).upper()      if continue\_check == "Y":          a, b, action = operations.getOperationsData()      match action:          case "+":              result = functions.addNums(a, b)          case "-":              result = functions.subtractNums(a, b)          case "\*":              result = functions.multiplicateNums(a, b)          case "/":              result = functions.divideNums(a, b)      print(result) |

Посилання на github:

<https://github.com/Aatheeriss/TP-KB-221-Denis-Sazonenko.git>

Звіт до Теми №6

Робота з файлами

**Калькулятор з логуванням:**

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:

Використав бібліотеку logging для покращеного вигляду логів. Один з атрибутів що приймає logging.basicConfig є файл до якого записувати логи. До логів також записується час у який відбулася подія. Тип помилки або інформація.

Код functions.py:

|  |
| --- |
| import logging  logging.basicConfig(level=logging.INFO, filename="topic\_06\calculator\_with\_logging\py\_log.log",filemode="w",                      format="%(asctime)s %(levelname)s %(message)s")  def addNums(a, b):      result = a + b      logging.info(f"User recieved a result of {result}")      return result  def subtractNums(a, b):      result = a - b      logging.info(f"User recieved a result of {result}")      return result  def multiplicateNums(a, b):      result = a\*b      logging.info(f"User recieved a result of {result}")      return result  def divideNums(a, b):      try:          result = a / b          logging.info(f"User recieved a result of {result}")          return result        except ZeroDivisionError:          logging.error("ZeroDivisionError", exc\_info=True)          print("Division by zero is not allowed!") |

Код operations.py:

|  |
| --- |
| import logging  logging.basicConfig(level=logging.INFO, filename="topic\_06\calculator\_with\_logging\py\_log.log",filemode="w",                      format="%(asctime)s %(levelname)s %(message)s")  def getOperationsData():      while True:              def getNums():                  a = float(input("Enter the first number: "))                  b = float(input("Enter the second number: "))                  return a, b              try:                  a, b = getNums()                  action = input("Enter action (+, -, \*, /): ").strip()                  logging.info(f"User entered the values of a and b: {a} and {b}, action: {action}.")                  if action not in ('+', '-', '\*', '/'):                      print("Invalid action. Please enter +, -, \*, or /.")                      logging.info(f"User entered invalid action. Retrying.")                      continue                  return a, b, action              except ValueError:                  print('Invalid input. Please enter valid numbers.')                  logging.error("ValueError", exc\_info=True) |

Код calc.py залишився незміненим з topic\_05.

**Сортування списку з lambda-функцією:**

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 програму що зчитує список з файлу та сортує з атрибутом “key”, якому присвоюється lambda-функція.

Код програми:

|  |
| --- |
| students = []  with open("topic\_06\students.txt") as file:      for line in file:          name, mark = line.rstrip().split(",")            student = {}          student["name"] = name          student["mark"] = mark          students.append(student)  for student in sorted(students, key=lambda student: student["mark"]):      print(f"Name is {student['name']} mark is {student['mark']}") |

Звіт до Теми №7

ООП

**Методи класу:**

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

Хід виконання завдання:

\_\_init\_\_ є конструктором класу і викликається автоматично при створенні нового об’єкта класу.

\_\_str\_\_ є методом який повертає рядок, який представляє об’єкт у зручному для читання форматі.

Приклад використання:

|  |
| --- |
| class Car:      def \_\_init\_\_(self, manufacturer, model, year):          self.manufacturer = manufacturer          self.model = model          self.year = year      def \_\_str\_\_(self):          return f"{self.year} {self.manufacturer} {self.model}"  my\_car = Car("Hyondai", "Tucson", 2008)  print(my\_car) |

**Класс 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:

Створив класс, створив список з об’єктів класу. Відсортував список за ключем age.

Код програми:

|  |
| --- |
| class Student:      def \_\_init\_\_(self, name, age):          self.name = name          self.age = age    students = [      Student("Artem", 18),      Student("Denis", 19),      Student("Misha", 17),      Student("Anna", 16)  ]  sorted\_students = sorted(students, key=lambda student: student.age)  for student in sorted\_students:      print(f"Student: {student.name}, Age: {student.age}") |

**Калькулятор з ООП:**

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

Хід виконання завдання:

Переписав модулі калькулятора з додаванням класів, також дещо змінився й основний файл.

Код основного файлу:

|  |
| --- |
| import operations  import functions  import logging  logging.basicConfig(      level=logging.INFO,      filename="topic\_07/calculator/py\_log.log",      filemode="w",      format="%(asctime)s %(levelname)s %(message)s"  )  while True:      a, b, action = operations.Operations.operationsInput()      if a is None or b is None or action is None:          continue      result = None      if action == "+":          result = functions.Functions.add(a, b)      elif action == "-":          result = functions.Functions.subtract(a, b)      elif action == "\*":          result = functions.Functions.multiply(a, b)      elif action == "/":          result = functions.Functions.divide(a, b)      else:          print("Unsupported operation.")          continue      def main():          print("Result: " + str(result))          logging.info(f"User entered {a}, {b} and '{action}'; recieved the result: {result}")      main()      trigger = input("Continue? [Y/N]").upper()      if trigger != "Y":          break |

Код модуля operations.py:

|  |
| --- |
| import logging  logging.basicConfig(level=logging.INFO, filename="topic\_07\calculator\py\_log.log",filemode="w",                      format="%(asctime)s %(levelname)s %(message)s")  class Operations:      def operationsInput():          try:              a = float(input("Enter 1 number: "))              b = float(input("Enter 2 number: "))          except ValueError:              logging.error(ValueError, exc\_info=True)              print("Wrong number, try again!")              return None, None, None            action = input("Enter action (+, -, \*, /): ")          if action not in "+-/\*":              logging.info("User entered wrong action.")              print("Wrong action, try again!")              return None, None, None          return a, b, action |

Код модуля functions.py:

|  |
| --- |
| import logging  logging.basicConfig(level=logging.INFO, filename="topic\_07\calculator\py\_log.log",filemode="w",                      format="%(asctime)s %(levelname)s %(message)s")  class Functions:      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):          try:              return a / b          except ZeroDivisionError:              print("Division by zero is not allowed!!!")              logging.error("ZeroDivisionError", exc\_info=True) |