**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:

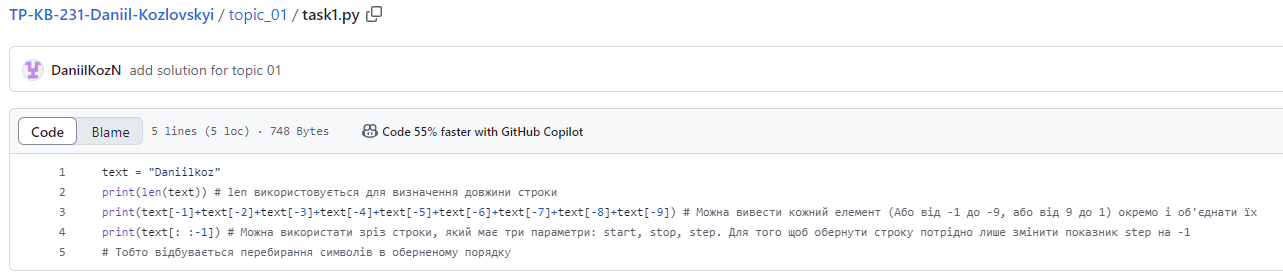
Існує 2 методи для обернення рядку у зворотному напрямку: Можна виводити кожний елемент окремо і об’єднати їх. Або можна використовувати зріз строки, вказавши в 3 параметрі функції (step) значення -1, тобто перебирання символів у оберненому напрямку.

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:

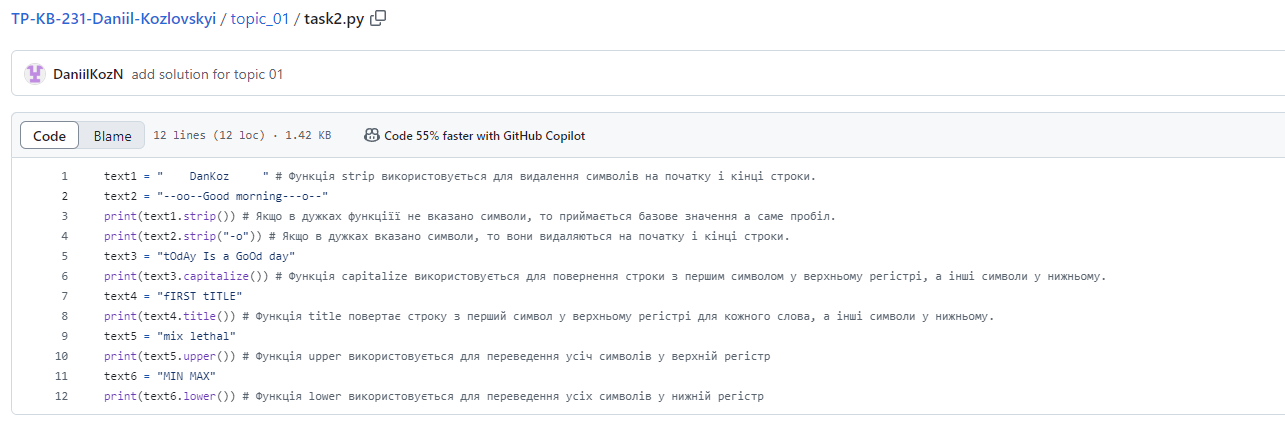
Для кожної функції створюю окремий рядок, який найкраще проілюструє роботу функції. Функція strip, використовують для видалення символів на початку і кінці строки, якщо в дужках вказати символи то видаляться будуть вони, тому створюю два рядка: Перший з великою кількістю пробілів, а інший з символами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

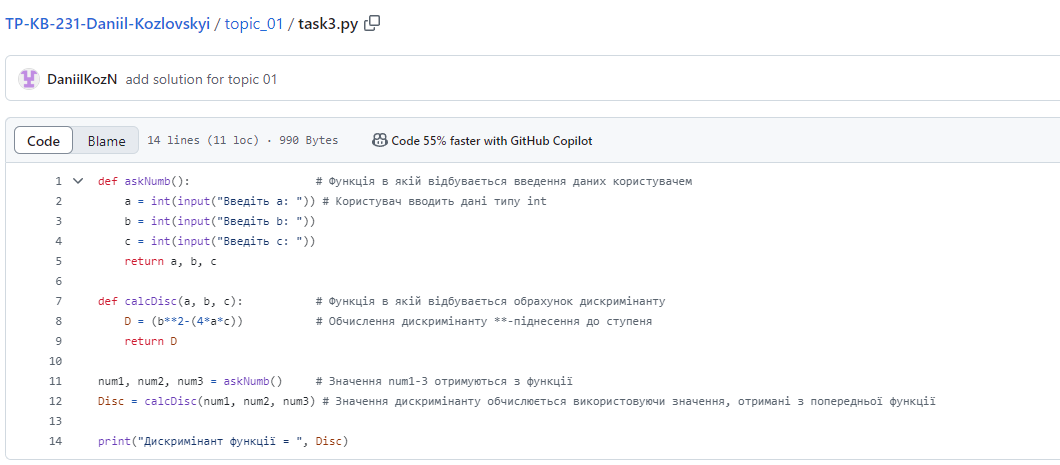
Для реалізації завдання було написано дві функції власного виробництва: Перша необхідна для вводу даних ( a,b,c ), а друга обраховує дискримінант. Після функцій вказано змінні, які беруть дані з функцій. Після всіх обрахунків виводиться результат.

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:

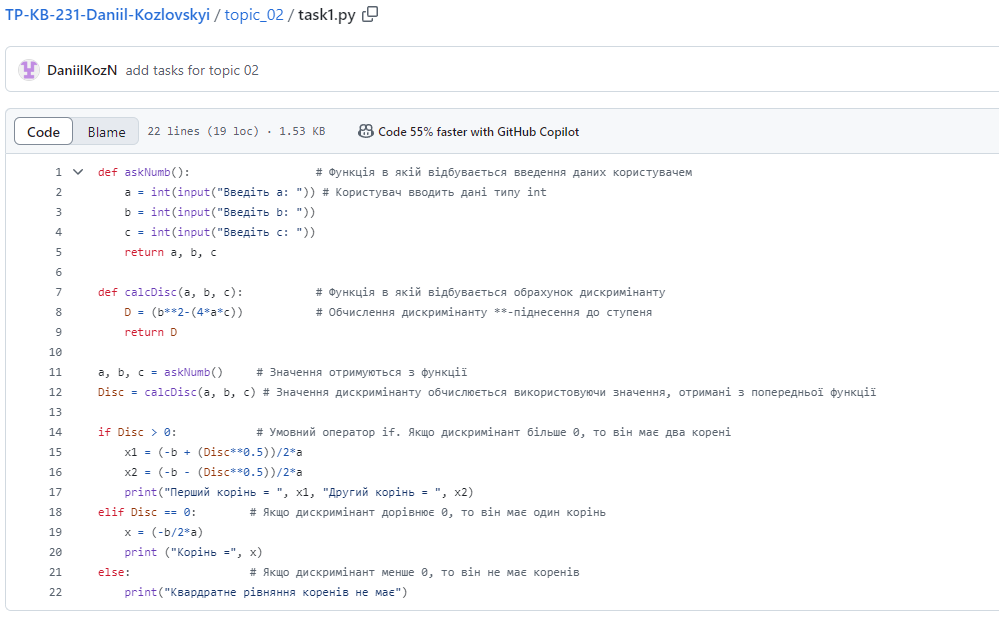
Для реалізації завдання було використано код, який був написаний у попередній темі, проте він закінчується на обрахунках дискримінанту. Тому наступним кроком буде написання коду у якому відбувається обрахунок коренів. Але кількість коренів залежить від значення дискримінанту. Тому використовую умовні 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    a, b, c = askNumb()  Disc = calcDisc(a, b, c)  if Disc > 0:  x1 = (-b + (Disc\*\*0.5))/2\*a  x2 = (-b - (Disc\*\*0.5))/2\*a  print("Перший корінь = ", x1, "Другий корінь = ", x2)  elif Disc == 0:  x = (-b/2\*a)  print ("Корінь =", x)  else:  print("Квардратне рівняння коренів не має") |

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:

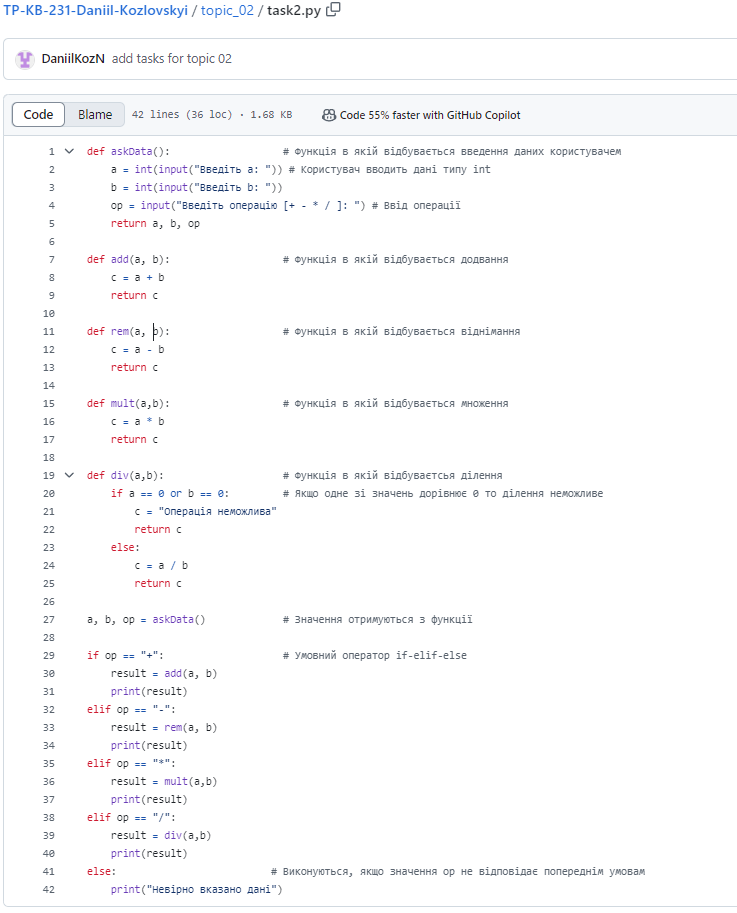
Для реалізації завдання необхідно було створити декілька функцій: Функція для введення даних (перше число, друге число і операція), окремі функції для кожної операції (додавання, віднімання, множення, ділення). Також потрібно враховувати, що ділити на 0 неможна тому в функції ділення потрібно додати умовні переходи, щоб при умові, якщо одне з чисел 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 a == 0 or 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 a == 0 or 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:

