**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:

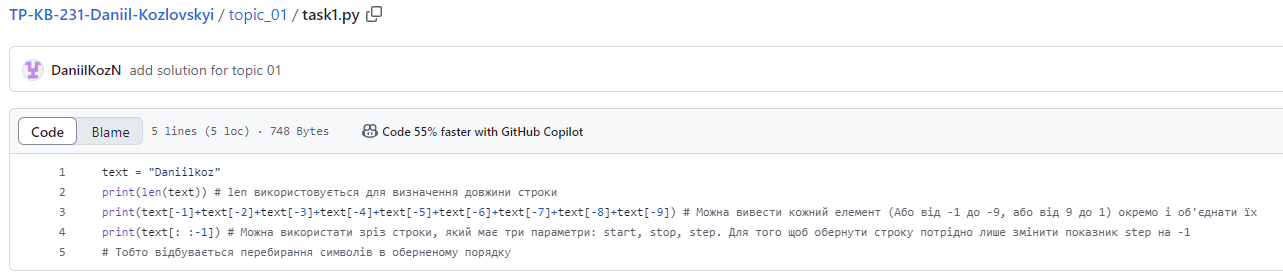
Існує 2 методи для обернення рядку у зворотному напрямку: Можна виводити кожний елемент окремо і об’єднати їх. Або можна використовувати зріз строки, вказавши в 3 параметрі функції (step) значення -1, тобто перебирання символів у оберненому напрямку.

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:

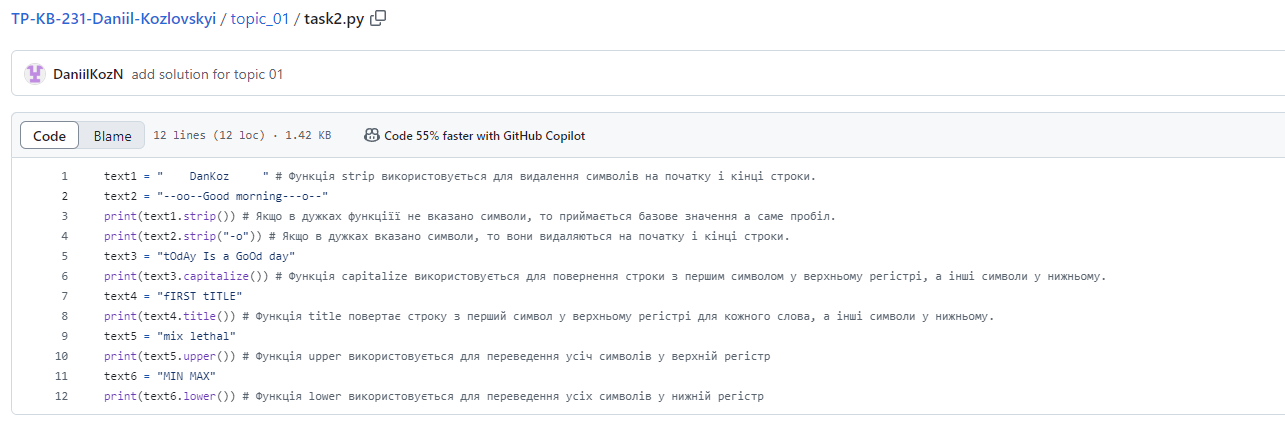
Для кожної функції створюю окремий рядок, який найкраще проілюструє роботу функції. Функція strip, використовують для видалення символів на початку і кінці строки, якщо в дужках вказати символи то видаляться будуть вони, тому створюю два рядка: Перший з великою кількістю пробілів, а інший з символами «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:

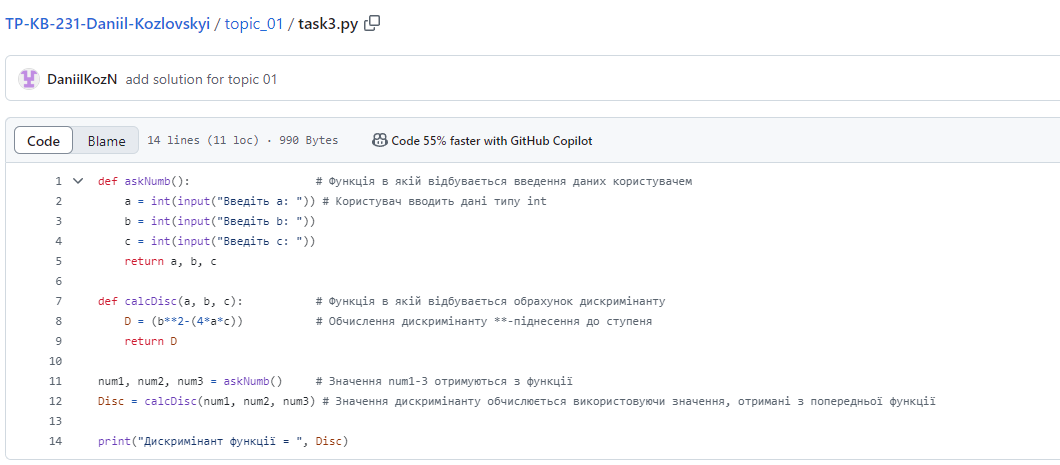
Для реалізації завдання було написано дві функції власного виробництва: Перша необхідна для вводу даних ( a,b,c ), а друга обраховує дискримінант. Після функцій вказано змінні, які беруть дані з функцій. Після всіх обрахунків виводиться результат.

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:

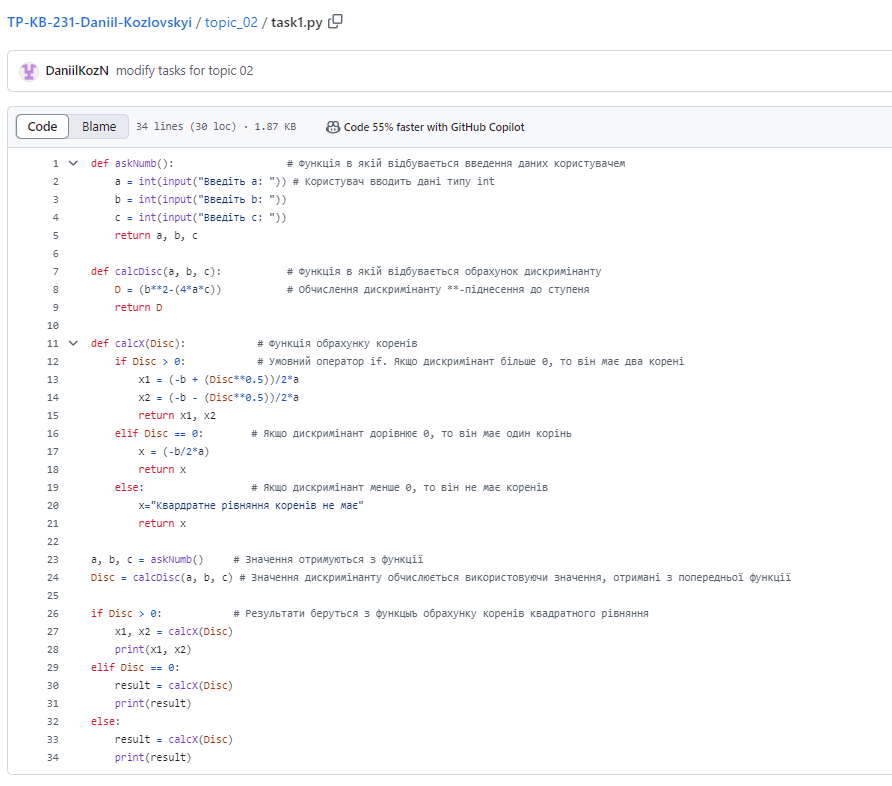
Для реалізації завдання було використано код, який був написаний у попередній темі, проте він закінчується на обрахунках дискримінанту. Тому наступним кроком буде написання коду у якому відбувається обрахунок коренів. Але кількість коренів залежить від значення дискримінанту. Тому використовую умовні 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:

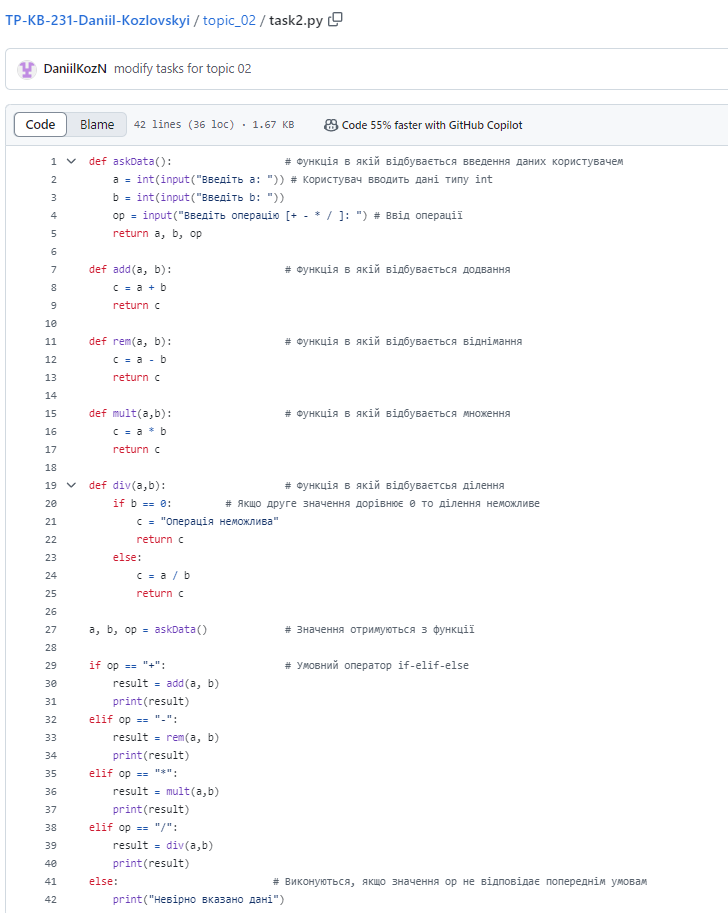
Для реалізації завдання необхідно було створити декілька функцій: Функція для введення даних (перше число, друге число і операція), окремі функції для кожної операції (додавання, віднімання, множення, ділення). Також потрібно враховувати, що ділити на 0 неможна тому в функції ділення потрібно додати 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:

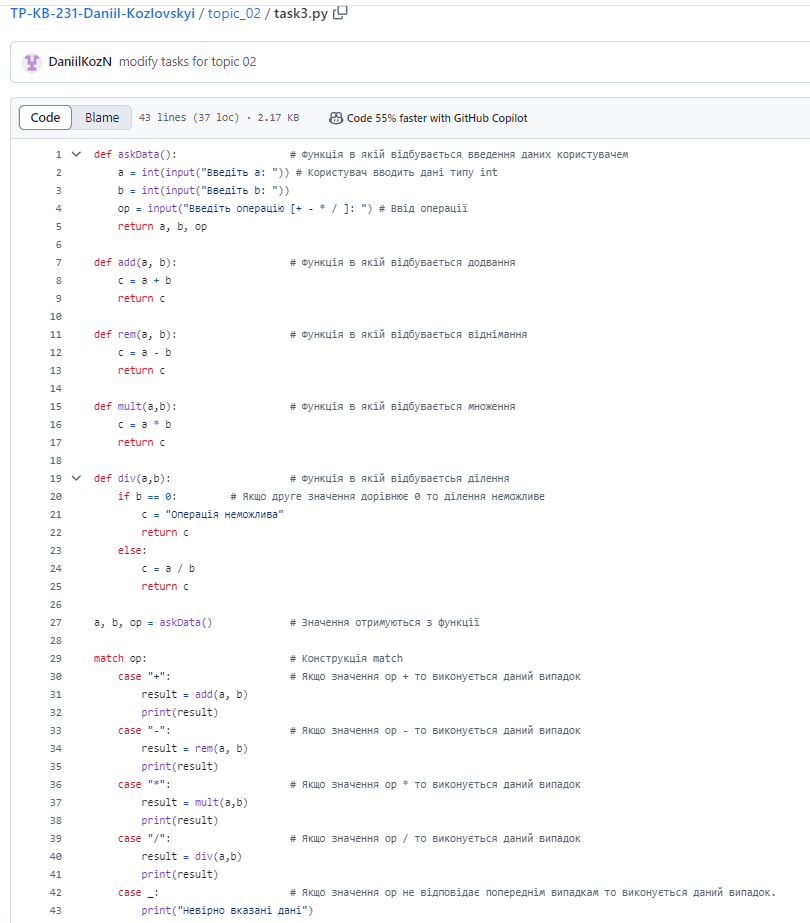
Для реалізації завдання можна використати код з попереднього, так як основною зміню є використання match конструкцій замість if else. Отже копіюю функцію для введення даних і функції операцій. В цьому завданні використовую match конструкцію для того щоб розписати усі випадки, які може вказати користувач 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:

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:

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:

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:

Знімок екрану з посилання на github: