**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 "Daniilkoz" перетворити наступним чином "zokliinaD", вважаючи сталою довжину рядку в 9 символів.

Хід виконання завдання:

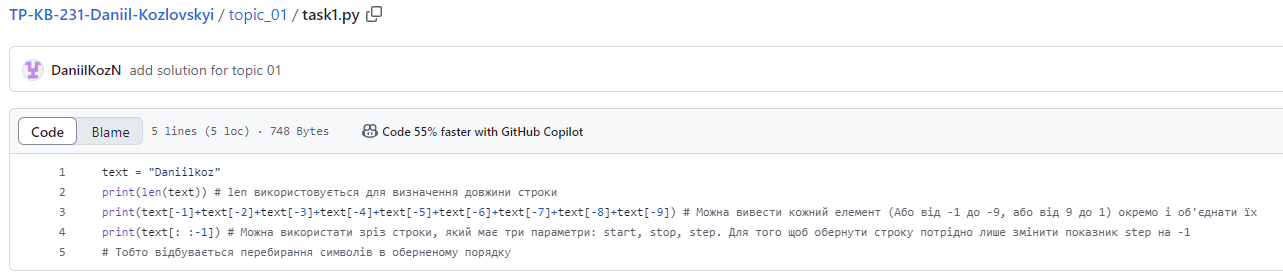
Існує 2 методи для обернення рядку у зворотному напрямку: Можна виводити кожний елемент окремо і об’єднати їх. Або можна використовувати зріз строки, вказавши в 3 параметрі функції (step) значення -1, тобто перебирання символів у оберненому напрямку.

Текст програми:

|  |
| --- |
| text = "Daniilkoz"  print(len(text))  print(text[-1]+text[-2]+text[-3]+text[-4]+text[-5]+text[-6]+text[-7]+text[-8]+text[-9])  print(text[: :-1]) |

Посилання на github: <https://github.com/DaniilKozN/TP-KB-231-Daniil-Kozlovskyi/blob/main/topic_01/task1.py>

Знімок екрану з посилання на github:



**Тестування функцій**

Необхідно протестувати функції, що працюють з рядками, а саме: strip(), capitalize(), title(), upper(), lower().

Хід виконання завдання:

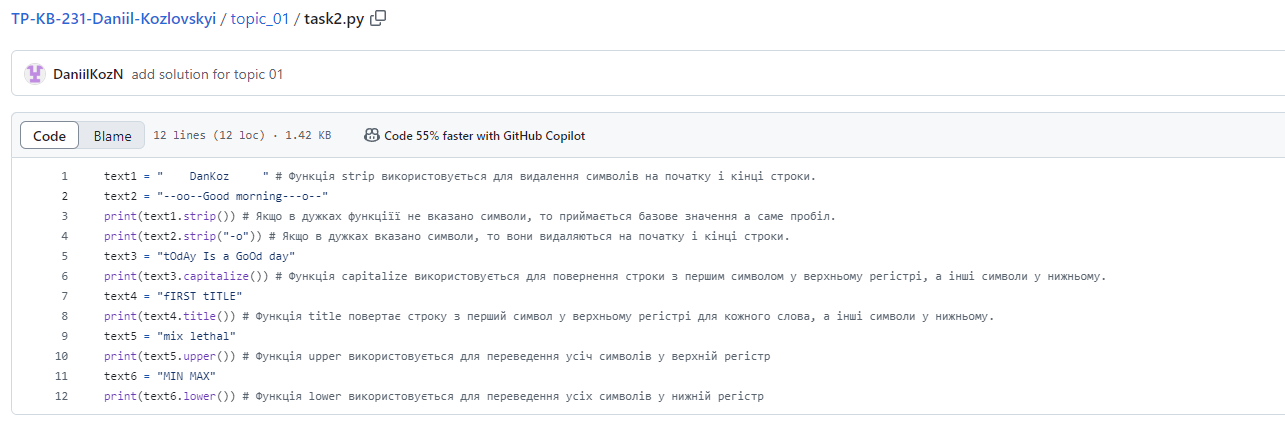
Для кожної функції створюю окремий рядок, який найкраще проілюструє роботу функції. Функція strip, використовують для видалення символів на початку і кінці строки, якщо в дужках вказати символи то видаляться будуть вони, тому створюю два рядка: Перший з великою кількістю пробілів, а інший з символами «o-». Наступні функції: capitalize і title виконують схожі функції: Перша повертаю строку лише з першим символом у верхньому регістрі, а друга повертає строку з першим символом у верхньому регістрі у кожному слові. Наступні дві функції є протилежними одна одній. Функція upper повертає усі символи у верхньому регістрі, а lower повертає у нижньому.

Текст програми:

|  |
| --- |
| text1 = " DanKoz "  text2 = "--oo--Good morning---o--"  print(text1.strip())  print(text2.strip("-o"))  text3 = "tOdAy Is a GoOd day"  print(text3.capitalize())  text4 = "fIRST tITLE"  print(text4.title())  text5 = "mix lethal"  print(text5.upper())  text6 = "MIN MAX"  print(text6.lower()) |

Посилання на github: <https://github.com/DaniilKozN/TP-KB-231-Daniil-Kozlovskyi/blob/main/topic_01/task2.py>

Знімок екрану з посилання на github:



**Функція пошуку дискримінанта**

Необхідно написати функцію пошуку дискримінанту квадратного рівняння.

Хід виконання завдання:

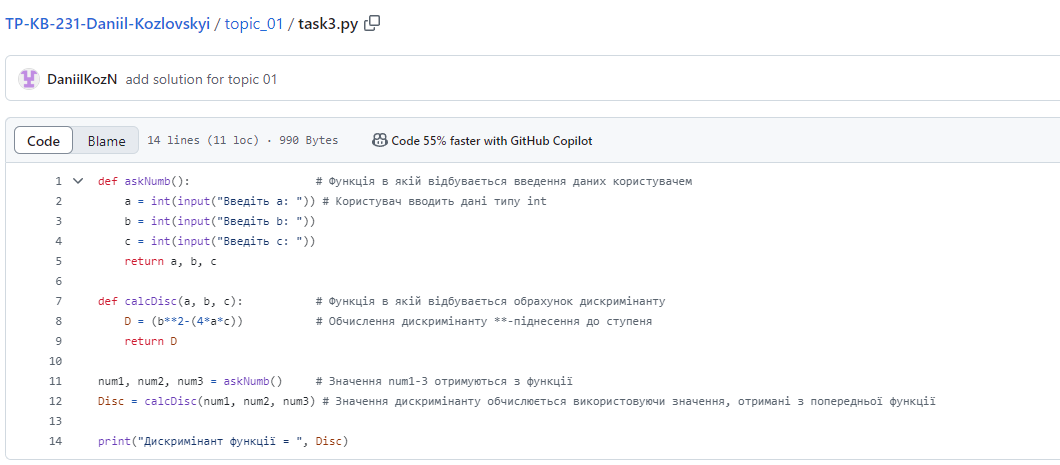
Для реалізації завдання було написано дві функції власного виробництва: Перша необхідна для вводу даних ( a,b,c ), а друга обраховує дискримінант. Після функцій вказано змінні, які беруть дані з функцій. Після всіх обрахунків виводиться результат.

Текст програми:

|  |
| --- |
| def askNumb():      a = int(input("Введіть a: "))      b = int(input("Введіть b: "))      c = int(input("Введіть c: "))      return a, b, c  def calcDisc(a, b, c):      D = (b\*\*2-(4\*a\*c))      return D    num1, num2, num3 = askNumb()  Disc = calcDisc(num1, num2, num3)  print("Дискримінант функції = ", Disc) |

Посилання на github: <https://github.com/DaniilKozN/TP-KB-231-Daniil-Kozlovskyi/blob/main/topic_01/task3.py>

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:

**Функція пошуку коренів квадратного рівняння**

Необхідно написати функцію пошуку коренів квадратного рівняння використовуючи функцію розрахунку дискримінанту.

Хід виконання завдання:

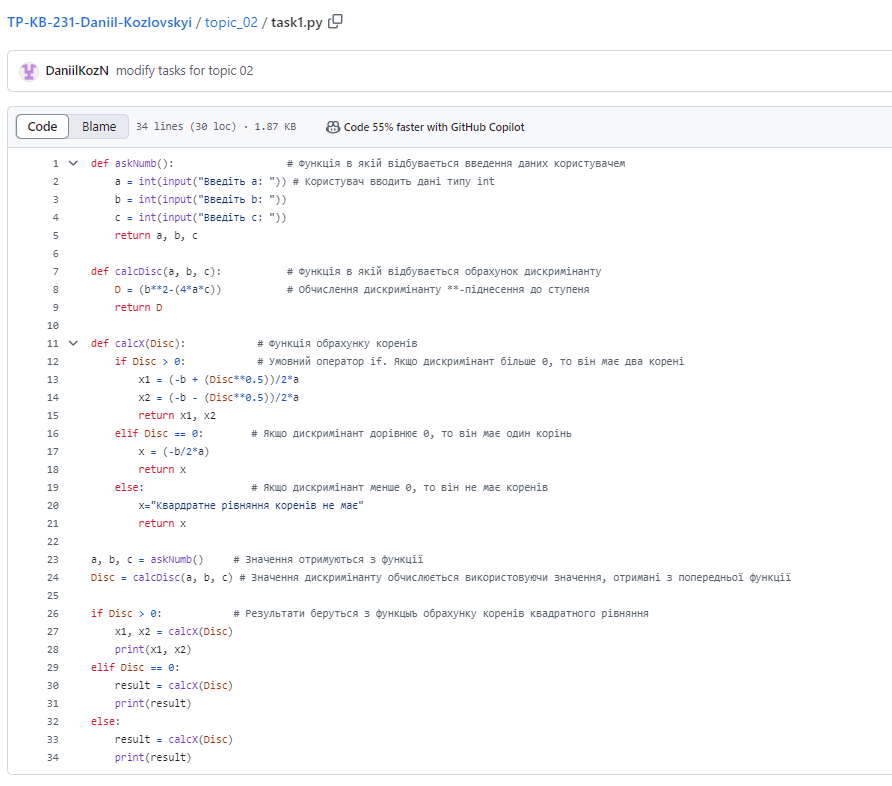
Для реалізації завдання було використано код, який був написаний у попередній темі, проте він закінчується на обрахунках дискримінанту. Тому наступним кроком буде написання коду у якому відбувається обрахунок коренів. Але кількість коренів залежить від значення дискримінанту. Тому використовую умовні переходи, щоб коли дискримінант більше 0, то відбувається обрахунок 2 коренів, а якщо дискримінант дорівнює 0, то обраховується один корінь, в інших випадках коренів рівняння не має.

Текст програми:

|  |
| --- |
| def askNumb():  a = int(input("Введіть a: "))  b = int(input("Введіть b: "))  c = int(input("Введіть c: "))  return a, b, c  def calcDisc(a, b, c):  D = (b\*\*2-(4\*a\*c))  return D  def calcX(Disc):  if Disc > 0:  x1 = (-b + (Disc\*\*0.5))/2\*a  x2 = (-b - (Disc\*\*0.5))/2\*a  return x1, x2  elif Disc == 0:  x = (-b/2\*a)  return x  else:  x="Квардратне рівняння коренів не має"  return x    a, b, c = askNumb()  Disc = calcDisc(a, b, c)  if Disc > 0:  x1, x2 = calcX(Disc)  print(x1, x2)  elif Disc == 0:  result = calcX(Disc)  print(result)  else:  result = calcX(Disc)  print(result) |

Посилання на github: <https://github.com/DaniilKozN/TP-KB-231-Daniil-Kozlovskyi/blob/main/topic_02/task1.py>

Знімок екрану з посилання на github:



**Програма калькулятор через if else**

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

Хід виконання завдання:

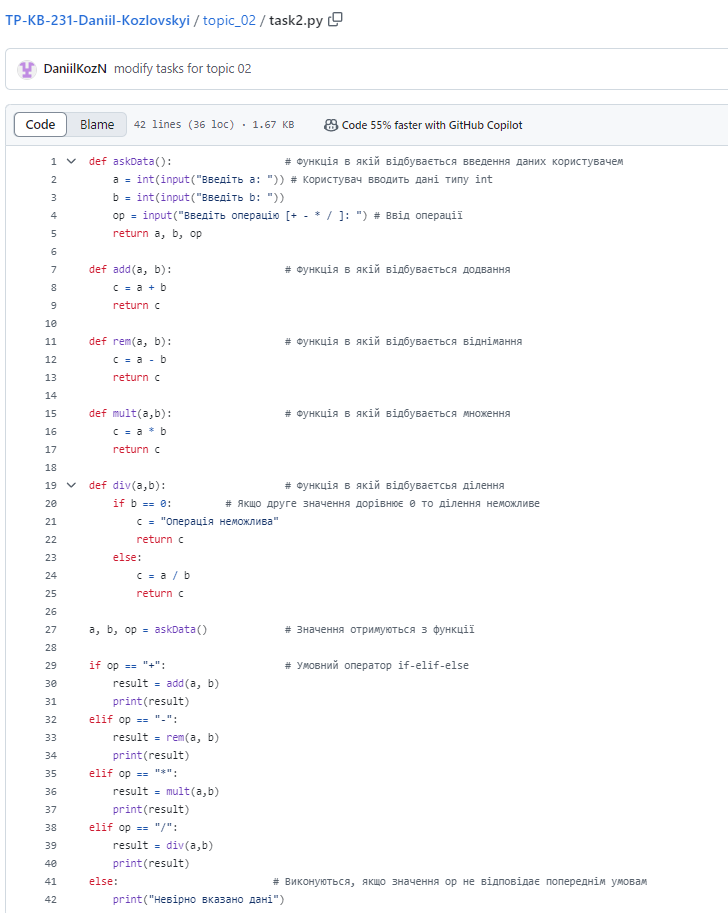
Для реалізації завдання необхідно було створити декілька функцій: Функція для введення даних (перше число, друге число і операція), окремі функції для кожної операції (додавання, віднімання, множення, ділення). Також потрібно враховувати, що ділити на 0 неможна тому в функції ділення потрібно додати умовні переходи, щоб при умові, якщо друге число дорівнює 0, то операція не виконується, а користувачу повідомляють про її неможливість. Після цього потрібно використовуючи if-elif-else конструкцію розписати усі можливі варіанти змінної «op», яка відповідає за операції. Також потрібно розписати умову, якщо користувач ввів інше значення в змінну.

Текст програми:

|  |
| --- |
| def askData():  a = int(input("Введіть a: "))  b = int(input("Введіть b: "))  op = input("Введіть операцію [+ - \* / ]: ")  return a, b, op  def add(a, b):  c = a + b  return c  def rem(a, b):  c = a - b  return c  def mult(a,b):  c = a \* b  return c  def div(a,b):  if b == 0:  c = "Операція неможлива"  return c  else:  c = a / b  return c  a, b, op = askData()  if op == "+":  result = add(a, b)  print(result)  elif op == "-":  result = rem(a, b)  print(result)  elif op == "\*":  result = mult(a,b)  print(result)  elif op == "/":  result = div(a,b)  print(result)  else:  print("Невірно вказано дані") |

Посилання на github: <https://github.com/DaniilKozN/TP-KB-231-Daniil-Kozlovskyi/blob/main/topic_02/task2.py>

Знімок екрану з посилання на github:



**Програма калькулятор через match**

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

Хід виконання завдання:

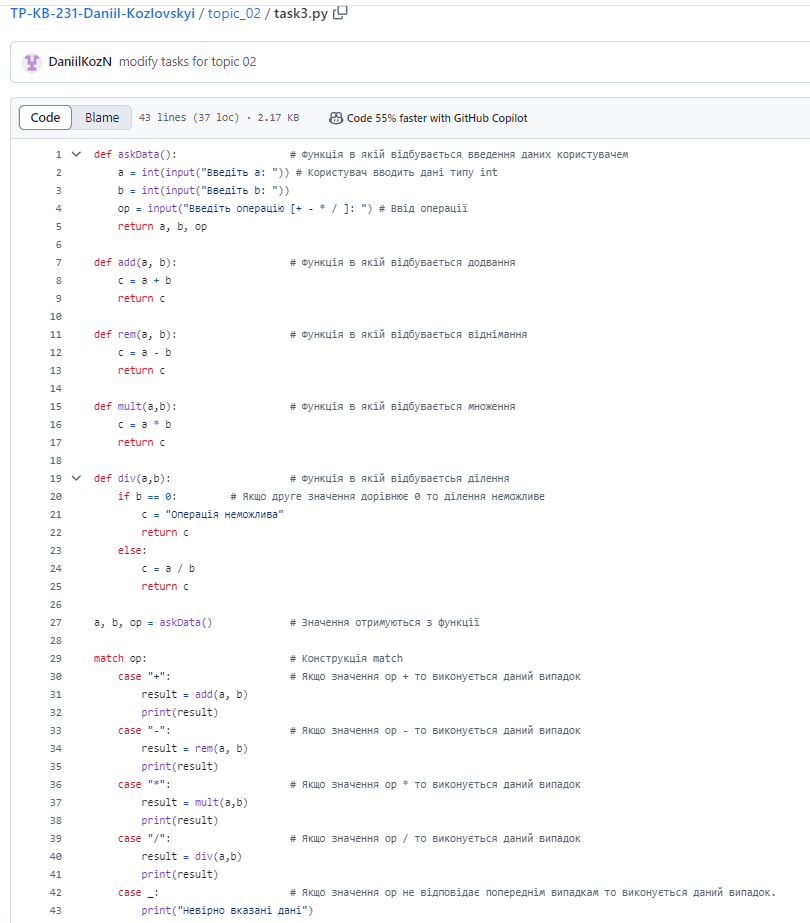
Для реалізації завдання можна використати код з попереднього, так як основною зміню є використання match конструкцій замість if else. Отже копіюю функцію для введення даних і функції операцій. В цьому завданні використовую match конструкцію для того щоб розписати усі випадки, які може вказати користувач в змінній «op». Також додаю випадок \_, який відповідає за будь-який інший користувацький ввід, якщо значення не відповідає попереднім випадкам ( + - \* / ).

Текст програми:

|  |
| --- |
| def askData():  a = int(input("Введіть a: "))  b = int(input("Введіть b: "))  op = input("Введіть операцію [+ - \* / ]: ")  return a, b, op  def add(a, b):  c = a + b  return c  def rem(a, b):  c = a - b  return c  def mult(a,b):  c = a \* b  return c  def div(a,b):  if b == 0:  c = "Операція неможлива"  return c  else:  c = a / b  return c  a, b, op = askData()  match op:  case "+":  result = add(a, b)  print(result)  case "-":  result = rem(a, b)  print(result)  case "\*":  result = mult(a,b)  print(result)  case "/":  result = div(a,b)  print(result)  case \_:  print("Невірно вказані дані") |

Посилання на github: <https://github.com/DaniilKozN/TP-KB-231-Daniil-Kozlovskyi/blob/main/topic_02/task3.py>

Знімок екрану з посилання на github:



Звіт до Теми №3

Цикли

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

**Програма 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:

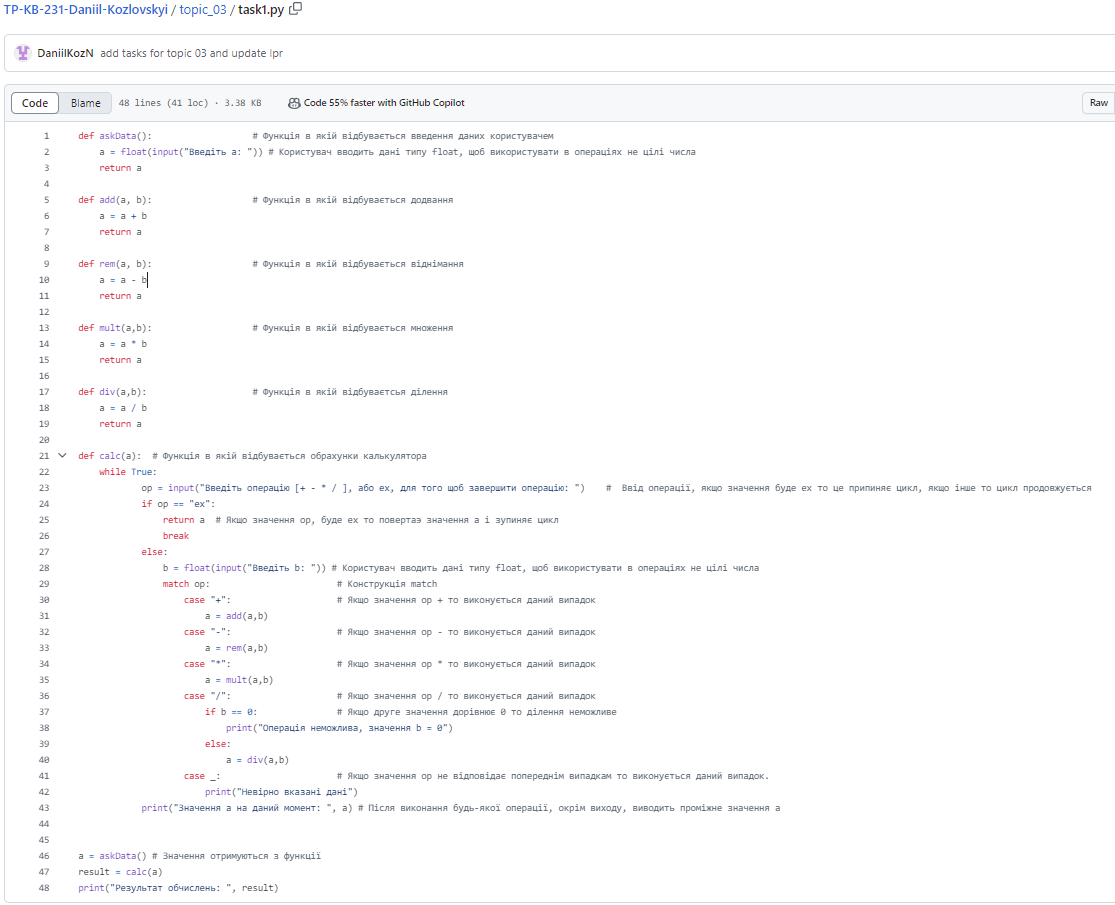
За основу було взято програму калькулятор з 3 завдання попередньої теми. Функції вводу та операцій не змінюю, окрім ділення, де позбуваюсь умови перевірки, так як перевірка буде відбуватися у функції калькулятор. Створюю функцію калькулятор, спочатку створюю нескінчений цикл, який зупиниться тільки при одній умові, якщо користувач впишу у значення змінної «op» - «ex», яке є умовою завершення циклу. Наступним строюю зміну «op», яку вводить користувач. Наступним створюю умовний перехід, який перевіряє значення «op». При умові, що «op» != «ex», то запитується друге число, після чого використовується умовний перехід, який використовувався у попередній темі, окрім ділення, де перевірка відбувається саме в цьому переході, а не в функції ділення. Після завершення обрахунків програмо повідомляє значення a, і продовжує цикл, доки користувач його не завершить.

Текст програми:

|  |
| --- |
| def askData():      a = float(input("Введіть a: "))      return a  def add(a, b):      a = a + b      return a  def rem(a, b):      a = a - b      return a  def mult(a,b):      a = a \* b      return a  def div(a,b):      a = a / b      return a  def calc(a):  #      while True:              op = input("Введіть операцію [+ - \* / ], або ex, для того щоб завершити операцію: ")              if op == "ex":                  return a                  break              else:                  b = float(input("Введіть b: "))                  match op:                      case "+":                          a = add(a,b)                      case "-":                          a = rem(a,b)                      case "\*":                          a = mult(a,b)                      case "/":                          if b == 0:                             print("Операція неможлива, значення b = 0")                          else:                              a = div(a,b)                      case \_:                          print("Невірно вказані дані")              print("Значення a на даний момент: ", a)    a = askData()  result = calc(a)  print("Результат обчислень: ", result) |

Посилання на github: <https://github.com/DaniilKozN/TP-KB-231-Daniil-Kozlovskyi/blob/main/topic_03/task1.py>

Знімок екрану з посилання на github:



**Програма тестування функції списків**

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

Хід виконання завдання:

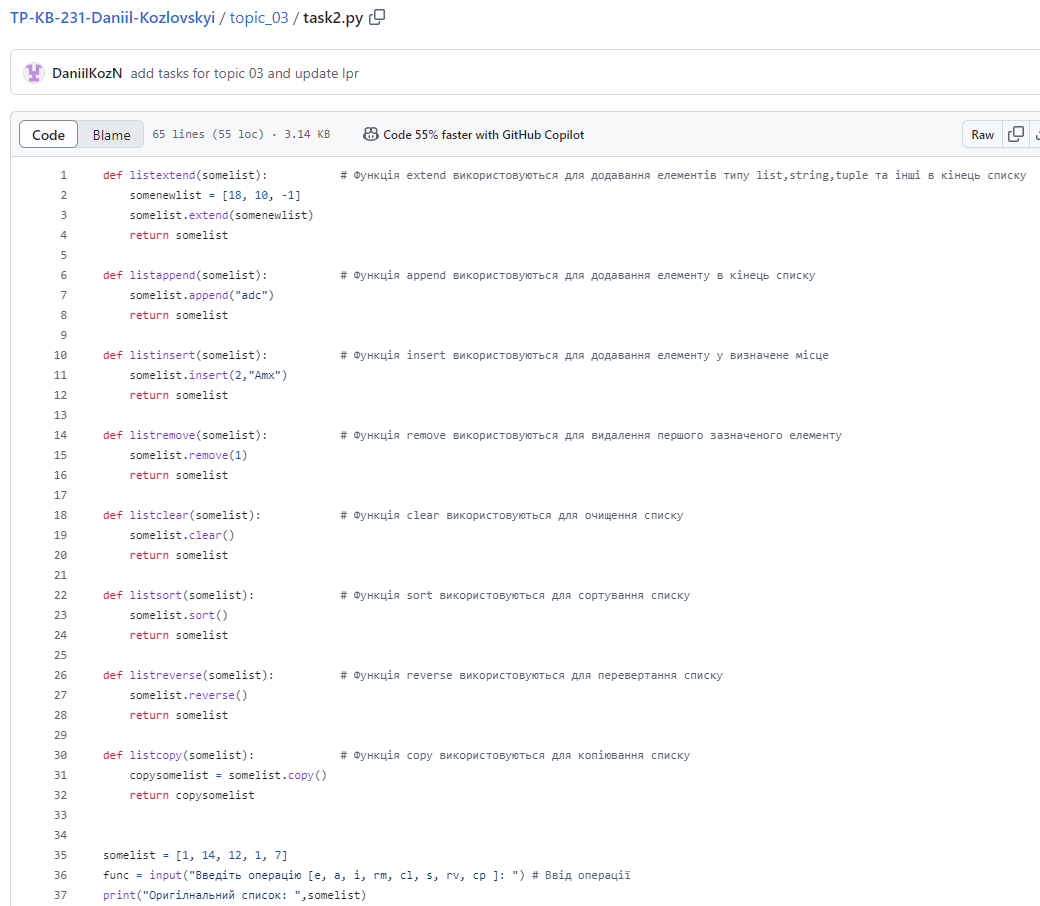
Для виконання даного завдання створив окремі функції для кожної операції, створив список з яким будуть операції і змінну, для визначення операції, також використовую умовний перехід. Функція extend використовуються для додавання елементів типу list,string,tuple та інші в кінець списку, функція append використовуються для додавання елементу в кінець списку, функція insert використовуються для додавання елементу у визначене місце, функція remove використовуються для видалення першого зазначеного елементу, функція clear використовуються для очищення списку, функція sort використовуються для cортування списку, функція reverse використовуються для перевертання списку, функція copy використовуються для копіювання списку.

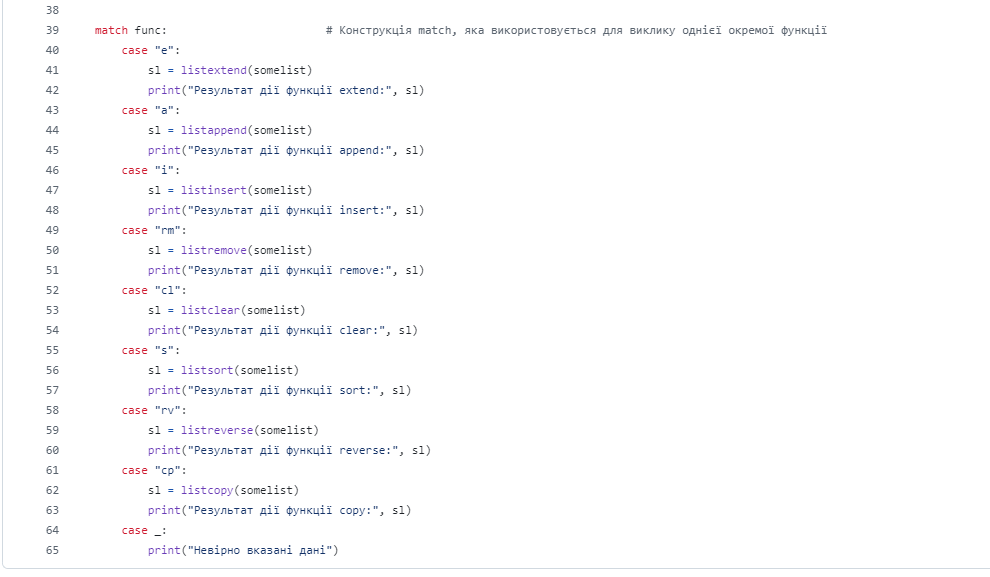
Текст програми:

|  |
| --- |
| def listextend(somelist):      somenewlist = [18, 10, -1]      somelist.extend(somenewlist)      return somelist  def listappend(somelist):      somelist.append("adc")      return somelist  def listinsert(somelist):      somelist.insert(2,"Amx")      return somelist  def listremove(somelist):      somelist.remove(1)      return somelist  def listclear(somelist):      somelist.clear()      return somelist  def listsort(somelist):      somelist.sort()      return somelist  def listreverse(somelist):      somelist.reverse()      return somelist  def listcopy(somelist):      copysomelist = somelist.copy()      return copysomelist  somelist = [1, 14, 12, 1, 7]  func = input("Введіть операцію [e, a, i, rm, cl, s, rv, cp ]: ") # Ввід операції  print("Оригілнальний список: ",somelist)  match func:      case "e":          sl = listextend(somelist)          print("Результат дії функції extend:", sl)      case "a":          sl = listappend(somelist)          print("Результат дії функції append:", sl)      case "i":          sl = listinsert(somelist)          print("Результат дії функції insert:", sl)      case "rm":          sl = listremove(somelist)          print("Результат дії функції remove:", sl)      case "cl":          sl = listclear(somelist)          print("Результат дії функції clear:", sl)      case "s":          sl = listsort(somelist)          print("Результат дії функції sort:", sl)      case "rv":          sl = listreverse(somelist)          print("Результат дії функції reverse:", sl)      case "cp":          sl = listcopy(somelist)          print("Результат дії функції copy:", sl)      case \_:          print("Невірно вказані дані") |

Посилання на github: <https://github.com/DaniilKozN/TP-KB-231-Daniil-Kozlovskyi/blob/main/topic_03/task2.py>

Знімок екрану з посилання на github:





**Програма тестування функції словників**

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

Хід виконання завдання:

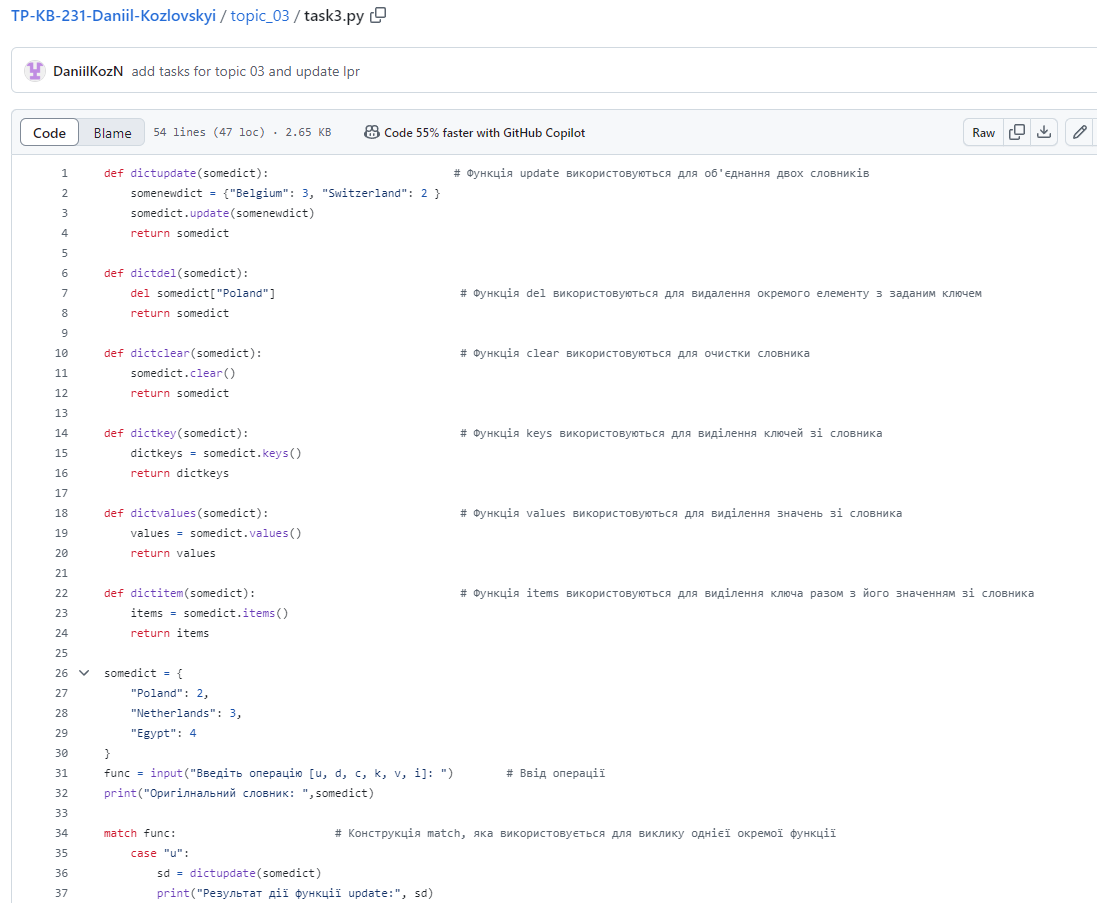
Повторюю стиль коду, як в попередньому завданні. Функція update використовуються для об'єднання двох словників, функція del використовуються для видалення окремого елементу з заданим ключем, функція clear використовуються для очистки словника, функція keys використовуються для виділення ключей зі словника, функція values використовуються для виділення значень зі словника, функція items використовуються для виділення ключа разом з його значенням зі словника.

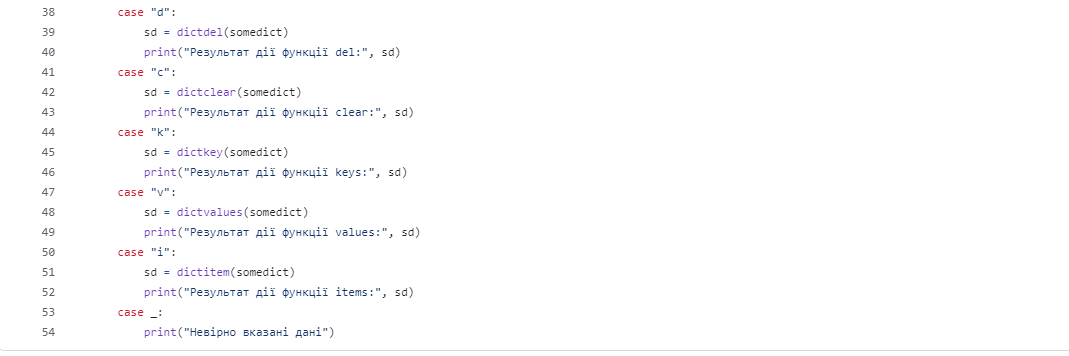
Текст програми:

|  |
| --- |
| def dictupdate(somedict):      somenewdict = {"Belgium": 3, "Switzerland": 2 }      somedict.update(somenewdict)      return somedict  def dictdel(somedict):      del somedict["Poland"]      return somedict  def dictclear(somedict):      somedict.clear()      return somedict  def dictkey(somedict):      dictkeys = somedict.keys()      return dictkeys  def dictvalues(somedict):      values = somedict.values()      return values  def dictitem(somedict):      items = somedict.items()      return items  somedict = {      "Poland": 2,      "Netherlands": 3,      "Egypt": 4  }  func = input("Введіть операцію [u, d, c, k, v, i]: ")  print("Оригілнальний словник: ",somedict)  match func:      case "u":          sd = dictupdate(somedict)          print("Результат дії функції update:", sd)      case "d":          sd = dictdel(somedict)          print("Результат дії функції del:", sd)      case "c":          sd = dictclear(somedict)          print("Результат дії функції clear:", sd)      case "k":          sd = dictkey(somedict)          print("Результат дії функції keys:", sd)      case "v":          sd = dictvalues(somedict)          print("Результат дії функції values:", sd)      case "i":          sd = dictitem(somedict)          print("Результат дії функції items:", sd)      case \_:          print("Невірно вказані дані") |

Посилання на github: <https://github.com/DaniilKozN/TP-KB-231-Daniil-Kozlovskyi/blob/main/topic_03/task3.py>

Знімок екрану з посилання на github:





**Програма пошуку позиції для вставки**

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

Хід виконання завдання:

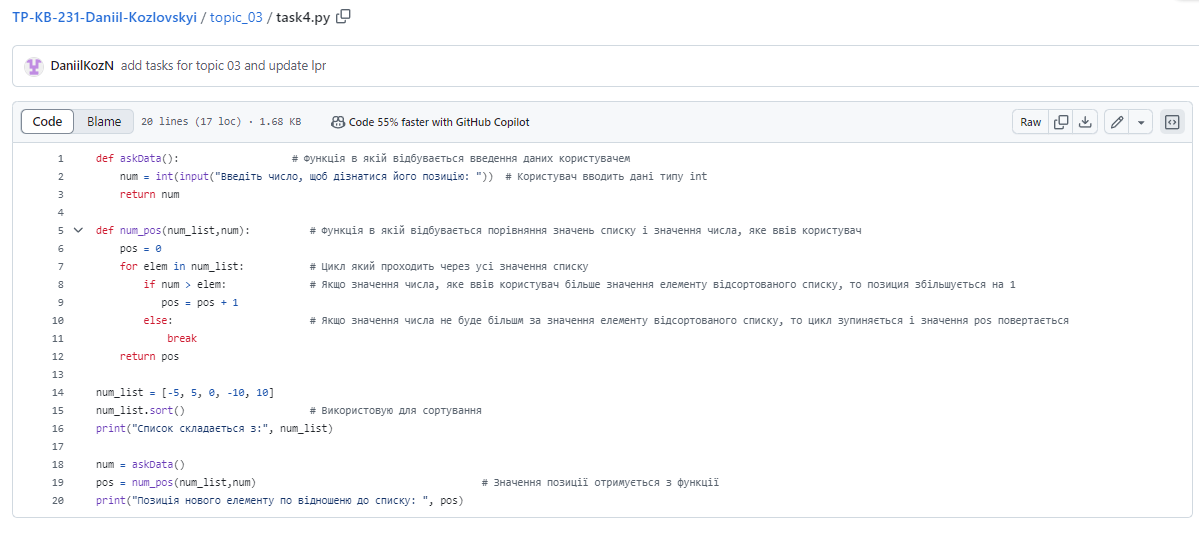
Для виконання завдання побудую функцію, яка порівнює значення, яке ввів користувач із кожним значенням елементу списку, якщо значення списку менше, то позиція збільшується, якщо ж менше або дорівнює, то цикл зупиняється. Цикл працює поки список має значення. Для порівняння створюю список, який відсортовую, використовуючи sort(). Потім створюю функцію вводу, в якій отримуємо значення, яке ввів користувач.

Текст програми:

|  |
| --- |
| def askData():      num = int(input("Введіть число, щоб дізнатися його позицію: "))      return num  def num\_pos(num\_list,num):      pos = 0      for elem in num\_list          if num > elem:             pos = pos + 1          else:              break      return pos    num\_list = [-5, 5, 0, -10, 10]  num\_list.sort()  print("Список складається з:", num\_list)  num = askData()  pos = num\_pos(num\_list,num)  print("Позиція нового елементу по відношеню до списку: ", pos) |

Посилання на github: <https://github.com/DaniilKozN/TP-KB-231-Daniil-Kozlovskyi/blob/main/topic_03/task4.py>

Знімок екрану з посилання на github:



Звіт до Теми №4

Виняткові ситуації

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

**Програма калькулятор з обробкою випадкових ситуацій**

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

Хід виконання завдання:

За основу візьму програму, яку написав в першому задачі попередньої теми. В програмі є три місця, де можливо викликати виняткову ситуацію:

1. Функція вводу значення a,
2. Ввід значення b,
3. Функція ділення.

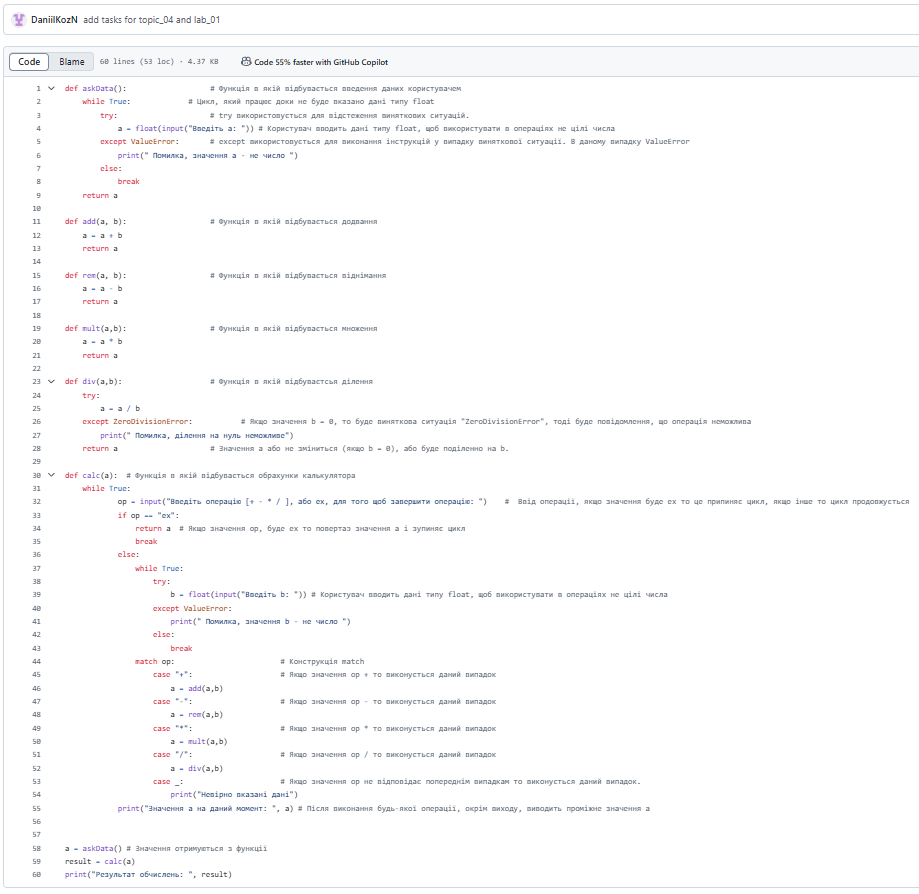
Для першого і другого випадку присутня виняткова ситуація типу «ValueError», отже використовую конструкцію «try except else». Якщо значення a вказано вірно, то нічого не відбувається і програма продовжує роботу без змін, якщо значення вказано не вірно, то виводиться повідомлення про помилку. Вкладаю дану операцію у нескінченний цикл, який зупиняється тільки коли дані вказано вірно. Наступним розгляну 3 випадок, якщо значення b = 0, то відбудиться виняткова подія «ZeroDivisionError», так як ділити на нуль неможливо. Знову використовую конструкцію «try except», але не вкладаю операцію в цикл, так як немає необхідності в декількох спробах ділення (Значення b не змінюється).

Текст програми:

|  |
| --- |
| def askData():      while True:          try:              a = float(input("Введіть a: "))          except ValueError:              print(" Помилка, значення a - не число ")          else:              break      return a  def add(a, b):      a = a + b      return a  def rem(a, b):      a = a - b      return a  def mult(a,b):      a = a \* b      return a  def div(a,b):      try:          a = a / b      except ZeroDivisionError:          print(" Помилка, ділення на нуль неможливе")      return a  def calc(a):      while True:              op = input("Введіть операцію [+ - \* / ], або ex, для того щоб завершити операцію: ")              if op == "ex":                  return a                  break              else:                  while True:                      try:                          b = float(input("Введіть b: "))                      except ValueError:                          print(" Помилка, значення b - не число ")                      else:                          break                  match op:                      case "+":                          a = add(a,b)                      case "-":                          a = rem(a,b)                      case "\*":                          a = mult(a,b)                      case                          a = div(a,b)                      case \_:                          print("Невірно вказані дані")              print("Значення a на даний момент: ", a)    a = askData()  result = calc(a)  print("Результат обчислень: ", result) |

Посилання на github: <https://github.com/DaniilKozN/TP-KB-231-Daniil-Kozlovskyi/blob/main/topic_04/task1-2.py>

Знімок екрану з посилання на github:



Звіт до Теми №5

Бібліотеки

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

**Гра 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:

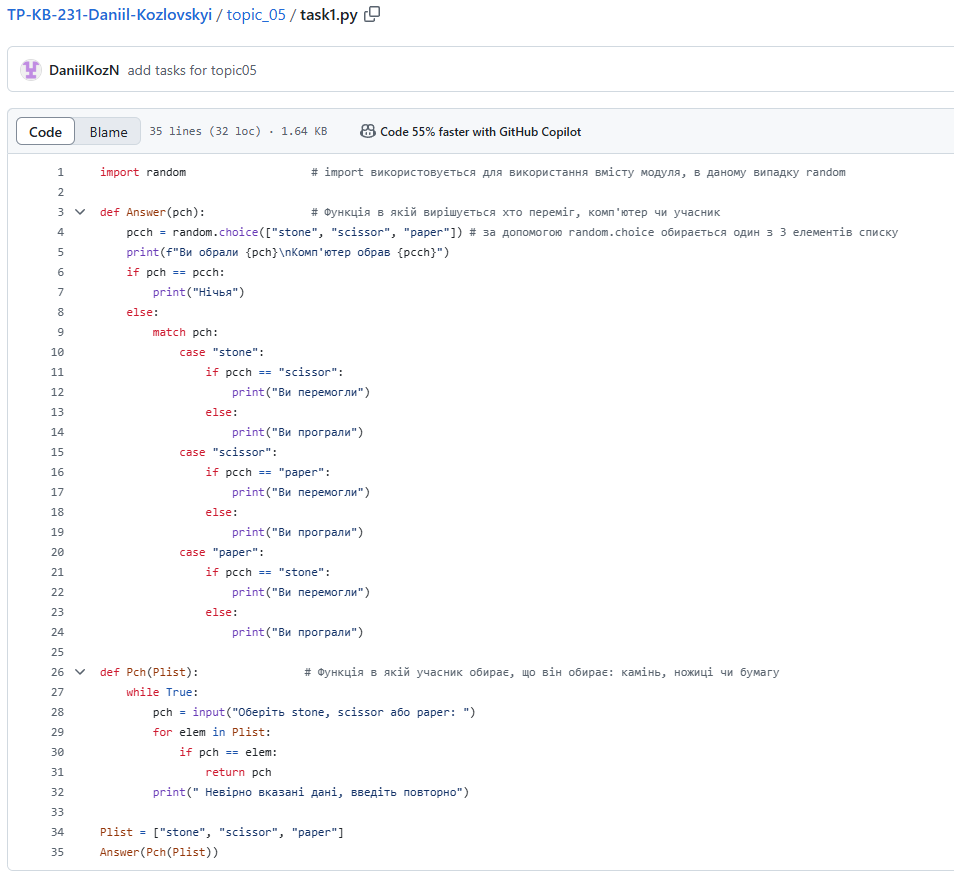
Почну з імпортування вмісту модулю random, для того щоб мати можливість генерувати вибір комп’ютера. Після цього напишу функцію в якій користувач обирає одне з 3 значень (камінь, ножиці, папір). В функції додаю перевірку, чи дорівнює ввід користувача одному з 3 значень. Наступною напишу функцію порівняння, в ній спочатку вибір генерує комп’ютер, а потім починається перевірка операцій, якщо обидва учасники обрали один елемент – нічия, якщо різні то залежно від вибору буде або перемога, або програш користувача.

Текст програми:

|  |
| --- |
| import random  def Answer(pch):      pcch = random.choice(["stone", "scissor", "paper"])      print(f"Ви обрали {pch}\nКомп'ютер обрав {pcch}")      if pch == pcch:          print("Нічья")      else:          match pch:              case "stone":                  if pcch == "scissor":                      print("Ви перемогли")                  else:                      print("Ви програли")              case "scissor":                  if pcch == "paper":                      print("Ви перемогли")                  else:                      print("Ви програли")              case "paper":                  if pcch == "stone":                      print("Ви перемогли")                  else:                      print("Ви програли")  def Pch(Plist):      while True:          pch = input("Оберіть stone, scissor або paper: ")          for elem in Plist:              if pch == elem:                  return pch          print(" Невірно вказані дані, введіть повторно")  Plist = ["stone", "scissor", "paper"]  Answer(Pch(Plist)) |

Посилання на github: <https://github.com/DaniilKozN/TP-KB-231-Daniil-Kozlovskyi/blob/main/topic_05/task1.py>

Знімок екрану з посилання на github:



**Програма конвертування іноземної валюти в українську гривню.**

Користувачу надається можливість введення кількості та типу валюти, результат роботи програми – конвертоване значення в українських гривнях або ре конвертоване значення в іншій валюті.

Хід виконання завдання:

Спочатку імпортую request, за допомогою якого можна отримати актуальний курс валют в гривнях у вигляді списку. Спочатку використаю функцію операцій, яку використовував попередньо, змінивши умови і операції (Показ, конвертація, вихід). Після цього реалізую функцію показу короткої назви валют (3 великі букви) з її ціною в гривнях, для цього використовую цикл, який пройде по всім елементам. Наступним реалізую функцію суми, яка знадобиться для визначення, яку кількість валюти потрібно конвертувати, за основу візьму функцію askData з попередньої теми.

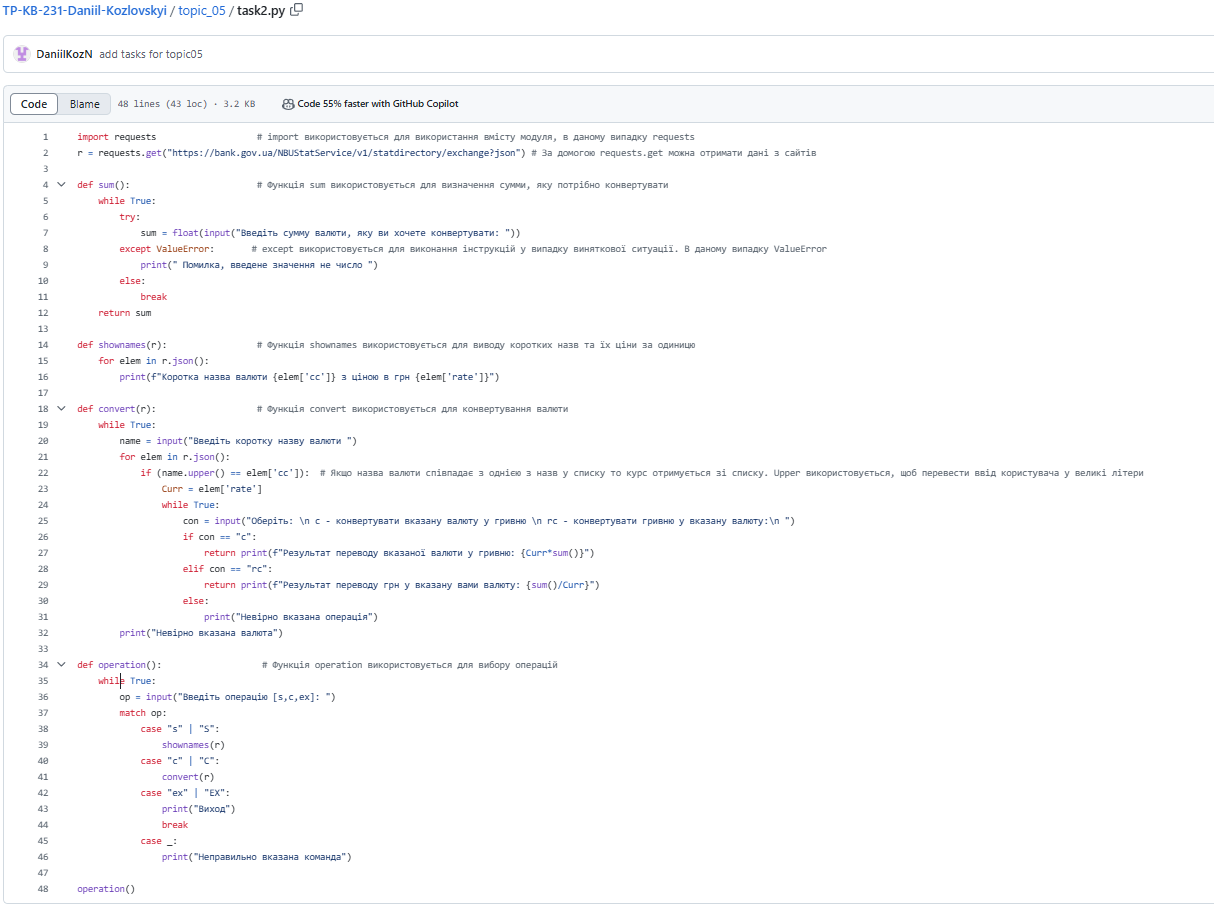
Останньою реалізую функцію конвертування, для цього спочатку дізнаємося у користувача, яку валюту він хоче конвертувати, після цього порівнюємо її з усіма короткими назвами у списку. Після знаходження валюти, зберігається її курс і запитується у користувача яку саме операцію він хоче виконати: конвертація вказаної валюти у гривні (Курс \* сума), чи конвертація гривні у вказану валюту (Сума / курс) , якщо операція буде вказано невірно то про це буде повідомлено користувача. Після цього користувача може або продовжити різні операції, або завершити роботу.

Текст програми:

|  |
| --- |
| import requests  r = requests.get("https://bank.gov.ua/NBUStatService/v1/statdirectory/exchange?json")  def sum():      while True:          try:              sum = float(input("Введіть сумму валюти, яку ви хочете конвертувати: "))          except ValueError:              print(" Помилка, введене значення не число ")          else:              break      return sum  def shownames(r):      for elem in r.json():          print(f"Коротка назва валюти {elem['cc']} з ціною в грн {elem['rate']}")  def convert(r):      while True:          name = input("Введіть коротку назву валюти ")          for elem in r.json():              if (name.upper() == elem['cc']):                  Curr = elem['rate']                  while True:                      con = input("Оберіть: \n c - конвертувати вказану валюту у гривню \n rc - конвертувати гривню у вказану валюту:\n ")                      if con == "c":                          return print(f"Результат переводу вказаної валюти у гривню: {Curr\*sum()}")                      elif con == "rc":                          return print(f"Результат переводу грн у вказану вами валюту: {sum()/Curr}")                      else:                          print("Невірно вказана операція")          print("Невірно вказана валюта")  def operation():      while True:          op = input("Введіть операцію [s,c,ex]: ")          match op:              case "s" | "S":                  shownames(r)              case "c" | "C":                  convert(r)              case "ex" | "EX":                  print("Виход")                  break              case \_:                  print("Неправильно вказана команда")  operation() |

Посилання на github: <https://github.com/DaniilKozN/TP-KB-231-Daniil-Kozlovskyi/blob/main/topic_05/task2.py>

Знімок екрану з посилання на github:



**Використання модулів для програми калькулятор.**

Розподілення програми калькулятор на модулі. Для функцій обчислення, операцій і самого калькулятору створюються окремі файлі.

Хід виконання завдання:

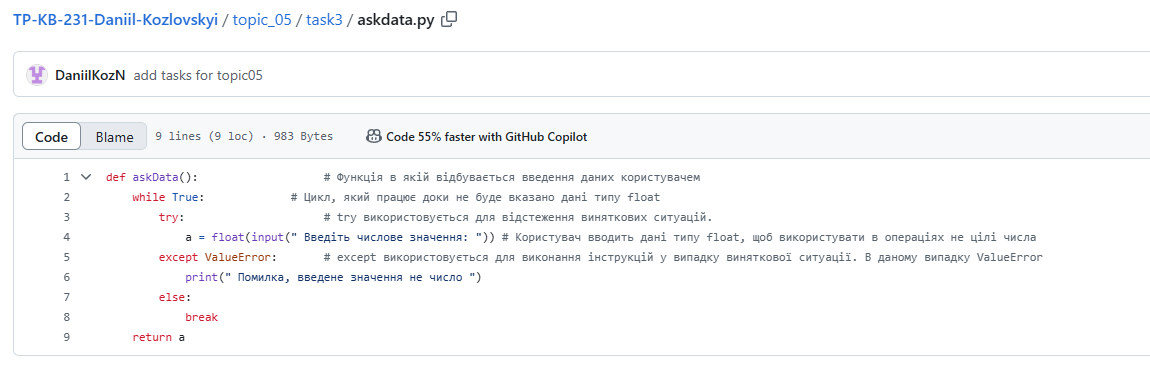
За основу візьму програму калькулятор з попередньої теми. Розділю його на модулі, спочатку перенесу функції обчислення (Додавання, віднімання, множення, ділення) в окремий файл func.py. А функцію askData в файл askData.py. Усі інші функцію перенесу в файл task3.py. В ньому спочатку імпортую функції з двох файлів через from import.

Текст програми:

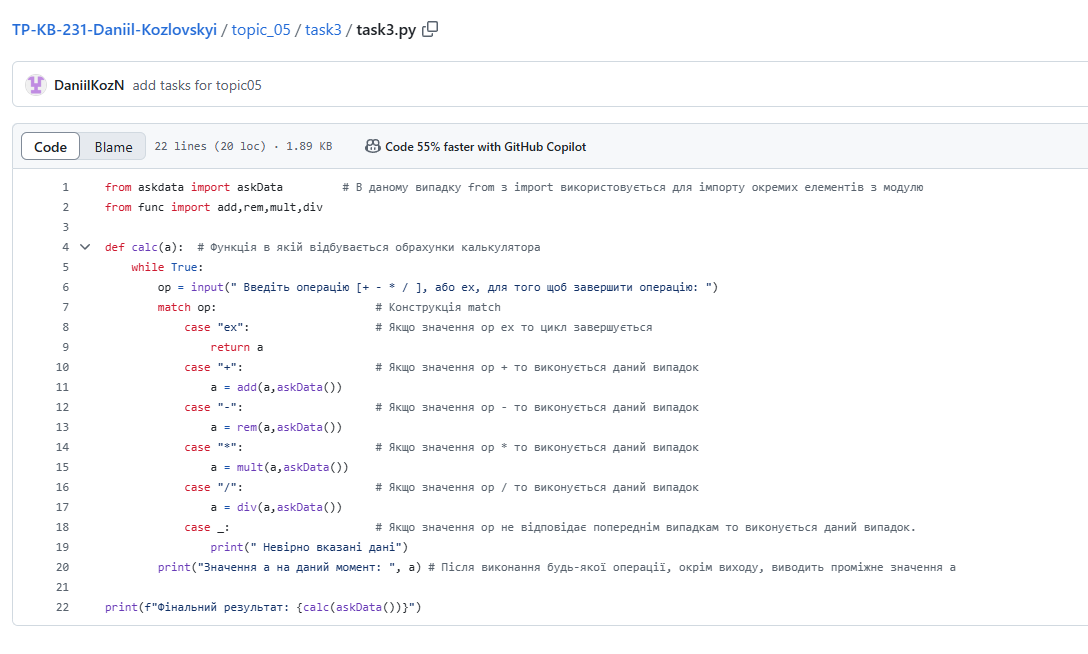
|  |
| --- |
| askData.py  def askData():      while True:          try:              a = float(input(" Введіть числове значення: "))          except ValueError:              print(" Помилка, введене значення не число ")          else:              break      return a  func.py  def add(a, b):      a = a + b      return a  def rem(a, b):      a = a - b      return a  def mult(a,b):      a = a \* b      return a  def div(a,b):      try:          a = a / b      except ZeroDivisionError:          print(" Помилка, ділення на нуль неможливе")      return a  task3.py  from askdata import askData  from func import add,rem,mult,div  def calc(a):  while True:  op = input(" Введіть операцію [+ - \* / ], або ex, для того щоб завершити операцію: ")  match op:  case "ex":  return a  case "+":  a = add(a,askData())  case "-":  a = rem(a,askData())  case "\*":  a = mult(a,askData())  case "/":  a = div(a,askData())  case \_:  print(" Невірно вказані дані")  print("Значення a на даний момент: ", a)    print(f"Фінальний результат: {calc(askData())}") |

Посилання на github: <https://github.com/DaniilKozN/TP-KB-231-Daniil-Kozlovskyi/tree/main/topic_05/task3>

Знімок екрану з посилання на github:







Звіт до Теми №6

Робота з файлами

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

**Використання lambda функцій для функції сортування.**

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:

Спочатку створюю не відсортований список, який складається з словників з двома параметрами. Після цього створюю функцію, в який буде проводитися тестування сортування з lambda-функцією. Для зручності створюю змінну op, для реалізації різних операцій та структуру match. При виводі кожного елемента списку який сортується через функцію sorted з ключем lambda за допомогою якого можна виконати сортування без використання окремої функції.

Текст програми:

|  |
| --- |
| students = [{'name':"Ivan", 'mark':82},           {'name':"Bob", 'mark':93},           {'name':"Anna", 'mark':66},           {'name':"Zybr", 'mark':81},           {'name':"Michael", 'mark':83}]  def sortlamb(students):      while True:          op=input("Введіть операцію: sn - для сортування відносно ім'я | sm - для сортування відносно оцінок | ex - для виходу: ")          match op:              case "sn" | "SN":                  for elem in sorted(students, key=lambda student: student["name"]):                      print(f"| Ім'я опитаного: {elem['name']} | Оцінка опитаного: {elem['mark']} ")                  print("Сортований список за ім'ям \n")              case "sm" | "SM":                  for elem in sorted(students, key=lambda student: student["mark"], reverse=True):                      print(f"| Ім'я опитаного: {elem['name']} | Оцінка опитаного: {elem['mark']} ")                  print("Сортований список за оцінкою \n")              case "ex" | "EX":                  break              case \_:                  print("Невірно вказана операція")  sortlamb(students) |

Посилання на github:

Знімок екрану з посилання на github:

**Механізм логування всіх дій, що виконує програма калькулятор.**

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:

За основу візьму програму калькулятор з попередньої теми. Отже маю три файли: askdata.py, func.py, task2.py. Додам до aksdata.py ще одну функцію, в якій визначається операція. В основний файл додаю open з ключем, який використовується для відкриття файлу і його редагування, без видалення попередніх логів. У циклі калькулятору додаю .write(), який використовується для записування вказаних даних у файл. Після обчислень, додаю умову, щоб коли відбувалося ділення на 0 про це повідомлялося. У кінець програми приписую .close(), який завершує сеанс з відкритим файлом.

Текст програми:

|  |
| --- |
| askdata.py  def askData():      while True:          try:              a = float(input(" Введіть числове значення: "))          except ValueError:              print(" Помилка, введене значення не число ")          else:              break      return a  def askOp():      while True:          op = input(" Введіть операцію [+ - \* / ], або ex, для того щоб завершити операцію: ")          if op in ["+", "-", "\*", "/", "ex"]:              return op          else:              print("Помилка, не вірно вказано операцію")  func.py  def add(a, b):      a = a + b      return a  def rem(a, b):      a = a - b      return a  def mult(a,b):      a = a \* b      return a  def div(a,b):      try:          a = a / b      except ZeroDivisionError:          print(" Помилка, ділення на нуль неможливе")      return a  task2.py  from askdata import askData,askOp  from func import add,rem,mult,div  def calc(a):      file.write(f"Begin of calc \n")      while True:          op = askOp()          a\_prev = a          match op:              case "ex":                  file.write(f"End of calc\n")                  return a              case "+":                  a,b = add(a,askData())              case "-":                  a,b = rem(a,askData())              case "\*":                  a,b = mult(a,askData())              case "/":                  a,b = div(a,askData())          print("Значення a на даний момент: ", a)          if op == "/" and b == 0:              file.write(f"{a\_prev} {op} {b} = Division is impossible. \n")          else:              file.write(f"{a\_prev} {op} {b} = {a}\n")  file = open("task2log", "a")  print(f"Фінальний результат: {calc(askData())}")  file.close() |

Посилання на github:

Знімок екрану з посилання на github: