**Звіт про виконання практичних завдань до лекцій з курсу Технології програмування на мові Python**

Звіт до Теми №1

Функції та змінні

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

**Перетворення рядка**

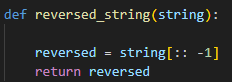
Необхідно рядок, що має вигляд "abcdefg123" перетворити наступним чином "321gfedcba", вважаючи сталою довжину рядку в 10 символів.

Хід виконання завдання:

1. Створив змінну vers\_1



1. Створив функції яка приймає як аргумент якусь змінну та за допомогою оператора нарізки перетворює рядок задом на перед і повертає його



1. Вивів змунну яку повертає функція у консоль



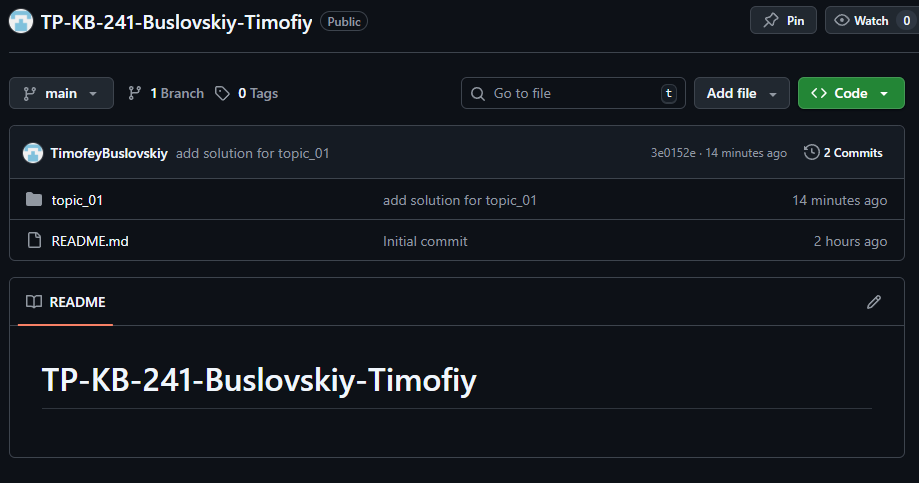
Текст програми:

|  |
| --- |
| vers\_1 = 'abcdefg123'    def reversed\_string(string):        reversed = string[:: -1]      return reversed  print("This is reversed string: " + reversed\_string(vers\_1)) |

Посилання на github:

https://github.com/TimofeyBuslovskiy/TP-KB-241-Buslovskiy-Timofiy

Знімок екрану з посилання на github:



**Тестування функцій що працюють з рядками**

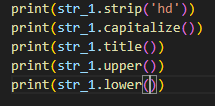
Виконати тестування функцій, що працюють з рядками: strip(), capitalize(), title(), upper(), lower().

Хід виконання завдання:

1. Створив змінну str\_1



1. Виводив значення цієї змінної використовуючи функції



Текст програми:

|  |
| --- |
| str\_1 = 'hello world'  print(str\_1.strip('hd'))  print(str\_1.capitalize())  print(str\_1.title())  print(str\_1.upper())  print(str\_1.lower()) |

Посилання на github:

<https://github.com/TimofeyBuslovskiy/TP-KB-241-Buslovskiy-Timofiy>

**Функція пошуку дискримінанту квадратного рівняння**

Написати функцію пошуку дискримінанту квадратного рівняння.

Хід виконання завдання:

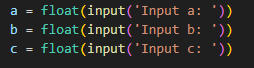
1. Створив функцію find\_discriminant яка приймає як аргумент данні



1. Далі оголосив змінну яка обробляє ці данні виконуючи формулу дискримінанту та повертає цю змінну



1. Далі оголисив три змінні які запитують данні та зводять їх до типу данних number за допомогою функції float



1. Наступним кроком оголосив змінну яка викликає функцію find\_discriminant та приймає в аргумент вище наведені змінні



1. Далі вивів цю змінну в консоль



Текст програми:

|  |
| --- |
| def find\_discriminant (a, b, c) :      discriminant = b\*\*2 - 4\*a\*c      return discriminant  a = float(input('Input a: '))  b = float(input('Input b: '))  c = float(input('Input c: '))  disc = find\_discriminant(a, b, c)  print( disc ) |

Посилання на github:

<https://github.com/TimofeyBuslovskiy/TP-KB-241-Buslovskiy-Timofiy>

**Висновки**

Під час виконання цього практичного завдання я навчився використовувати різні функції для виконання різних операцій зі стрічками,

Перетворювати рядок за допомогою оператора нарізки та створив функцію яка шукає дискримінант.

**Звіт до теми №2**

**Умовний перехід**

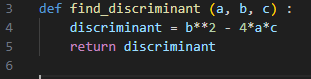
Під час виконання практичного завдання до Теми №2 було надано варіанти рішення до наступних задач:

**Функція пошуку коренів квадратного рівняння**

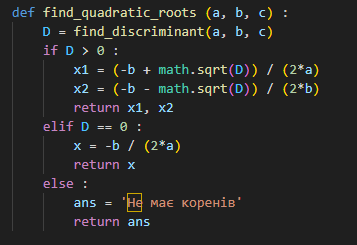
1. Імпортував бібліотеку math для того щоб у майбутньому шкуати квадратні корені



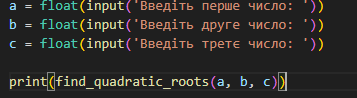
1. Скопіював функцію пошуку дискримінанту з попереднього практичного заняття



1. Оголосив функцію find\_quadratic\_roots у який викликається функція find discriminant і за допомогою умовних операторів роблю перевірку коренів дискримінанту



1. Далі оголосив змінні a, b, c які збирають данні у користувача і зводять їх до типу даних number



Останнім кроком викликається функція find\_quadratic\_roots у яку передаються вище згадані змінні

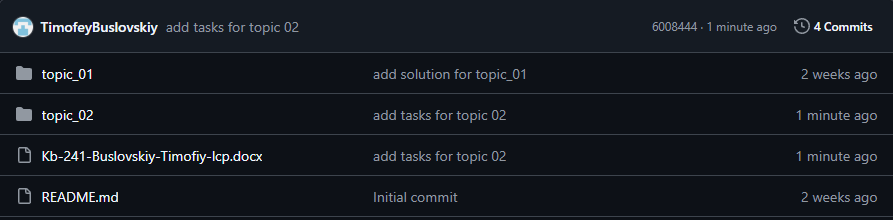
Текст програми:

|  |
| --- |
| import math  def find\_discriminant (a, b, c) :      discriminant = b\*\*2 - 4\*a\*c      return discriminant  def find\_quadratic\_roots (a, b, c) :      D = find\_discriminant(a, b, c)      if D > 0 :          x1 = (-b + math.sqrt(D)) / (2\*a)          x2 = (-b - math.sqrt(D)) / (2\*b)          return x1, x2      elif D == 0 :          x = -b / (2\*a)          return x      else :          ans = 'Не має коренів'          return ans    a = float(input('Введіть перше число: '))  b = float(input('Введіть друге число: '))  c = float(input('Введіть третє число: '))  print(find\_quadratic\_roots(a, b, c)) |

Посилання на github:

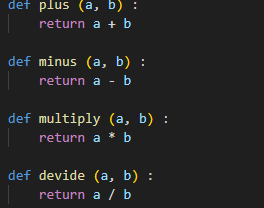
https://github.com/TimofeyBuslovskiy/TP-KB-241-Buslovskiy-Timofiy

Знімок екрану з посилання на github:

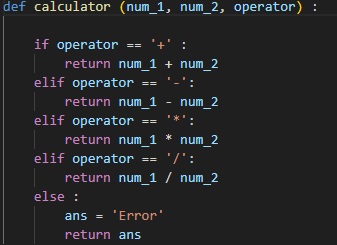


**Калькулятор з використанням If Else**

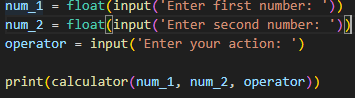
1. Оголосив функції які повертають дії між аргументами які приймаються



1. Оголосив функцію calculator яка приймає в аргументи числа та оператор та робить перевірку



1. Оголосив змінні який приймають данні від користувача



Та виконується виклик функціїї calculator яка приймає як аргумент вище згадані змінні

Текст програми:

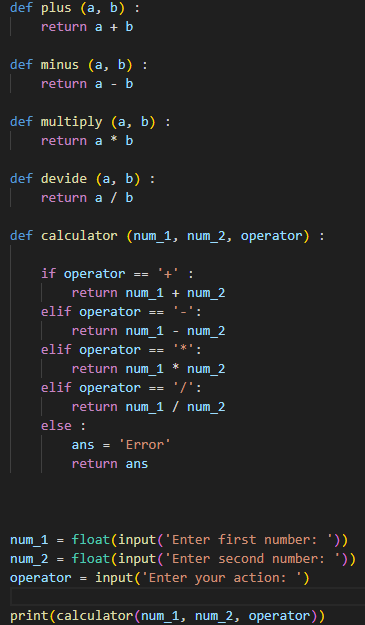
|  |
| --- |
| def plus (a, b) :      return a + b  def minus (a, b) :      return a - b  def multiply (a, b) :      return a \* b  def devide (a, b) :      return a / b  def calculator (num\_1, num\_2, operator) :      if operator == '+' :          return num\_1 + num\_2      elif operator == '-':          return num\_1 - num\_2      elif operator == '\*':          return num\_1 \* num\_2      elif operator == '/':          return num\_1 / num\_2      else :          ans = 'Error'          return ans    num\_1 = float(input('Enter first number: '))  num\_2 = float(input('Enter second number: '))  operator = input('Enter your action: ')  print(calculator(num\_1, num\_2, operator)) |

Посилання на github:

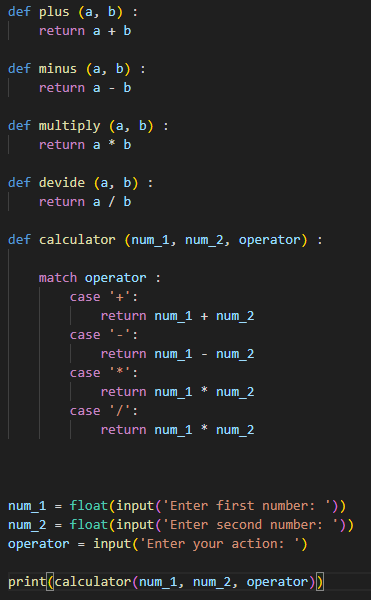
<https://github.com/TimofeyBuslovskiy/TP-KB-241-Buslovskiy-Timofiy>

**Калькулятор на основі конструкції Match**

1. Скопіював код з попереднього завдання



1. Зімнив перевірку if else на конструкцію match



Текст програми:

|  |
| --- |
| def plus (a, b) :      return a + b  def minus (a, b) :      return a - b  def multiply (a, b) :      return a \* b  def devide (a, b) :      return a / b  def calculator (num\_1, num\_2, operator) :      match operator :          case '+':              return num\_1 + num\_2          case '-':              return num\_1 - num\_2          case '\*':              return num\_1 \* num\_2          case '/':              return num\_1 \* num\_2    num\_1 = float(input('Enter first number: '))  num\_2 = float(input('Enter second number: '))  operator = input('Enter your action: ')  print(calculator(num\_1, num\_2, operator)) |

Посилання на github:

<https://github.com/TimofeyBuslovskiy/TP-KB-241-Buslovskiy-Timofiy>

**Висновок**

Під час виконання практичного завдання я мінімально попрацював з бібліотекою math, використовував логічні оператори if else та match, закріпив свої знання з роботи функцій та використовував логічне мишлення для досягнення поставлених цілей

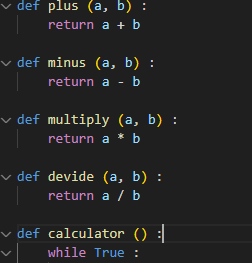
**Звіт до теми №2**

**Умовний перехід**

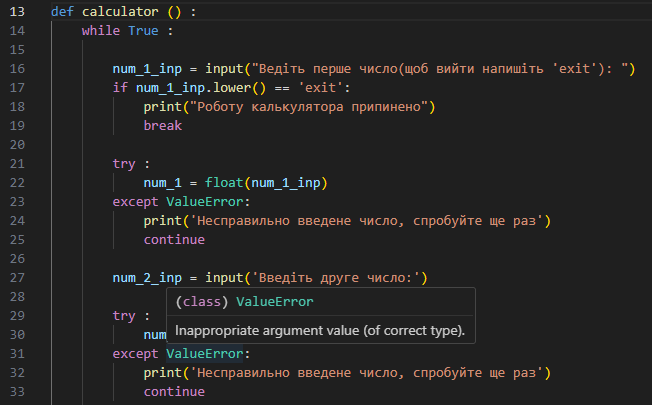
Під час виконання практичного завдання до Теми №2 було надано варіанти рішення до наступних задач:

**Калькулятор з постійним запитом на введення**

1. Скопіював функції оперування з числами з попереднбої роботи

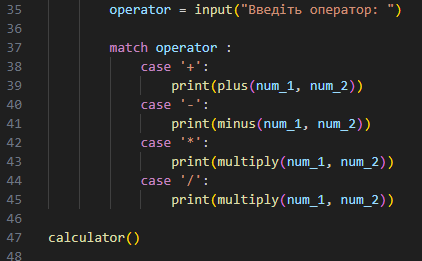


1. Оголосив функцію calculator та оголосив цикл в середині функції



В циклі оголосив змінну num\_1\_inp і робив перевірку, якщо зведена змінна до нижнього регістру num\_1\_inp дорівнює exit, то робота калькулятора припиняється. Далі перевірка на правильність введеного числа та зведення введеного числа до типу даних numner

1. Наступним кроком скопіював код з попередньої практичної роботи(трохи змінивши його), застосувавги його в середині коду



Текст програми:

|  |
| --- |
| def plus (a, b) :      return a + b  def minus (a, b) :      return a - b  def multiply (a, b) :      return a \* b  def devide (a, b) :      return a / b  def calculator () :      while True :          num\_1\_inp = input("Ведіть перше число(щоб вийти напишіть 'exit'): ")          if num\_1\_inp.lower() == 'exit':              print("Роботу калькулятора припинено")              break          try :              num\_1 = float(num\_1\_inp)          except ValueError:              print('Несправильно введене число, спробуйте ще раз')              continue          num\_2\_inp = input('Введіть друге число:')          try :              num\_2 = float(num\_2\_inp)          except ValueError:              print('Несправильно введене число, спробуйте ще раз')              continue          operator = input("Введіть оператор: ")          match operator :              case '+':                  print(plus(num\_1, num\_2))              case '-':                  print(minus(num\_1, num\_2))              case '\*':                  print(multiply(num\_1, num\_2))              case '/':                  print(multiply(num\_1, num\_2))    calculator() |

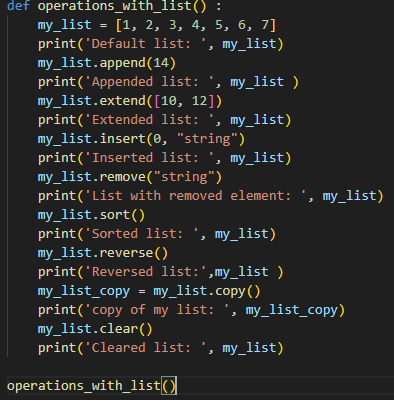
Посилання на github:

https://github.com/TimofeyBuslovskiy/TP-KB-241-Buslovskiy-Timofiy

Знімок екрану з посилання на github:

**Програма тестування функцій**

1. Оголосив функцію де використовував різні функції для оперування з масивами



В кінці викликав функцію

Текст програми:

|  |
| --- |
| def operations\_with\_list() :      my\_list = [1, 2, 3, 4, 5, 6, 7]      print('Default list: ', my\_list)      my\_list.append(14)      print('Appended list: ', my\_list )      my\_list.extend([10, 12])      print('Extended list: ', my\_list)      my\_list.insert(0, "string")      print('Inserted list: ', my\_list)      my\_list.remove("string")      print('List with removed element: ', my\_list)      my\_list.sort()      print('Sorted list: ', my\_list)      my\_list.reverse()      print('Reversed list:',my\_list )      my\_list\_copy = my\_list.copy()      print('copy of my list: ', my\_list\_copy)      my\_list.clear()      print('Cleared list: ', my\_list)  operations\_with\_list() |

Посилання на github:

<https://github.com/TimofeyBuslovskiy/TP-KB-241-Buslovskiy-Timofiy>