Міністерство освіти і науки України

Національний університет «Чернігівська політехніка»

Навчально-науковий інститут електронних та інформаційних технологій

Кафедра кібербезпеки та математичного моделювання

**Розрахункова-графічна робота**

**з дисципліни**

**«Технології програмування на мові Python»**

виконала

студентка групи КБ-232

Ребенок Є. О.

перевірив

Дюба І.М\_\_\_\_\_\_\_\_\_

2024

**Зміст**

[**Звіт до Теми №1** 3](#_Toc185339549)

[**Звіт до Теми №2** 7](#_Toc185339550)

[**Звіт до Теми №3** 15](#_Toc185339551)

[**Звіт до Теми №4** 33](#_Toc185339552)

[**Звіт до Теми №5** 37](#_Toc185339553)

[**Звіт до Теми №6** 47](#_Toc185339554)

[**Звіт до Теми №7** 54](#_Toc185339555)

# **Звіт до Теми №1**

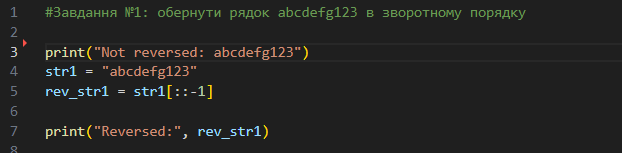
Функції та змінні

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:



Вивід:



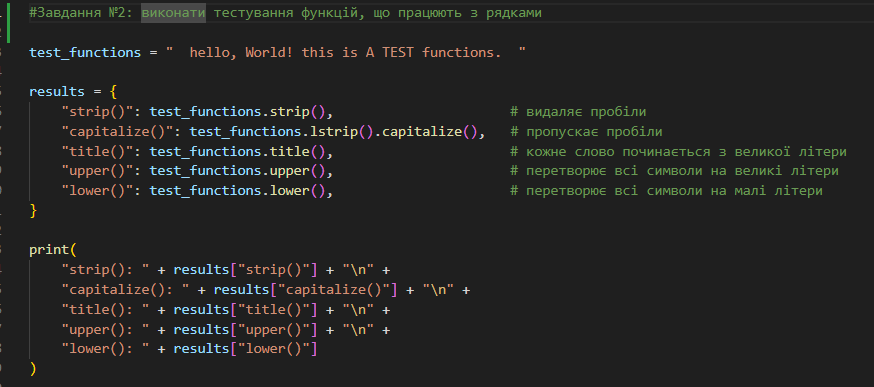
Текст програми:

|  |
| --- |
| print("Not reversed: abcdefg123")  str1 = "abcdefg123"  rev\_str1 = str1[::-1]  print("Reversed:", rev\_str1) |

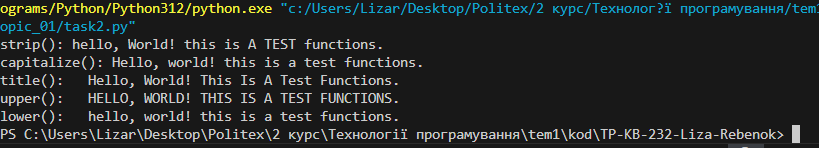
**Робота з рядками**

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

Хід виконання завдання:



Вивід:



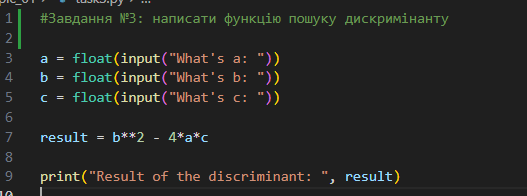
Текст програми:

|  |
| --- |
| test\_functions = " hello, World! this is A TEST functions. "  results = {  "strip()": test\_functions.strip(), # видаляє пробіли  "capitalize()": test\_functions.lstrip().capitalize(),# пропускає пробіли  "title()": test\_functions.title(), # кожне слово починається з великої літери  "upper()": test\_functions.upper(), # перетворює всі символи на великі літери  "lower()": test\_functions.lower(), # перетворює всі символи на малі літери  }  print(  "strip(): " + results["strip()"] + "\n" +  "capitalize(): " + results["capitalize()"] + "\n" +  "title(): " + results["title()"] + "\n" +  "upper(): " + results["upper()"] + "\n" +  "lower(): " + results["lower()"]  ) |

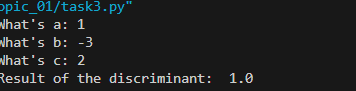
**Пошук дискримінанта квадратного рівняння**

Написати функцію пошуку дискримінанту квадратного рівняння.

Хід виконання завдання:



Вивід:



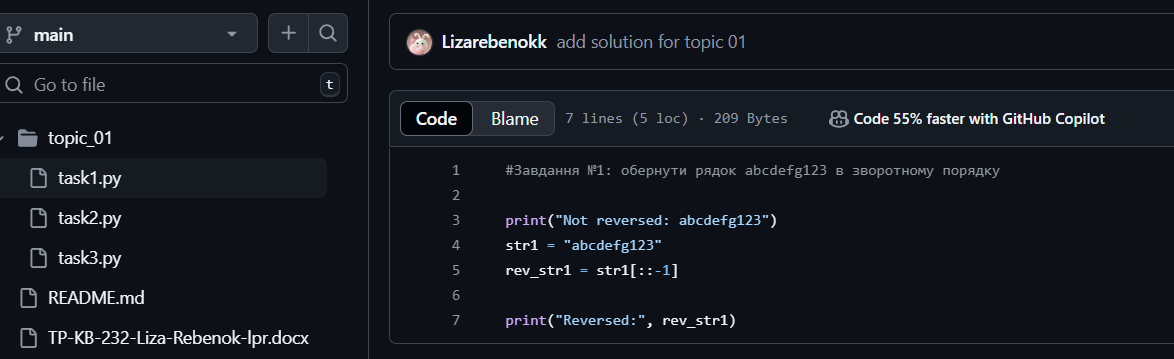
Текст програми:

|  |
| --- |
| a = float(input("What's a: "))  b = float(input("What's b: "))  c = float(input("What's c: "))  result = b\*\*2 - 4\*a\*c  print("Result of the discriminant: ", result) |

Посилання на github:

<https://github.com/Lizarebenokk>

Знімок екрану з посилання на github:



# **Звіт до Теми №2**

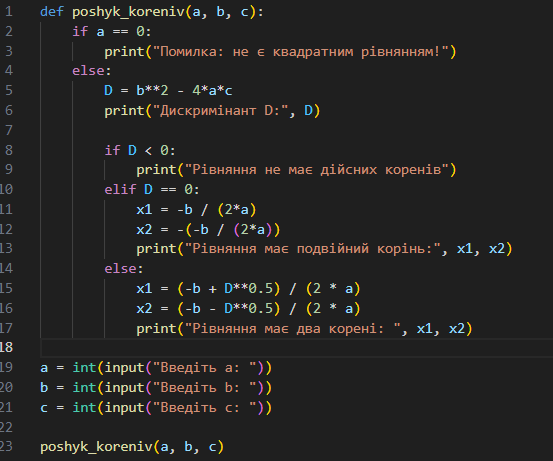
Умовний перехід

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

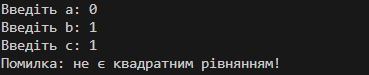
**Написання функцій**

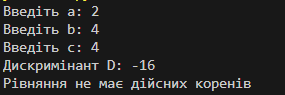
Написати функцію пошуку коренів квадратного рівняння використовуючи функцію розрахунку дискримінанту з попередньої теми та умовні переходи.

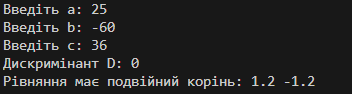
Хід виконання завдання:

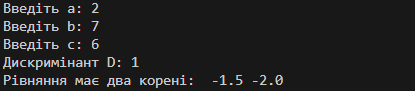


Вивід:









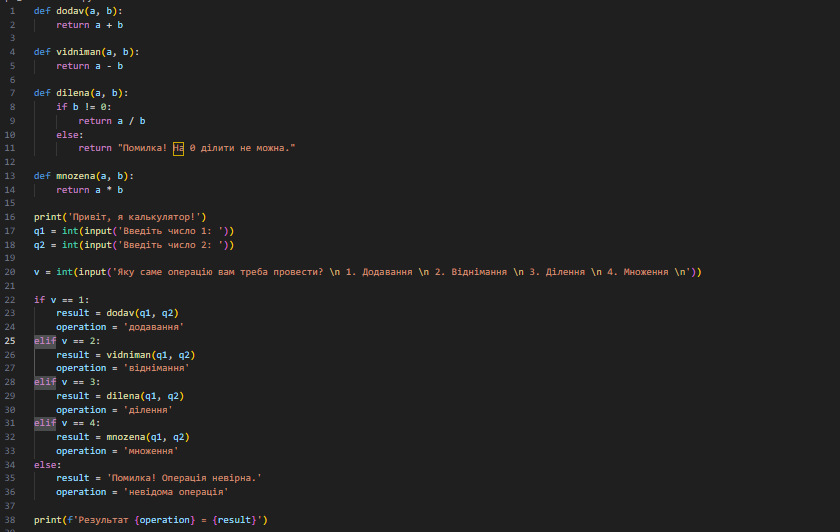
Текст програми:

|  |
| --- |
| def poshyk\_koreniv(a, b, c):  if a == 0:  print("Помилка: не є квадратним рівнянням!")  else:  D = b\*\*2 - 4\*a\*c  print("Дискримінант D:", D)  if D < 0:  print("Рівняння не має дійсних коренів")  elif D == 0:  x1 = -b / (2\*a)  x2 = -(-b / (2\*a))  print("Рівняння має подвійний корінь:", x1, x2)  else:  x1 = (-b + D\*\*0.5) / (2 \* a)  x2 = (-b - D\*\*0.5) / (2 \* a)  print("Рівняння має два корені: ", x1, x2)  a = int(input("Введіть a: "))  b = int(input("Введіть b: "))  c = int(input("Введіть c: "))    poshyk\_koreniv(a, b, c) |

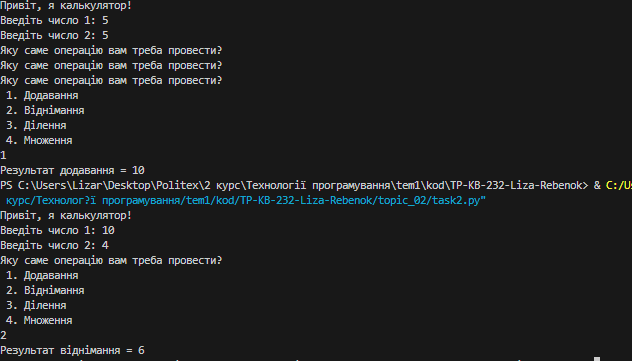
**Написання програми калькулятор**

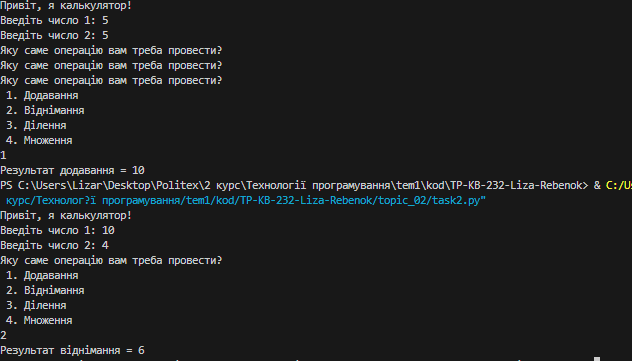
Написати програму калькулятор використовуючи if else конструкцію. Кожна операція має бути виконана в окремій функції.

Хід виконання завдання:



Вивід:



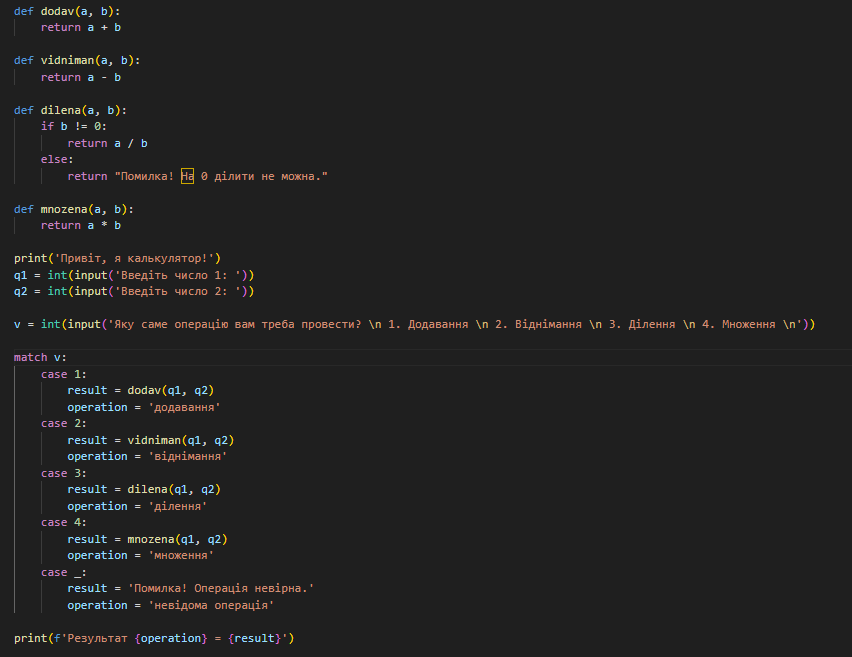


Текст програми:

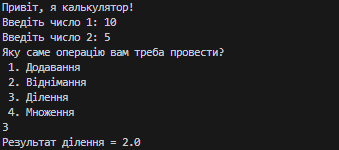
|  |
| --- |
| def dodav(a, b):  return a + b  def vidniman(a, b):  return a - b  def dilena(a, b):  if b != 0:  return a / b  else:  return "Помилка! На 0 ділити не можна."  def mnozena(a, b):  return a \* b  print('Привіт, я калькулятор!')  q1 = int(input('Введіть число 1: '))  q2 = int(input('Введіть число 2: '))  v = int(input('Яку саме операцію вам треба провести? \n 1. Додавання \n 2. Віднімання \n 3. Ділення \n 4. Множення \n'))  if v == 1:  result = dodav(q1, q2)  operation = 'додавання'  elif v == 2:  result = vidniman(q1, q2)  operation = 'віднімання'  elif v == 3:  result = dilena(q1, q2)  operation = 'ділення'  elif v == 4:  result = mnozena(q1, q2)  operation = 'множення'  else:  result = 'Помилка! Операція невірна.'  operation = 'невідома операція'  print(f'Результат {operation} = {result}') |

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

Хід виконання завдання:



Вивід:



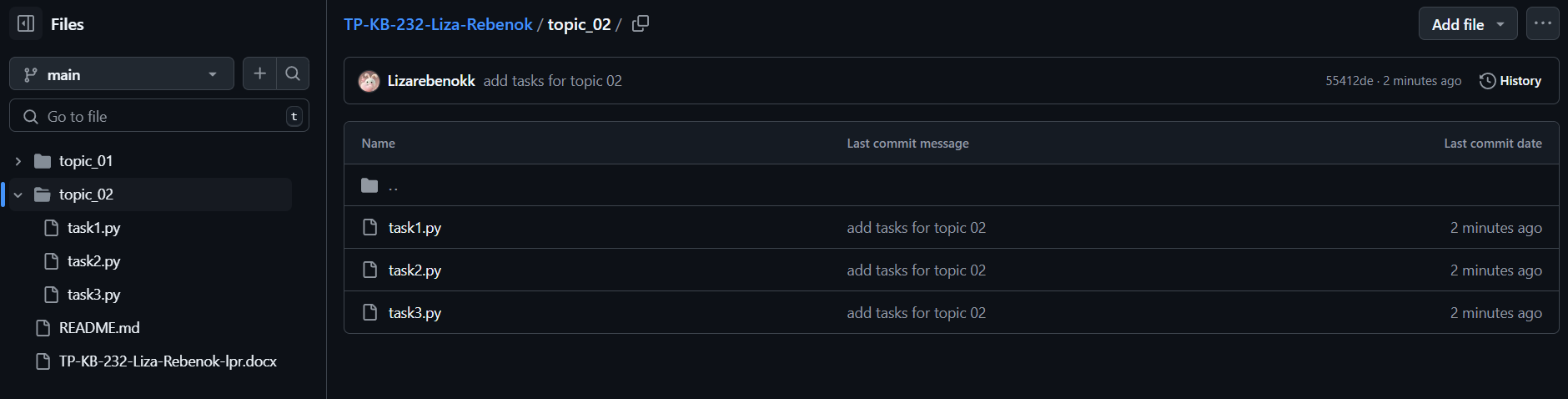
Текст програми:

|  |
| --- |
| def dodav(a, b):  return a + b  def vidniman(a, b):  return a - b  def dilena(a, b):  if b != 0:  return a / b  else:  return "Помилка! На 0 ділити не можна."  def mnozena(a, b):  return a \* b  print('Привіт, я калькулятор!')  q1 = int(input('Введіть число 1: '))  q2 = int(input('Введіть число 2: '))  v = int(input('Яку саме операцію вам треба провести? \n 1. Додавання \n 2. Віднімання \n 3. Ділення \n 4. Множення \n'))  match v:  case 1:  result = dodav(q1, q2)  operation = 'додавання'  case 2:  result = vidniman(q1, q2)  operation = 'віднімання'  case 3:  result = dilena(q1, q2)  operation = 'ділення'  case 4:  result = mnozena(q1, q2)  operation = 'множення'  case \_:  result = 'Помилка! Операція невірна.'  operation = 'невідома операція'  print(f'Результат {operation} = {result}') |

Посилання на github:

<https://github.com/Lizarebenokk>

Знімок екрану з посилання на github:



# **Звіт до Теми №3**

Цикли

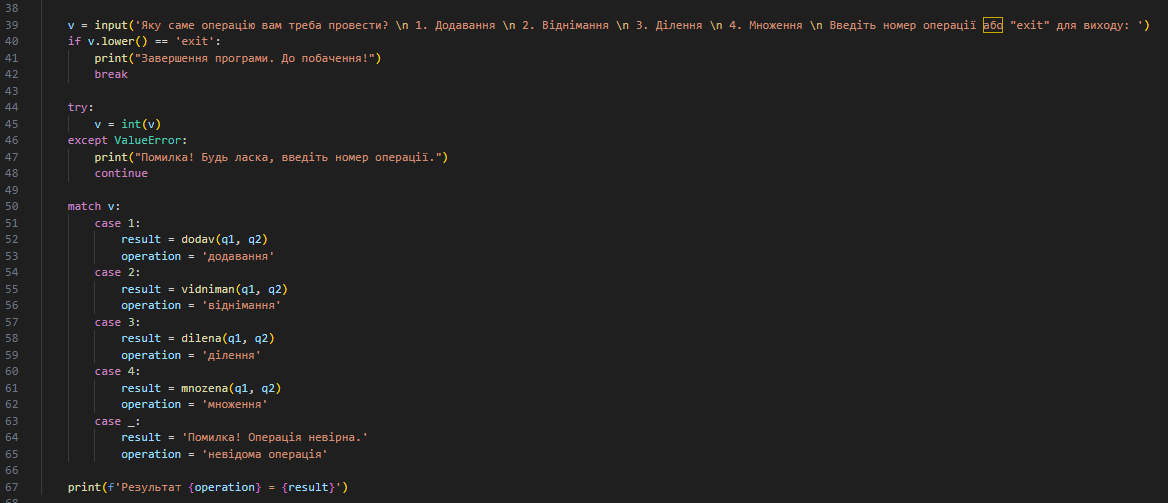
Під час виконання практичного завдання до Теми №3 було надано варіанти рішення до наступних задач:

**Написати програму калькулятор**

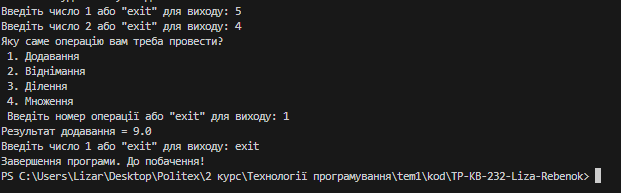
Написати програму калькулятор з постійними запитами на введення нових даних та операцій. За основу взяти програму калькулятор з попередньої теми. Реалізувати механізм завершення програми після отримання відповідної команди.

Хід виконання завдання:





Вивід:



Текст програми:

|  |
| --- |
| def dodav(a, b):  return a + b  def vidniman(a, b):  return a - b  def dilena(a, b):  if b != 0:  return a / b  else:  return "Помилка! На 0 ділити не можна."  def mnozena(a, b):  return a \* b  print('Привіт, я калькулятор!')  while True:  q1 = input('Введіть число 1 або "exit" для виходу: ')  if q1.lower() == 'exit':  print("Завершення програми. До побачення!")  break  try:  q1 = int(q1)  except ValueError:  print("Помилка! Будь ласка, введіть число.")  continue  q2 = input('Введіть число 2 або "exit" для виходу: ')  if q2.lower() == 'exit':  print("Завершення програми. До побачення!")  break  try:  q2 = int(q2)  except ValueError:  print("Помилка! Будь ласка, введіть число.")  continue  v = input('Яку саме операцію вам треба провести? \n 1. Додавання \n 2. Віднімання \n 3. Ділення \n 4. Множення \n Введіть номер операції або "exit" для виходу: ')  if v.lower() == 'exit':  print("Завершення програми. До побачення!")  break  try:  v = int(v)  except ValueError:  print("Помилка! Будь ласка, введіть номер операції.")  continue  match v:  case 1:  result = dodav(q1, q2)  operation = 'додавання'  case 2:  result = vidniman(q1, q2)  operation = 'віднімання'  case 3:  result = dilena(q1, q2)  operation = 'ділення'  case 4:  result = mnozena(q1, q2)  operation = 'множення'  case \_:  result = 'Помилка! Операція невірна.'  operation = 'невідома операція'  print(f'Результат {operation} = {result}') |

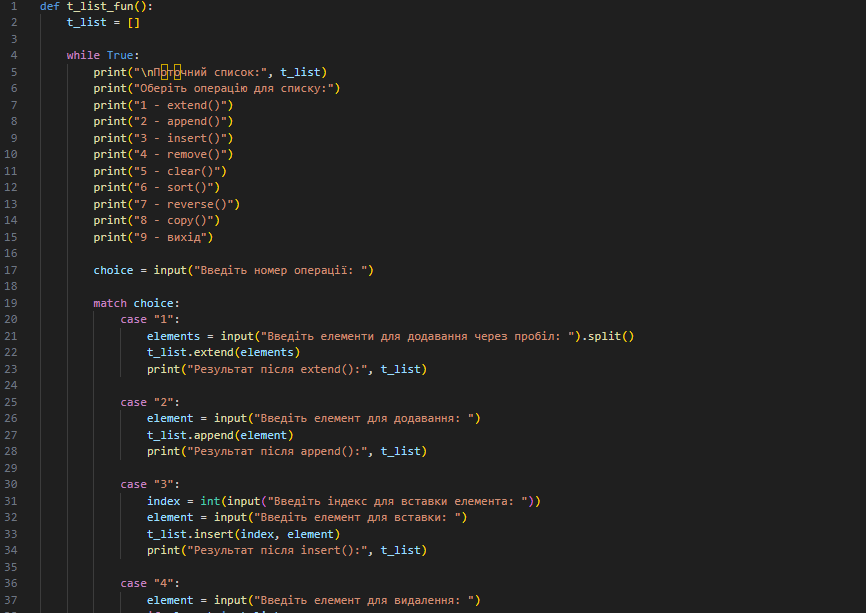
**Написати програму тестування функцій списків**

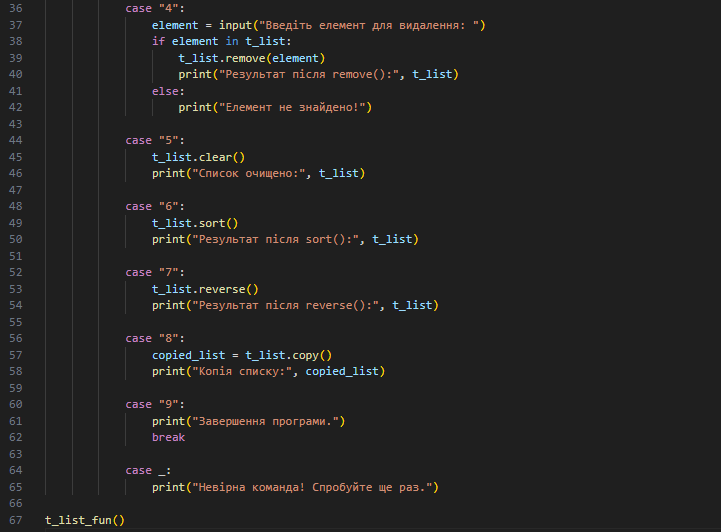
Написати програму тестування функцій списків таких як: extend(), append(), insert(id, val), remove(val), clear(), sort(), reverse(), copy().

Теоретичні відомості

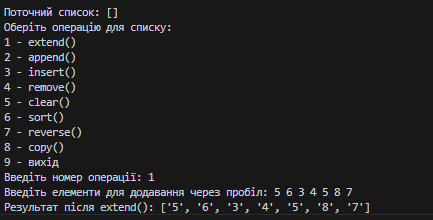
* extend(iterable) – додає декілька елементів у список;
* append(value) - додає один елемент до кінця списку;
* insert(index, value) - вставляє елемент на вказану позицію в списку;
* remove(value) - видаляє елемент за вказаним значенням.
* clear() – очищає весь список.
* sort(reverse=False) - сортує список за зростанням.
* reverse() – перевертає порядок елементів у списку на зворотній.
* copy() - створює поверхневу копію списку.

Хід виконання завдання:

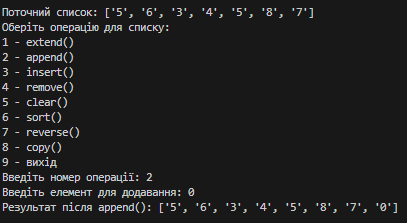




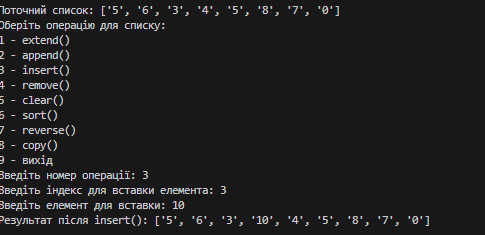
Вивід extend():



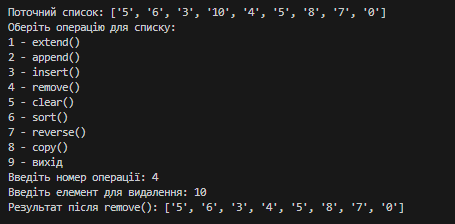
Вивід append():



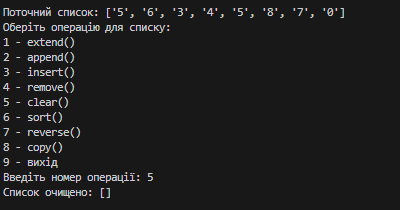
Вивід insert(id, val):



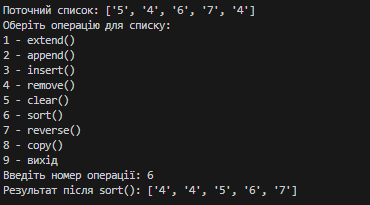
Вивід remove(val):



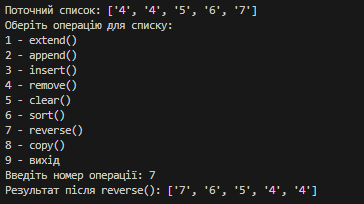
Вивід clear():



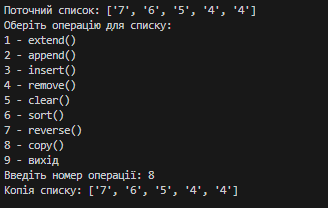
Вивід sort():



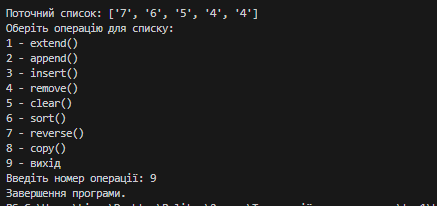
Вивід reverse():



Вивід copy():



Вивід exit:



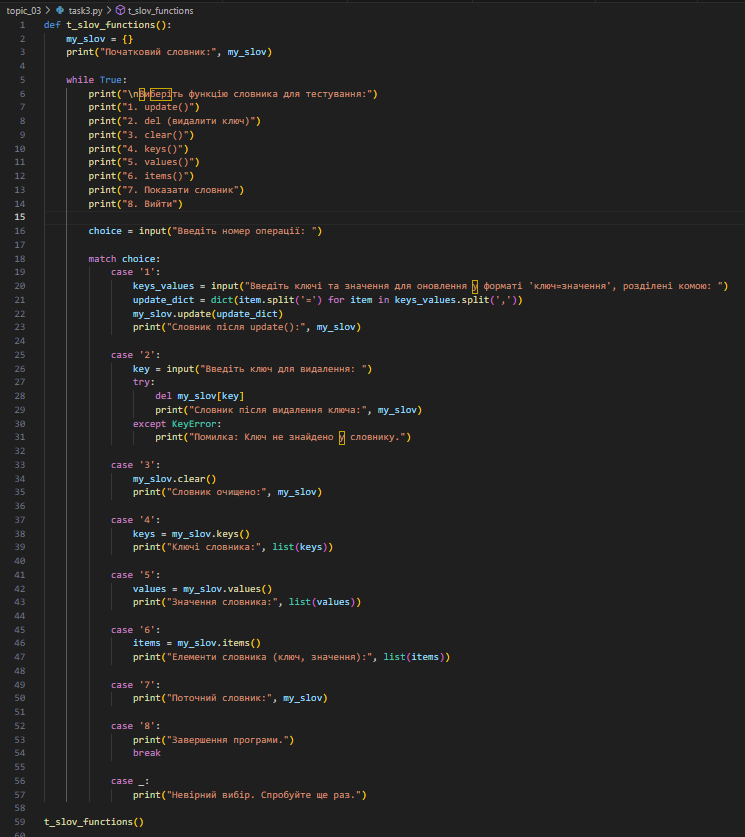
Текст програми:

|  |
| --- |
| def t\_list\_fun():  t\_list = []  while True:  print("\nПоточний список:", t\_list)  print("Оберіть операцію для списку:")  print("1 - extend()")  print("2 - append()")  print("3 - insert()")  print("4 - remove()")  print("5 - clear()")  print("6 - sort()")  print("7 - reverse()")  print("8 - copy()")  print("9 - вихід")  choice = input("Введіть номер операції: ")  match choice:  case "1":  elements = input("Введіть елементи для додавання через пробіл: ").split()  t\_list.extend(elements)  print("Результат після extend():", t\_list)  case "2":  element = input("Введіть елемент для додавання: ")  t\_list.append(element)  print("Результат після append():", t\_list)  case "3":  index = int(input("Введіть індекс для вставки елемента: "))  element = input("Введіть елемент для вставки: ")  t\_list.insert(index, element)  print("Результат після insert():", t\_list)  case "4":  element = input("Введіть елемент для видалення: ")  if element in t\_list:  t\_list.remove(element)  print("Результат після remove():", t\_list)  else:  print("Елемент не знайдено!")  case "5":  t\_list.clear()  print("Список очищено:", t\_list)  case "6":  t\_list.sort()  print("Результат після sort():", t\_list)  case "7":  t\_list.reverse()  print("Результат після reverse():", t\_list)  case "8":  copied\_list = t\_list.copy()  print("Копія списку:", copied\_list)  case "9":  print("Завершення програми.")  break  case \_:  print("Невірна команда! Спробуйте ще раз.")  t\_list\_fun() |

**Написати програму калькулятор**

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

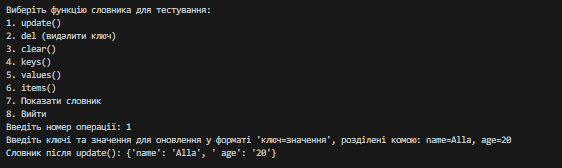
Хід виконання завдання:

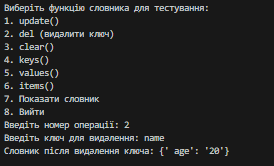


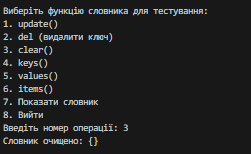
Теоретичні відомості

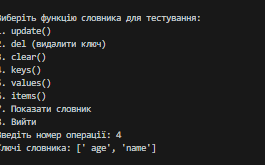
* update(other) - додає/оновлює елементи з іншого словника;
* del dict[key] - видаляє елемент за вказаним ключем;
* clear() - очищає словник, видаляючи всі елементи;
* values() - повертає всі значення словника;
* items() - повертає пари (ключ, значення) з словника.

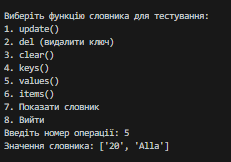
Вивід:

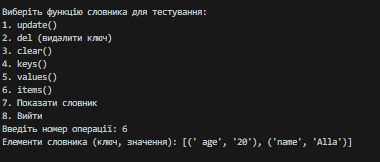


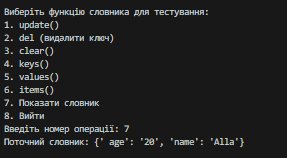


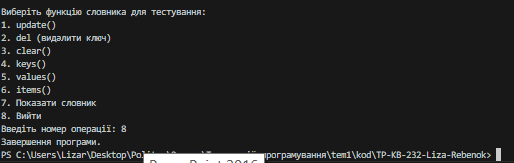












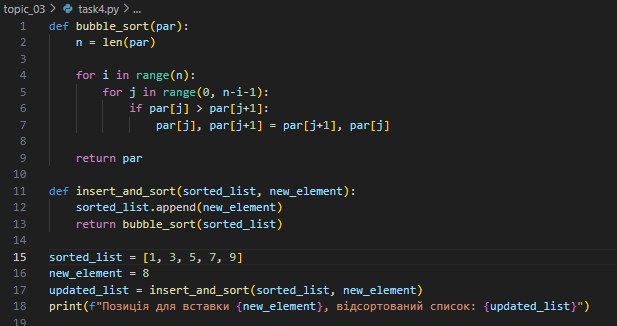
Текст програми:

|  |
| --- |
| def t\_slov\_functions():  my\_slov = {}  print("Початковий словник:", my\_slov)  while True:  print("\nВиберіть функцію словника для тестування:")  print("1. update()")  print("2. del (видалити ключ)")  print("3. clear()")  print("4. keys()")  print("5. values()")  print("6. items()")  print("7. Показати словник")  print("8. Вийти")  choice = input("Введіть номер операції: ")  match choice:  case '1':  keys\_values = input("Введіть ключі та значення для оновлення у форматі 'ключ=значення', розділені комою: ")  update\_dict = dict(item.split('=') for item in keys\_values.split(','))  my\_slov.update(update\_dict)  print("Словник після update():", my\_slov)  case '2':  key = input("Введіть ключ для видалення: ")  try:  del my\_slov[key]  print("Словник після видалення ключа:", my\_slov)  except KeyError:  print("Помилка: Ключ не знайдено у словнику.")  case '3':  my\_slov.clear()  print("Словник очищено:", my\_slov)  case '4':  keys = my\_slov.keys()  print("Ключі словника:", list(keys))  case '5':  values = my\_slov.values()  print("Значення словника:", list(values))  case '6':  items = my\_slov.items()  print("Елементи словника (ключ, значення):", list(items))  case '7':  print("Поточний словник:", my\_slov)  case '8':  print("Завершення програми.")  break  case \_:  print("Невірний вибір. Спробуйте ще раз.")  t\_slov\_functions() |

**Написати функцію пошуку позиції для вставки нового елементу в список**

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

Хід виконання завдання:



Вивід:





Текст програми:

|  |
| --- |
| def bubble\_sort(par):  n = len(par)    for i in range(n):  for j in range(0, n-i-1):  if par[j] > par[j+1]:  par[j], par[j+1] = par[j+1], par[j]    return par  def insert\_and\_sort(sorted\_list, new\_element):  sorted\_list.append(new\_element)  return bubble\_sort(sorted\_list)  sorted\_list = [1, 3, 5, 7, 9]  new\_element = 4  updated\_list = insert\_and\_sort(sorted\_list, new\_element)  print(f"Позиція для вставки {new\_element}, відсортований список: {updated\_list}") |

Посилання на github:

<https://github.com/Lizarebenokk>

# **Звіт до Теми №4**

Виняткові ситуації

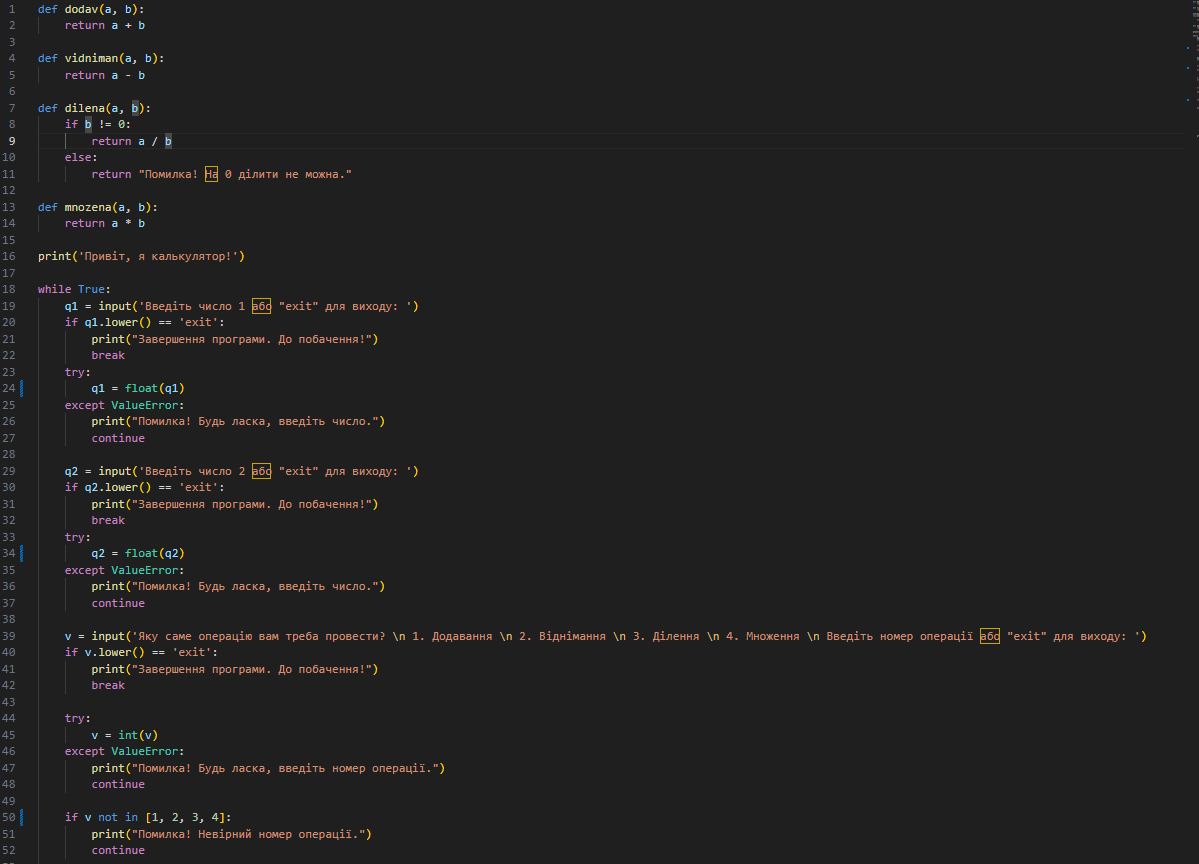
Під час виконання практичного завдання до Теми №4 було надано варіанти рішення до наступних задач:

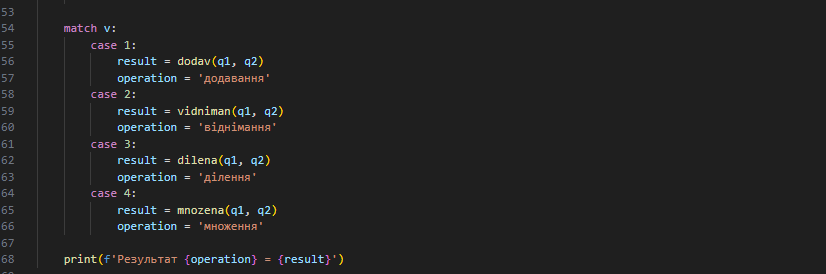
**1) Розширити програму калькулятор функцією запитів даних для виконання операцій від користувача, що обробляє виняткові ситуації.**

**2) Розширити функцію ділення обробкою виняткової ситуації ділення но нуль**

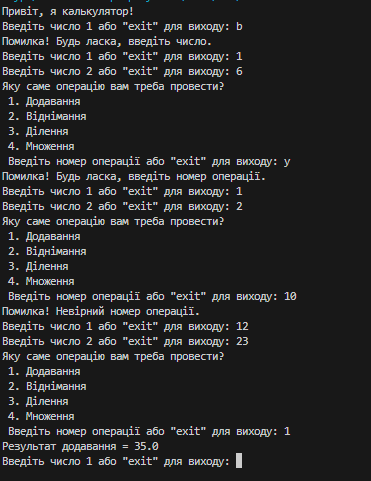
**3) Ознайомитись зі списком виняткових ситуацій за посиланням** [**https://docs.python.org/3/library/exceptions.html**](https://docs.python.org/3/library/exceptions.html)

Хід виконання завдання:





Вивід:



Текст програми:

|  |
| --- |
| def dodav(a, b):  return a + b  def vidniman(a, b):  return a - b  def dilena(a, b):  if b != 0:  return a / b  else:  return "Помилка! На 0 ділити не можна."  def mnozena(a, b):  return a \* b  print('Привіт, я калькулятор!')  while True:  q1 = input('Введіть число 1 або "exit" для виходу: ')  if q1.lower() == 'exit':  print("Завершення програми. До побачення!")  break  try:  q1 = float(q1)  except ValueError:  print("Помилка! Будь ласка, введіть число.")  continue  q2 = input('Введіть число 2 або "exit" для виходу: ')  if q2.lower() == 'exit':  print("Завершення програми. До побачення!")  break  try:  q2 = float(q2)  except ValueError:  print("Помилка! Будь ласка, введіть число.")  continue  v = input('Яку саме операцію вам треба провести? \n 1. Додавання \n 2. Віднімання \n 3. Ділення \n 4. Множення \n Введіть номер операції або "exit" для виходу: ')  if v.lower() == 'exit':  print("Завершення програми. До побачення!")  break  try:  v = int(v)  except ValueError:  print("Помилка! Будь ласка, введіть номер операції.")  continue  if v not in [1, 2, 3, 4]:  print("Помилка! Невірний номер операції.")  continue  match v:  case 1:  result = dodav(q1, q2)  operation = 'додавання'  case 2:  result = vidniman(q1, q2)  operation = 'віднімання'  case 3:  result = dilena(q1, q2)  operation = 'ділення'  case 4:  result = mnozena(q1, q2)  operation = 'множення'  print(f'Результат {operation} = {result}') |

Посилання на github:

<https://github.com/Lizarebenokk>

# **Звіт до Теми №5**

Бібліотеки

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

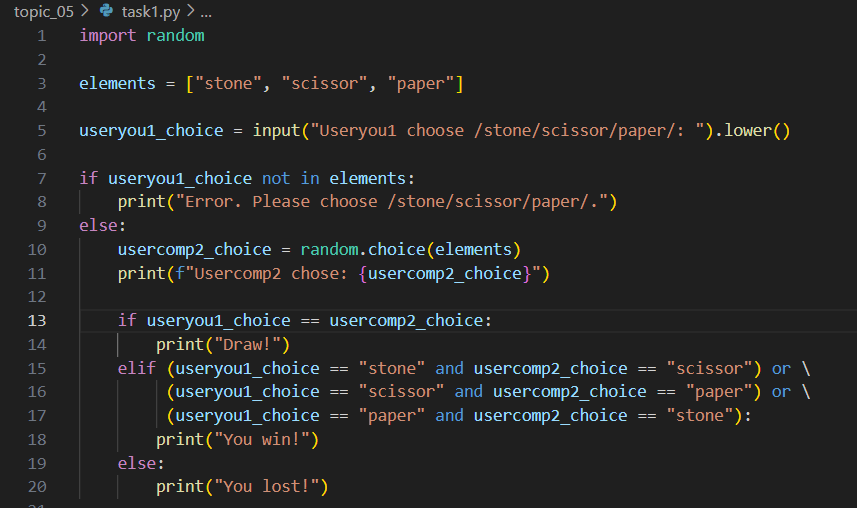
**1) Гра з комп’ютером: камінь, ножиці, папір. Програма виконує запит від користувача на введення одного із значень ["stone", "scissor", "paper"]. Наступним кроком, використовуючи модуль random, програма у випадковому порядку вибирає одне із значень ["stone", "scissor", "paper"]. В залежності від умови, що камінь перемагає ножиці, ножиці перемагають папір, а папір перемагає камінь визначити переможця.**

**2) 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.**

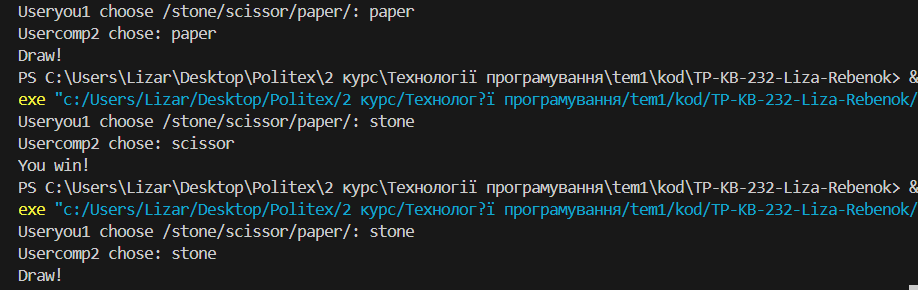
**3) Використання модулів для програми калькулятор. Функції додавання, віднімання, множення та ділення перенести в файл functions.py. Функції запиту на введення даних для операцій та самих операцій перемістити в файл operations.py. Програму калькулятор реалізувати в файлі calc.py, до якого підключають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

**Гра з комп’ютером: камінь, ножиці, папір**

Хід виконання завдання:



Вивід:

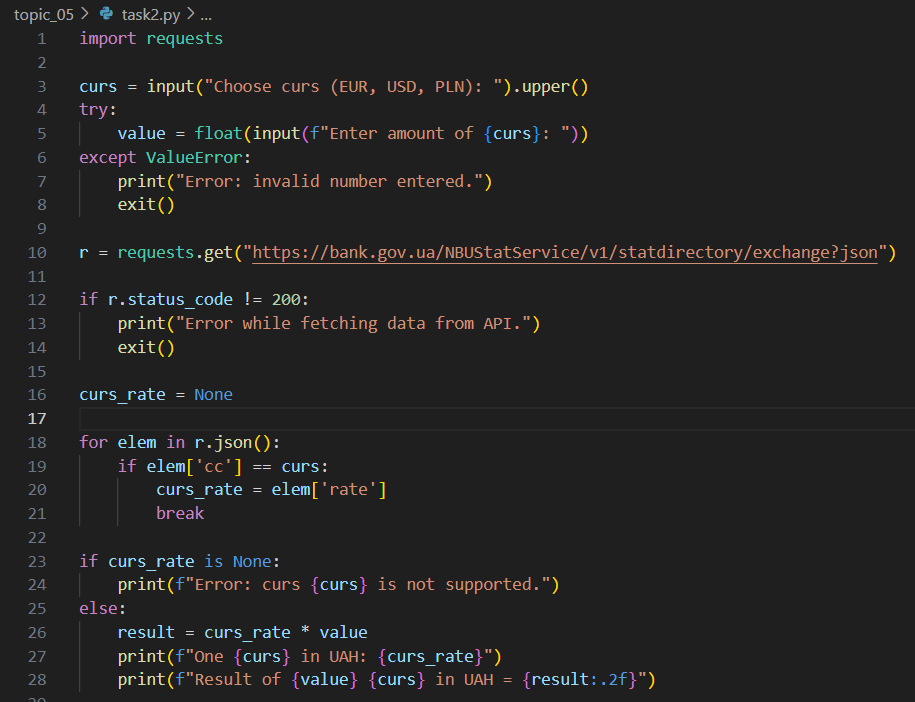


Текст програми:

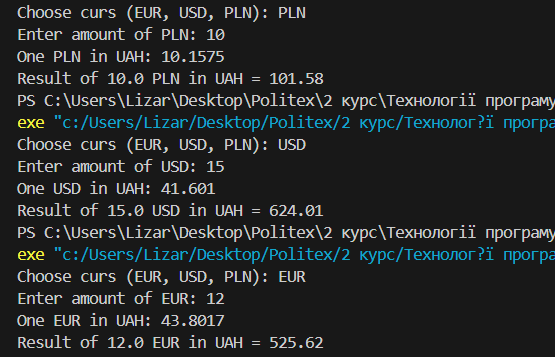
|  |
| --- |
| import random  elements = ["stone", "scissor", "paper"]  useryou1\_choice = input("Useryou1 choose /stone/scissor/paper/: ").lower()  if useryou1\_choice not in elements:  print("Error. Please choose /stone/scissor/paper/.")  else:  usercomp2\_choice = random.choice(elements)  print(f"Usercomp2 chose: {usercomp2\_choice}")  if useryou1\_choice == usercomp2\_choice:  print("Draw!")  elif (useryou1\_choice == "stone" and usercomp2\_choice == "scissor") or \  (useryou1\_choice == "scissor" and usercomp2\_choice == "paper") or \  (useryou1\_choice == "paper" and usercomp2\_choice == "stone"):  print("You win!")  else:  print("You lost!") |

**Програма конвертування іноземної валюти в українську гривню**

Хід виконання завдання:



Вивід:



Текст програми:

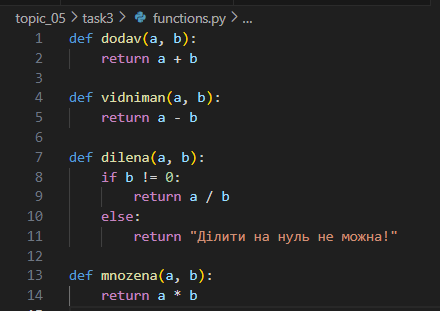
|  |
| --- |
| import requests  curs = input("Choose curs (EUR, USD, PLN): ").upper()  try:  value = float(input(f"Enter amount of {curs}: "))  except ValueError:  print("Error: invalid number entered.")  exit()  r = requests.get("https://bank.gov.ua/NBUStatService/v1/statdirectory/exchange?json")  if r.status\_code != 200:  print("Error while fetching data from API.")  exit()  curs\_rate = None  for elem in r.json():  if elem['cc'] == curs:  curs\_rate = elem['rate']  break  if curs\_rate is None:  print(f"Error: curs {curs} is not supported.")  else:  result = curs\_rate \* value  print(f"One {curs} in UAH: {curs\_rate}")  print(f"Result of {value} {curs} in UAH = {result:.2f}") |

**Використання модулів для програми калькулятор**

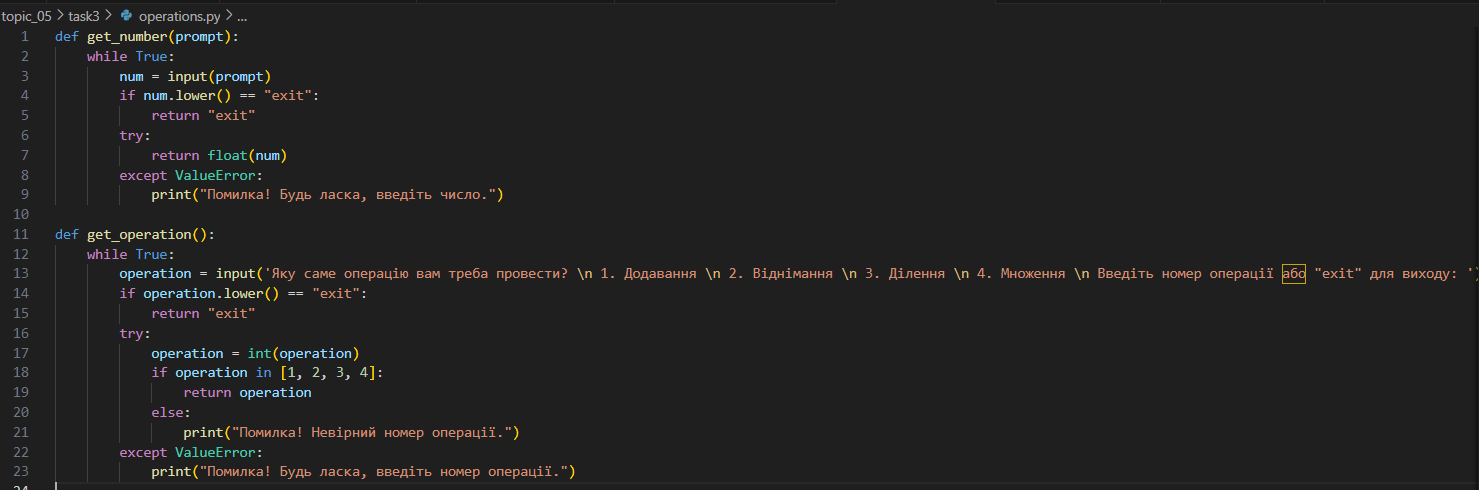
Хід виконання завдання calc.py:



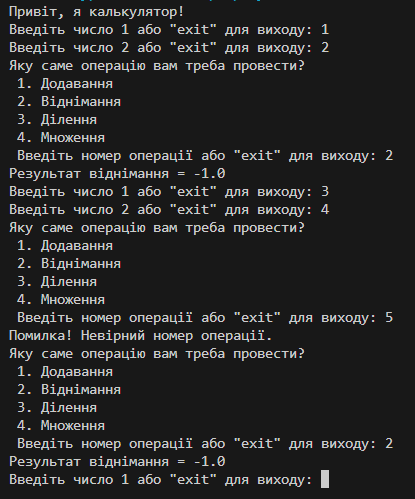
Хід виконання завдання functions.py:



Хід виконання завдання operations.py:



Вивід:



Текст програми calc.py:

|  |
| --- |
| from functions import dodav, vidniman, dilena, mnozena  from operations import get\_number, get\_operation  print('Привіт, я калькулятор!')  while True:  q1 = get\_number('Введіть число 1 або "exit" для виходу: ')  if q1 == "exit":  print("Завершення програми. До побачення!")  break  q2 = get\_number('Введіть число 2 або "exit" для виходу: ')  if q2 == "exit":  print("Завершення програми. До побачення!")  break  operation = get\_operation()  if operation == "exit":  print("Завершення програми. До побачення!")  break  match operation:  case 1:  result = dodav(q1, q2)  operation\_name = 'додавання'  case 2:  result = vidniman(q1, q2)  operation\_name = 'віднімання'  case 3:  result = dilena(q1, q2)  operation\_name = 'ділення'  case 4:  result = mnozena(q1, q2)  operation\_name = 'множення'  print(f'Результат {operation\_name} = {result}') |

Текст програми functions.py:

|  |
| --- |
| def dodav(a, b):  return a + b  def vidniman(a, b):  return a - b  def dilena(a, b):  if b != 0:  return a / b  else:  return "Ділити на нуль не можна!"  def mnozena(a, b):  return a \* b |

Текст програми operations.py:

|  |
| --- |
| def get\_number(prompt):  while True:  num = input(prompt)  if num.lower() == "exit":  return "exit"  try:  return float(num)  except ValueError:  print("Помилка! Будь ласка, введіть число.")  def get\_operation():  while True:  operation = input('Яку саме операцію вам треба провести? \n 1. Додавання \n 2. Віднімання \n 3. Ділення \n 4. Множення \n Введіть номер операції або "exit" для виходу: ')  if operation.lower() == "exit":  return "exit"  try:  operation = int(operation)  if operation in [1, 2, 3, 4]:  return operation  else:  print("Помилка! Невірний номер операції.")  except ValueError:  print("Помилка! Будь ласка, введіть номер операції.") |

Посилання на github:

<https://github.com/Lizarebenokk>

# **Звіт до Теми №6**

Робота з файлами

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

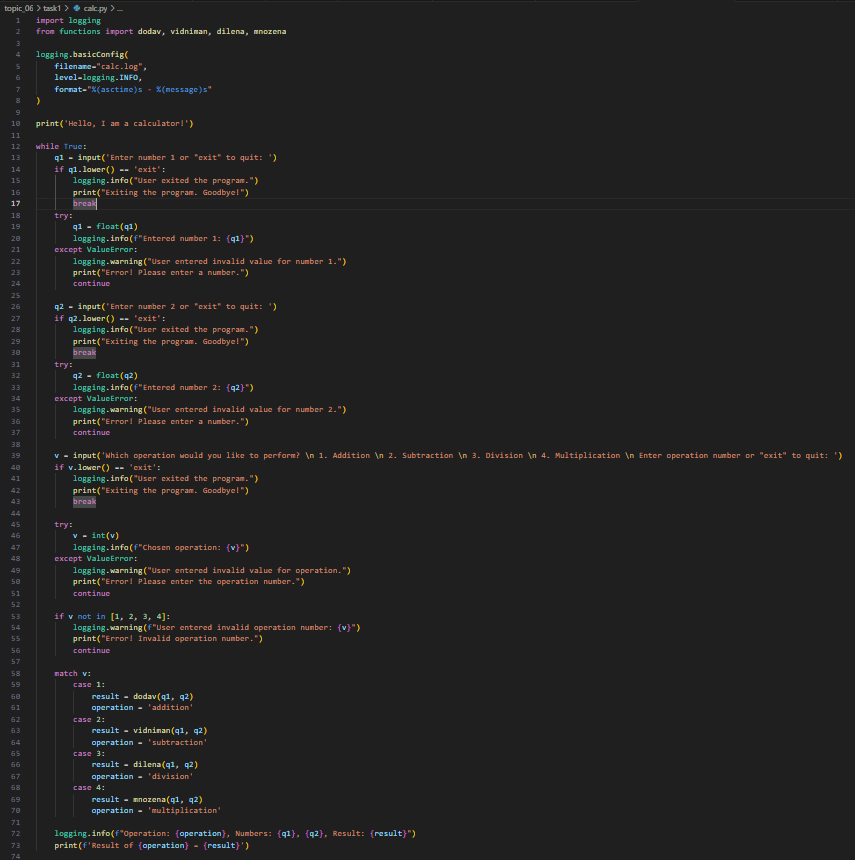
**Попередні умови:** **Реалізована програма калькулятор в файлі calc.py, до якого підключають файл functions.py та operations.py. Інструкції до оформлення вмісту файлів розміщенні в завданні 3 до теми 5.**

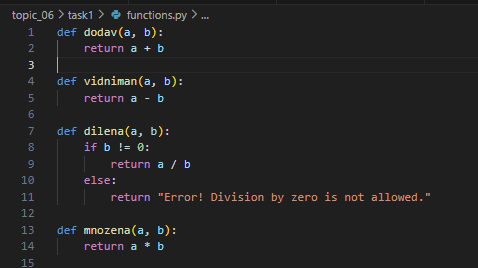
**1) Розробити механізм логування всіх дій, що виконує програма. Забезпечити зберігання інформації про введені данні, виконану операцію та результат виконання операції над даними.**

**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.**

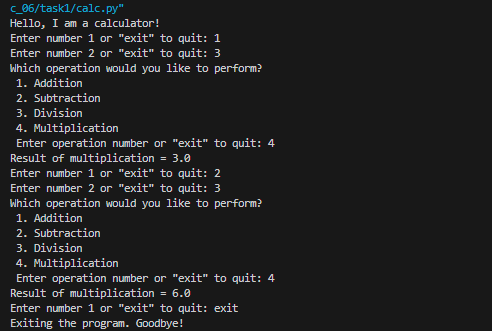
**Розробити механізм логування всіх дій, що виконує програма**

Хід виконання завдання:

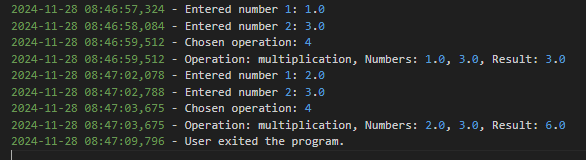




Вивід:



Запис до файлу calc.log:



Текст програми calc.py:

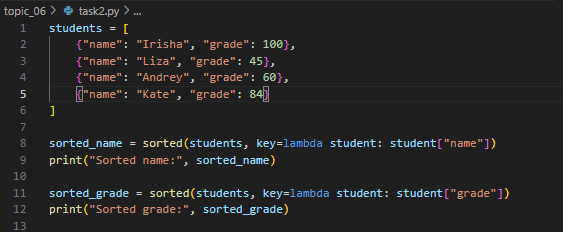
|  |
| --- |
| import logging  from functions import dodav, vidniman, dilena, mnozena  logging.basicConfig(  filename="calc.log",  level=logging.INFO,  format="%(asctime)s - %(message)s"  )  print('Hello, I am a calculator!')  while True:  q1 = input('Enter number 1 or "exit" to quit: ')  if q1.lower() == 'exit':  logging.info("User exited the program.")  print("Exiting the program. Goodbye!")  break  try:  q1 = float(q1)  logging.info(f"Entered number 1: {q1}")  except ValueError:  logging.warning("User entered invalid value for number 1.")  print("Error! Please enter a number.")  continue  q2 = input('Enter number 2 or "exit" to quit: ')  if q2.lower() == 'exit':  logging.info("User exited the program.")  print("Exiting the program. Goodbye!")  break  try:  q2 = float(q2)  logging.info(f"Entered number 2: {q2}")  except ValueError:  logging.warning("User entered invalid value for number 2.")  print("Error! Please enter a number.")  continue  v = input('Which operation would you like to perform? \n 1. Addition \n 2. Subtraction \n 3. Division \n 4. Multiplication \n Enter operation number or "exit" to quit: ')  if v.lower() == 'exit':  logging.info("User exited the program.")  print("Exiting the program. Goodbye!")  break  try:  v = int(v)  logging.info(f"Chosen operation: {v}")  except ValueError:  logging.warning("User entered invalid value for operation.")  print("Error! Please enter the operation number.")  continue  if v not in [1, 2, 3, 4]:  logging.warning(f"User entered invalid operation number: {v}")  print("Error! Invalid operation number.")  continue  match v:  case 1:  result = dodav(q1, q2)  operation = 'addition'  case 2:  result = vidniman(q1, q2)  operation = 'subtraction'  case 3:  result = dilena(q1, q2)  operation = 'division'  case 4:  result = mnozena(q1, q2)  operation = 'multiplication'  logging.info(f"Operation: {operation}, Numbers: {q1}, {q2}, Result: {result}")  print(f'Result of {operation} = {result}') |

Текст програми functions.py:

|  |
| --- |
| def dodav(a, b):  return a + b  def vidniman(a, b):  return a - b  def dilena(a, b):  if b != 0:  return a / b  else:  return "Error! Division by zero is not allowed."  def mnozena(a, b):  return a \* b |

**Сортування списку, використовуючи стандартну функцію sorted()**

Хід виконання завдання:



Вивід:



Текст програми:

|  |
| --- |
| students = [  {"name": "Irisha", "grade": 100},  {"name": "Liza", "grade": 45},  {"name": "Andrey", "grade": 60},  {"name": "Kate", "grade": 84}  ]  sorted\_name = sorted(students, key=lambda student: student["name"])  print("Sorted name:", sorted\_name)  sorted\_grade = sorted(students, key=lambda student: student["grade"])  print("Sorted grade:", sorted\_grade) |

Посилання на github:

<https://github.com/Lizarebenokk>

# **Звіт до Теми №7**

Об'єктно-орієнтоване програмування

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**](https://docs.python.org/3/tutorial/classes.html)

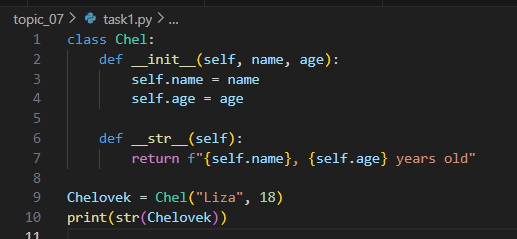
**2) Ознайомитись з існуючими за замовченням методами класу по типу \_\_init\_\_(self) \_\_str\_\_(self)\_\_ та надати приклади використання.**

**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.**

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

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

Хід виконання завдання:



Вивід:

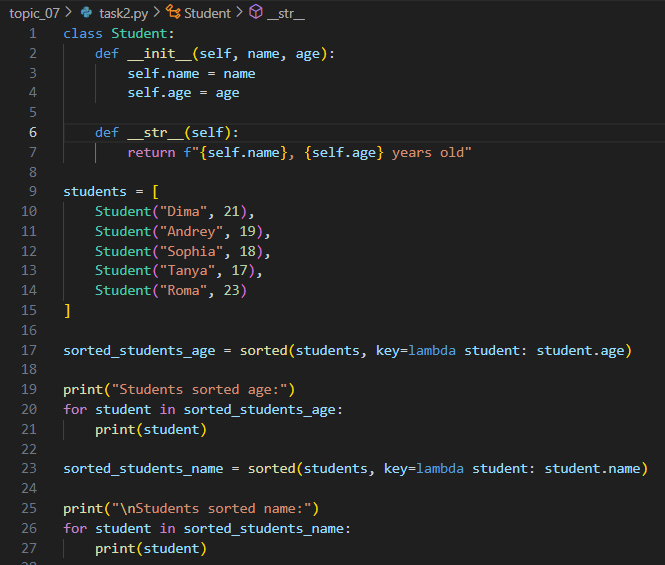


Текст програми:

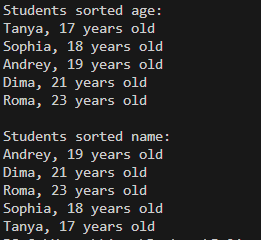
|  |
| --- |
| class Chel:  def \_\_init\_\_(self, name, age):  self.name = name  self.age = age  def \_\_str\_\_(self):  return f"{self.name}, {self.age} years old"  Chelovek = Chel("Liza", 18)  print(str(Chelovek)) |

**Розробити клас Student атрибутами якого э два параметра name та age**

Хід виконання завдання:



Вивід:

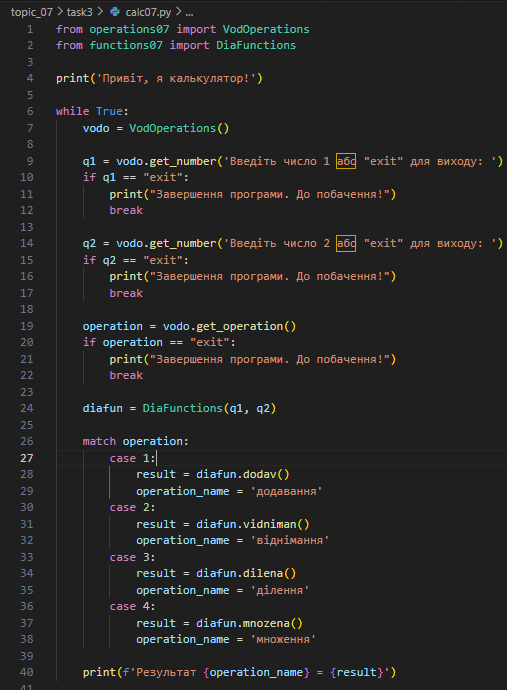


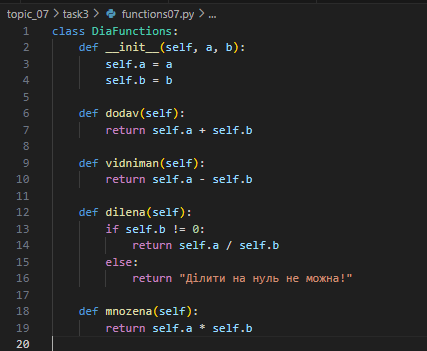
Текст програми:

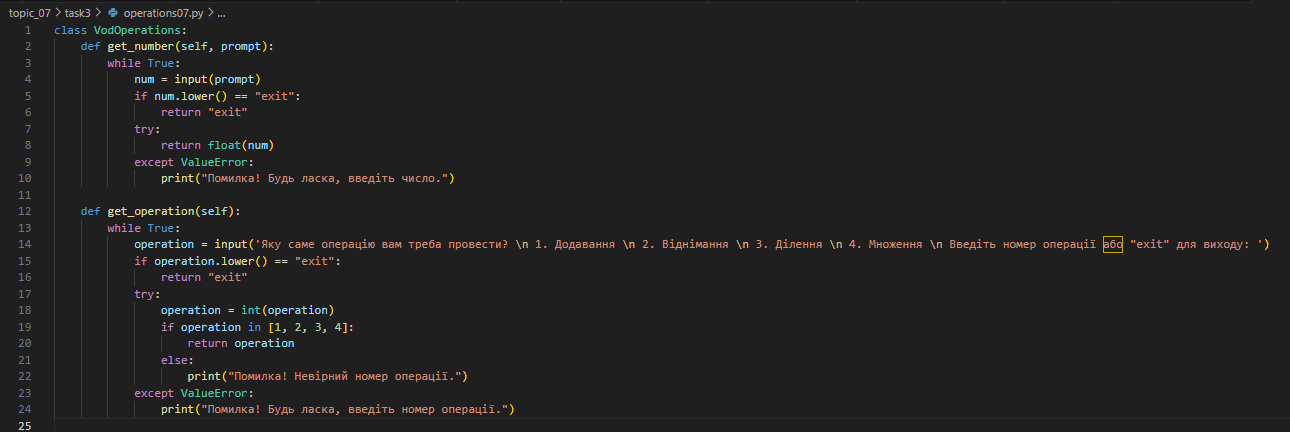
|  |
| --- |
| class Student:  def \_\_init\_\_(self, name, age):  self.name = name  self.age = age  def \_\_str\_\_(self):  return f"{self.name}, {self.age} years old"  students = [  Student("Dima", 21),  Student("Andrey", 19),  Student("Sophia", 18),  Student("Tanya", 17),  Student("Roma", 23)  ]  sorted\_students\_age = sorted(students, key=lambda student: student.age)  print("Students sorted age:")  for student in sorted\_students\_age:  print(student)  sorted\_students\_name = sorted(students, key=lambda student: student.name)  print("\nStudents sorted name:")  for student in sorted\_students\_name:  print(student) |

**Використовуючи принципи ООП переписати програму Калькулятор. Використати модульний підхід.**

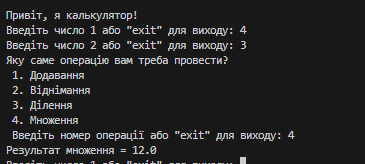
Хід виконання завдання:







Вивід:



Текст програми calc07.py:

|  |
| --- |
| from operations07 import VodOperations  from functions07 import DiaFunctions  print('Привіт, я калькулятор!')  while True:  vodo = VodOperations()    q1 = vodo.get\_number('Введіть число 1 або "exit" для виходу: ')  if q1 == "exit":  print("Завершення програми. До побачення!")  break  q2 = vodo.get\_number('Введіть число 2 або "exit" для виходу: ')  if q2 == "exit":  print("Завершення програми. До побачення!")  break  operation = vodo.get\_operation()  if operation == "exit":  print("Завершення програми. До побачення!")  break  diafun = DiaFunctions(q1, q2)  match operation:  case 1:  result = diafun.dodav()  operation\_name = 'додавання'  case 2:  result = diafun.vidniman()  operation\_name = 'віднімання'  case 3:  result = diafun.dilena()  operation\_name = 'ділення'  case 4:  result = diafun.mnozena()  operation\_name = 'множення'  print(f'Результат {operation\_name} = {result}') |

Текст програми functions07.py:

|  |
| --- |
| class DiaFunctions:  def \_\_init\_\_(self, a, b):  self.a = a  self.b = b  def dodav(self):  return self.a + self.b  def vidniman(self):  return self.a - self.b  def dilena(self):  if self.b != 0:  return self.a / self.b  else:  return "Ділити на нуль не можна!"  def mnozena(self):  return self.a \* self.b |

Текст програми operations.py:

|  |
| --- |
| class VodOperations:  def get\_number(self, prompt):  while True:  num = input(prompt)  if num.lower() == "exit":  return "exit"  try:  return float(num)  except ValueError:  print("Помилка! Будь ласка, введіть число.")  def get\_operation(self):  while True:  operation = input('Яку саме операцію вам треба провести? \n 1. Додавання \n 2. Віднімання \n 3. Ділення \n 4. Множення \n Введіть номер операції або "exit" для виходу: ')  if operation.lower() == "exit":  return "exit"  try:  operation = int(operation)  if operation in [1, 2, 3, 4]:  return operation  else:  print("Помилка! Невірний номер операції.")  except ValueError:  print("Помилка! Будь ласка, введіть номер операції.") |

Посилання на github:

<https://github.com/Lizarebenokk>