**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. Спочатку створила змінну string та задала їй значення вказане в завданні, а саме "abcdefg123".
  2. Далі використала функцію print, що виводить на екран значення вказане в дужках. В дужках виконала string[::-1], що бере значення змінної string та перевертає його в зворотньому напрямку. Синтаксиз виразу [::-1]: [start:stop:step], де start - індекс, з якого починається сріз, stop - індекс, на якому закінчується сріз, step - крок, на який здійснюється переміщення від start до stop. У нашому випадку start і stop не вказані, тобто ми беремо всю послідовність, step встановлено в -1, тобто ми переміщуємося в зворотному напрямку.

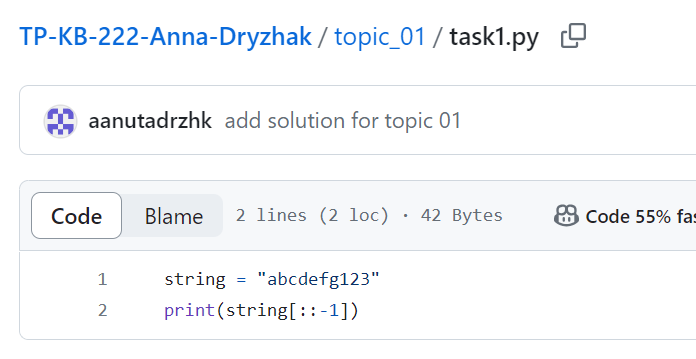
Текст програми:

|  |
| --- |
| string = "abcdefg123"  print(string[::-1]) |

Посилання на github:

[TP-KB-222-Anna-Dryzhak/topic\_01/task1.py at main · aanutadrzhk/TP-KB-222-Anna-Dryzhak (github.com)](https://github.com/aanutadrzhk/TP-KB-222-Anna-Dryzhak/blob/main/topic_01/task1.py)

Знімок екрану з посилання на github:



**Виконання функцій**

Необхідно виконати такі функції як: strip, capitalize, title, upper, lower та інші.

Хід виконання завдання:

Створила 6 однакових макетів коду для виконання 6 різних функцій. В макеті спочатку створено змінну pib значення якої дорівнює моєму ПІБ. За допомогою фунції print(pib) вивожу на екран своє ПІБ. Далі сворюю змінну testpib, що дорівнює значенню змінної pib з використанням певної функції для роботи з рядками. І наступною функцією print(testpib) я вивожу на екран значення змінної testpib. В кінці також додала print(“ “), щоб між виконанням різних функцій була відстань.

Значення 6 різних функцій, що я використала в завданні №2:

* + strip() - видаляє пробіли з початку та кінця рядка
  + сapitalize() - перша літера рядка стає великою
  + title() - перша літера кожного слова стає великою
  + upper() - всі літери у рядку стають великими
  + lower() - всі літери у рядку стають маленькими
  + swapcase() - всі великі літери у рядку стають маленькими, а всі маленькі - великими

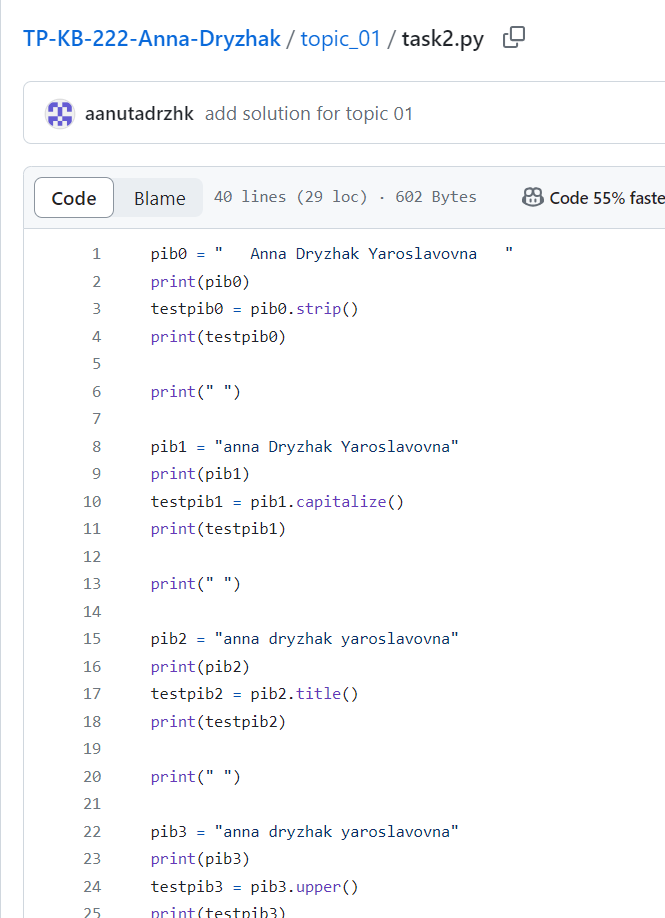
Текст програми:

|  |
| --- |
| pib0 = "Anna Dryzhak Yaroslavovna"  print(pib0)  testpib0 = pib0.strip()  print(testpib0)    print(" ")    pib1 = "anna Dryzhak Yaroslavovna"  print(pib1)  testpib1 = pib1.capitalize()  print(testpib1)    print(" ")    pib2 = "anna dryzhak yaroslavovna"  print(pib2)  testpib2 = pib2.title()  print(testpib2)    print(" ")    pib3 = "anna dryzhak yaroslavovna"  print(pib3)  testpib3 = pib3.upper()  print(testpib3)    print(" ")    pib4 = "ANNA DRYZHAK YAROSLAVOVNA"  print(pib4)  testpib4 = pib4.lower()  print(testpib4)    print(" ")    pib5 = "Anna Dryzhak Yaroslavovna"  print(pib5)  testpib5 = pib5.swapcase()  print(testpib5) |

Посилання на github:

[TP-KB-222-Anna-Dryzhak/topic\_01/task2.py at main · aanutadrzhk/TP-KB-222-Anna-Dryzhak (github.com)](https://github.com/aanutadrzhk/TP-KB-222-Anna-Dryzhak/blob/main/topic_01/task2.py)

Знімок екрану з посилання на github:



**Пошук дискримінанту квадратного рівняння**

Необхідно написати код, що буде шукати дискримінант квадратного рівняння, приймаючи значення a, b, c у користувача.

Хід виконання завдання:

* 1. За допомогою функції input("Please enter value a/b/c: "), користувач вводить значення для змінних a, b, c і ці значення конвертуються у цілі числа за допомогою int().
  2. Створюємо функцію findD(a, b ,c). Ця функція приймає значення a, b і c, що ввів користувач.
  3. В змінній result за допомогою формули дискримінанта b^2 – 4ac він вираховується.
  4. За допомогою слова return значення дискримінанта повертається з функції.
  5. Під кінець за допомогою фунції print(findD(a, b ,c)) результат пошуку дискримінанту виводиться на екран.

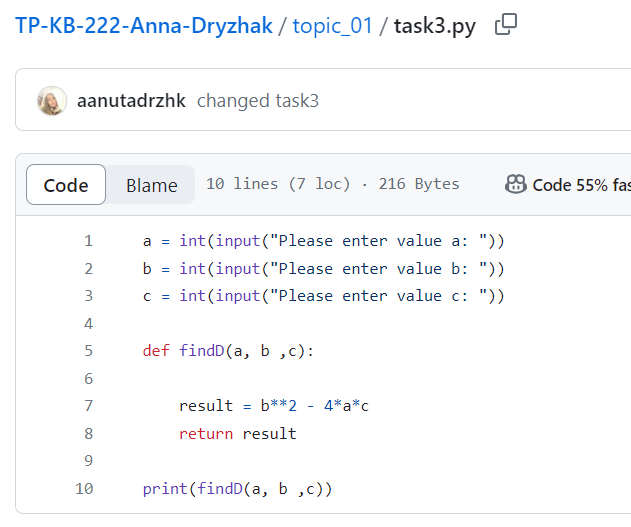
Текст програми:

|  |
| --- |
| a = int(input("Please enter value a: "))  b = int(input("Please enter value b: "))  c = int(input("Please enter value c: "))    def findD(a, b ,c):    result = b\*\*2 - 4\*a\*c  return result    print(findD(a, b ,c)) |

Посилання на github:

[TP-KB-222-Anna-Dryzhak/topic\_01/task3.py at main · aanutadrzhk/TP-KB-222-Anna-Dryzhak (github.com)](https://github.com/aanutadrzhk/TP-KB-222-Anna-Dryzhak/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:

**Пошук коренів квадратного рівняння**

Необхідно написати код пошуку коренів квадратного рівняння, враховуючи значення дискримінанту.

Хід виконання завдання:

* 1. Спочатку за допомогою функції input запрошуємо у користувача ввести значення a, b, c і переводимо ці значення в числа за допомогою int.
  2. Далі як в завданні з минулої теми робимо функцію для пошуку дискримінанту.
  3. Функція FindRoots(a, b, c) викликає findD(a, b, c) для обчислення дискримінанту. Потім вона перевіряє дискримінант:
     + Якщо D більше 0, то рівняння має два корені, і функція обчислює їх за формулами та повертає значення x1, x2 .
     + Якщо D дорівнює 0, то рівняння має один корінь, і функція повертає значення x1.
     + У всіх інших випадках рівняння не має коренів, і функція повертає None, тобто нічого.
  4. Результат виконання функції FindRoots(a, b, c) зберігається в змінній roots.
  5. Далі відбувається перевірка, чи має рівняння корені:
     + Якщо roots не є None, далі перевіряємо якщо roots є кортежем, то виводяться два корені x1 і x2. Якщо roots не є кортежем, то виводиться один корінь x1.
     + Якщо roots дорівнює None, то виводиться повідомлення, що рівняння не має коренів.

Текст програми:

|  |
| --- |
| a = int(input("Будь ласка, введіть значення a: "))  b = int(input("Будь ласка, введіть значення b: "))  c = int(input("Будь ласка, введіть значення c: "))    def findD(a, b ,c):  D = b\*\*2 - 4\*a\*c  return D      def FindRoots(a, b, c):  D = findD(a, b, c)  if D > 0:  x1 = (-b + D\*\*0.5) / 2\*a  x2 = (-b - D\*\*0.5) / 2\*a  return x1, x2  elif D == 0:  x1 = -b / 2\*a  return x1  else:  return None    roots = FindRoots(a, b, c)  if roots is not None:  if isinstance(roots, tuple):  x1, x2 = roots  print("Рівняння має два корені:")  print("x1 =", x1)  print("x2 =", x2)  else:  x1 = roots  print("Рівняння має один корінь:")  print("x1 =", x1)  else:  print("Рівняння не має коренів.") |

Посилання на github:

[TP-KB-222-Anna-Dryzhak/topic\_02/task1.py at main · aanutadrzhk/TP-KB-222-Anna-Dryzhak (github.com)](https://github.com/aanutadrzhk/TP-KB-222-Anna-Dryzhak/blob/main/topic_02/task1.py)

Знімок екрану з посилання на github:



**Калькулятор**

Необхідно написати код для обчислення операції додавання, віднімання, множення та ділення.

Хід виконання завдання:

* 1. Спочатку за допомогою функції input запрошуємо у користувача ввести значення першого числа, другого та операцію, яку треба зробити з ними і переводимо перші два значення в числа за допомогою int.
  2. Далі створюємо 4 аналогічні функції додавання, віднімання, множення та ділення. У функцію ділення додаємо умову, якщо число b дорівнює 0, то виводимо текст, що ділення на 0 не можливе. В інших випадках виконуємо операцію ділення.
  3. Якщо операція, що ввів користувач дорівнює “+”, тобто додавання, то в змінній result виконуємо створену нами функцію addition(a,b). Аналогічно робимо з 3 іншими значеннями (-,\*,/). А якщо користувач ввів якесь інше значення, то в змінну result записуємо, що операція введена неправильно.
  4. За допомогою функції print виводимо значення змінної result на екран.

Текст програми:

|  |
| --- |
| a = int(input("Введіть перше число: "))  b = int(input("Введіть друге число: "))  op = (input("Операція, яку треба виконати: "))    def addition(a,b):  return a + b  def subtraction(a,b):  return a - b  def multiplication(a,b):  return a \* b  def division(a,b):  if b == 0:  return "Ділення на 0 не можливе"  return a / b    if op == "+":  result = addition(a,b)  elif op == "-":  result = subtraction(a,b)  elif op == "\*":  result = multiplication(a,b)  elif op == "/":  result = division(a,b)  else:  result = "Неправильно введена операція, спробуйте \*,/,-,+"    print(result) |

Посилання на github:

[TP-KB-222-Anna-Dryzhak/topic\_02/task2.py at main · aanutadrzhk/TP-KB-222-Anna-Dryzhak (github.com)](https://github.com/aanutadrzhk/TP-KB-222-Anna-Dryzhak/blob/main/topic_02/task2.py)

Знімок екрану з посилання на github:



**Калькулятор 2**

Необхідно написати код для обчислення операції додавання, віднімання, множення та ділення з використанням match.

Хід виконання завдання:

* 1. Частина коду до виконання функції match аналогічна до минулого завдання.
  2. Далі виконуємо функцію match над значенням змінної op:
     + Якщо користувач ввів +, то виводимо на екран результат функції addition(a,b).
     + Якщо користувач ввів -, то виводимо на екран результат функції subtraction(a,b).
     + Якщо користувач ввів \*, то виводимо на екран результат функції multiplication(a,b).
     + Якщо користувач ввів /, то виводимо на екран результат функції division(a,b).
     + А якщо користувач ввів якесь інше значення, то виводимо на екран напис, що операція введена неправильно.

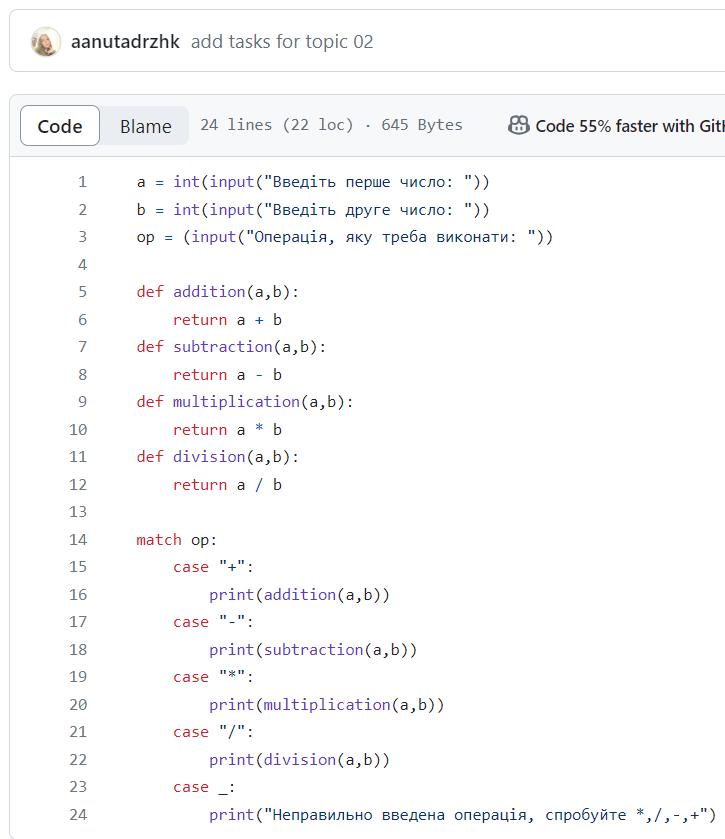
Текст програми:

|  |
| --- |
| a = int(input("Введіть перше число: "))  b = int(input("Введіть друге число: "))  op = (input("Операція, яку треба виконати: "))    def addition(a,b):  return a + b  def subtraction(a,b):  return a - b  def multiplication(a,b):  return a \* b  def division(a,b):  if b == 0:  return "Ділення на 0 не можливе"  return a / b  match op:  case "+":  print(addition(a,b))  case "-":  print(subtraction(a,b))  case "\*":  print(multiplication(a,b))  case "/":  print(division(a,b))  case \_:  print("Неправильно введена операція, спробуйте \*,/,-,+") |

Посилання на github:

[TP-KB-222-Anna-Dryzhak/topic\_02/task3.py at main · aanutadrzhk/TP-KB-222-Anna-Dryzhak (github.com)](https://github.com/aanutadrzhk/TP-KB-222-Anna-Dryzhak/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:

**Калькулятор 3**

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:

* 1. За основу беремо програму калькулятор з минулої теми.
  2. Додаємо цикл while, який виводить на екран, що треба ввести, щоб виконати відповідну операцію та далі відповідно просить ввести значення та розраховує їх. Оскільки цей цикл є безперервним (бо умовою є True), то я додала умову, якщо в операцію було введенно “0”, то цикл закінчується.

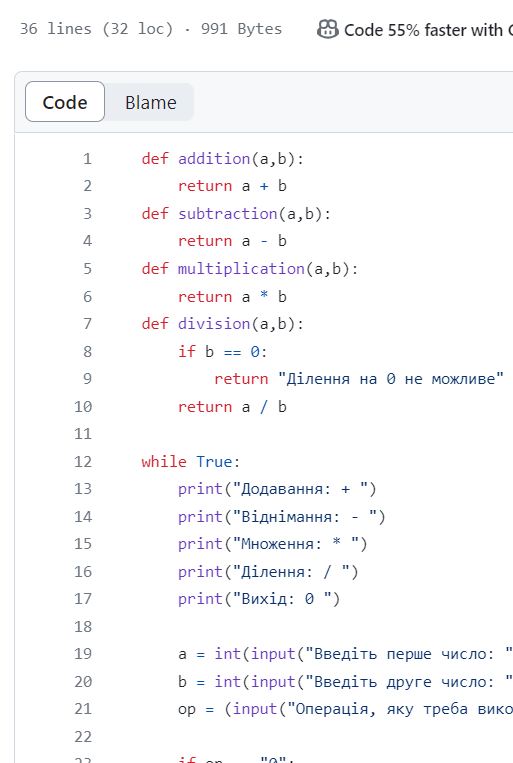
Текст програми:

|  |
| --- |
| def addition(a,b):  return a + b  def subtraction(a,b):  return a - b  def multiplication(a,b):  return a \* b  def division(a,b):  if b == 0:  return "Ділення на 0 не можливе"  return a / b  while True:  print("Додавання: + ")  print("Віднімання: - ")  print("Множення: \* ")  print("Ділення: / ")  print("Вихід: 0 ")  a = int(input("Введіть перше число: "))  b = int(input("Введіть друге число: "))  op = (input("Операція, яку треба виконати: "))  if op == "0":  break  match op:  case "+":  print(addition(a,b))  case "-":  print(subtraction(a,b))  case "\*":  print(multiplication(a,b))  case "/":  print(division(a,b))  case \_:  print("Неправильно введена операція, спробуйте \*,/,-,+") |

Посилання на github:

[TP-KB-222-Anna-Dryzhak/topic\_03/task1.py at main · aanutadrzhk/TP-KB-222-Anna-Dryzhak (github.com)](https://github.com/aanutadrzhk/TP-KB-222-Anna-Dryzhak/blob/main/topic_03/task1.py)

Знімок екрану з посилання на github:



**Тестування функцій 1**

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

Хід виконання завдання:

* 1. Спочатку створюємо змінну list зі значенням списку 1, 2, 3. Та виводимо на екран початковий список.
  2. Тестуємо функції:
     + extend(iterable): дозволяє додати всі елементи в кінець поточного списку.
     + append(value): додає один вказаний елемент в кінець списку.
     + insert(index, value): дозволяє вставити значення value на позицію index у списку.
     + remove(value):видаляє елемент у списку зі значенням value
     + clear(): очищає весь вміст списку.
     + sort(): сортує елементи списку в зростаючому порядку за замовчуванням.
     + reverse(): обертає порядок елементів у списку.
     + copy(): створює та повертає копію списку.

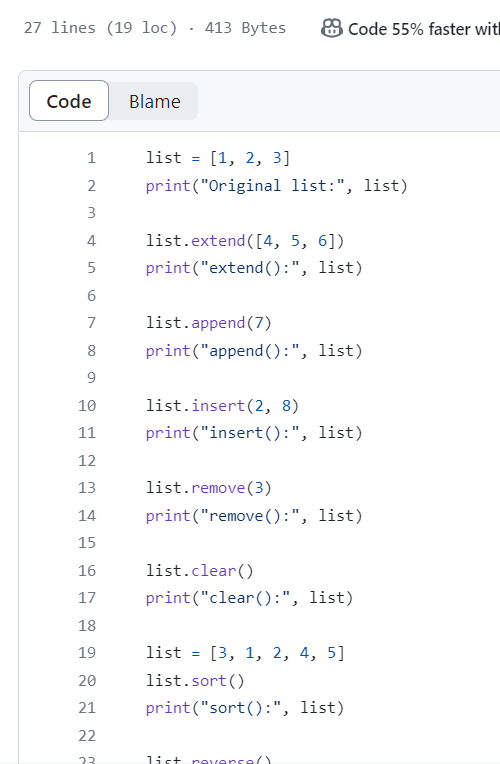
Текст програми:

|  |
| --- |
| list = [1, 2, 3]  print("Original list:", list)  list.extend([4, 5, 6])  print("extend():", list)  list.append(7)  print("append():", list)  list.insert(2, 8)  print("insert():", list)  list.remove(3)  print("remove():", list)  list.clear()  print("clear():", list)  list = [3, 1, 2, 4, 5]  list.sort()  print("sort():", list)  list.reverse()  print("reverse():", list)    list\_copy = list.copy()  print("copy():", list\_copy) |

Посилання на github:

[TP-KB-222-Anna-Dryzhak/topic\_03/task2.py at main · aanutadrzhk/TP-KB-222-Anna-Dryzhak (github.com)](https://github.com/aanutadrzhk/TP-KB-222-Anna-Dryzhak/blob/main/topic_03/task2.py)

Знімок екрану з посилання на github:



**Тестування функцій 2**

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

Хід виконання завдання:

* 1. Спочатку створюємо змінну dict зі значенням списку 1, 2, 3. Та виводимо на екран початковий словник.
  2. Тестуємо функції:
     + update(other\_dict): додавання пар ключ-значення з іншого словника other\_dict до поточного словника. Якщо ключі вже існують у поточному словнику, значення будуть оновлені.
     + del dict[key]: видалення ключа та його відповідного значення зі словника.
     + clear(): очищення всього вмісту словника.
     + keys(): отримання списку всіх ключів у словнику.
     + values(): отримання списку всіх значень у словнику.
     + items(): отримання списку всіх пар ключ-значення у словнику.

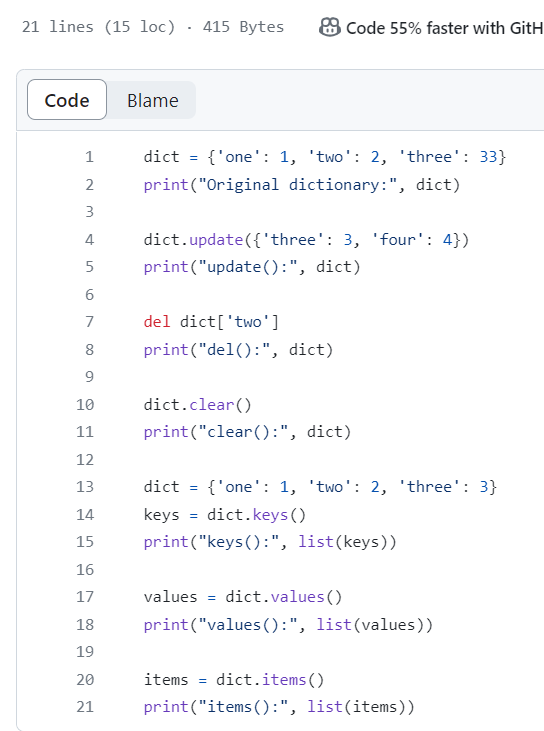
Текст програми:

|  |
| --- |
| dict = {'one': 1, 'two': 2, 'three': 33}  print("Original dictionary:", dict)  dict.update({'three': 3, 'four': 4})  print("update():", dict)  del dict['two']  print("del():", dict)  dict.clear()  print("clear():", dict)  dict = {'one': 1, 'two': 2, 'three': 3}  keys = dict.keys()  print("keys():", list(keys))  values = dict.values()  print("values():", list(values))  items = dict.items()  print("items():", list(items)) |

Посилання на github:

[TP-KB-222-Anna-Dryzhak/topic\_03/task3.py at main · aanutadrzhk/TP-KB-222-Anna-Dryzhak (github.com)](https://github.com/aanutadrzhk/TP-KB-222-Anna-Dryzhak/blob/main/topic_03/task3.py)

Знімок екрану з посилання на github:



**Пошук позицій**

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

Хід виконання завдання:

* 1. Спочатку створюємо список numbers, який містить початкові значення [1, 3, 5].
  2. Далі створюємо безкінечний цикл while True, який продовжується, поки користувач не введе слово "вихід":
     + Програма виводить поточний стан списку numbers, використовуючи print().
     + Користувач повинен ввести нове число або вийти з програми, використовуючи input(). Введене значення зберігається у змінній new\_number.
     + Перевіряється, чи введене значення є "вихід". Якщо так, цикл завершується за допомогою break.
     + Якщо введене значення не є "вихід", то new\_number перетворюється в ціле число за допомогою int().
     + Визначаємо змінну insert\_index, яка дорівнює 0.
     + Потім виконується ітерація по числах у списку numbers за допомогою циклу for. Для кожного числа num у списку порівнюється new\_number та num. Якщо new\_number більше за num, змінна insert\_index збільшується на одиницю. Це визначає позицію, на яку потрібно вставити new\_number.
     + Далі функція insert() використовується для вставки new\_number на визначену позицію insert\_index у списку numbers.
     + Цикл while продовжується, дозволяючи користувачеві вводити інші числа або вийти з програми.

Текст програми:

|  |
| --- |
| numbers = [1, 3, 5,]  while True:  print("Список:", numbers)  new\_number = input("Введіть нове число, чи введіть 'вихід', якщо хочете вийти: ")  if new\_number == 'вихід':  break  new\_number = int(new\_number)  insert\_index = 0  for num in numbers:  if new\_number > num:  insert\_index += 1  numbers.insert(insert\_index, new\_number) |

Посилання на github:

[TP-KB-222-Anna-Dryzhak/topic\_03/task4.py at main · aanutadrzhk/TP-KB-222-Anna-Dryzhak (github.com)](https://github.com/aanutadrzhk/TP-KB-222-Anna-Dryzhak/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:

**Калькулятор 4**

Розширити програму калькулятор функцією запитів від користувача, що обробляє виняткові ситуації.

Хід виконання завдання:

* 1. Частина коду аналогічна до минулої реалізації калькулятора.
  2. Додано функцію value, яка отримує рядок promt та виводить його на екран і очікує від користувача введення цілого числа. Якщо користувач введе ціле число, то це число повертається з функції. А якщо користувач введе рядок або нечислове значення, виникає виняток ValueError і виводиться повідомлення "Неправильне значення (введіть число)", і цикл продовжує очікувати правильного введення від користувача.
  3. Ця функція використовується в змінних для значень a та b.

Текст програми:

|  |
| --- |
| def addition(a,b):  return a + b  def subtraction(a,b):  return a - b  def multiplication(a,b):  return a \* b  def division(a,b):  if b == 0:  return "Ділення на 0 не можливе"  return a / b  def value(promt:str):  while True:  try:  return int(input(promt))  except ValueError:  print("Неправильне значення (введіть число)")    while True:  print("Додавання: + ")  print("Віднімання: - ")  print("Множення: \* ")  print("Ділення: / ")  print("Вихід: 0 ")  a = value("Введіть значення a: ")  b = value("Введіть значення b: ")  op = (input("Операція, яку треба виконати: "))  if op == "0":  break  match op:  case "+":  print(addition(a,b))  case "-":  print(subtraction(a,b))  case "\*":  print(multiplication(a,b))  case "/":  print(division(a,b))  case \_:  print("Неправильно введена операція, спробуйте \*,/,-,+") |

Посилання на github:

[TP-KB-222-Anna-Dryzhak/topic\_04/task1.py at main · aanutadrzhk/TP-KB-222-Anna-Dryzhak (github.com)](https://github.com/aanutadrzhk/TP-KB-222-Anna-Dryzhak/blob/main/topic_04/task1.py)

Знімок екрану з посилання на github:



**Калькулятор 5**

Розширити функцію ділення обробкою виняткової ситуації ділення но нуль.

Хід виконання завдання:

* 1. Частина коду аналогічна до минулої реалізації калькулятора.
  2. Змінено функцію division(a,b). Якщо b не рівне 0, то виконується ділення. А якщо b = 0, то в блоці except обробляємо помилку ZeroDivisionError, що виникає при діленні на 0, а саме, виводим на екран "Ділення на нуль неможливе".

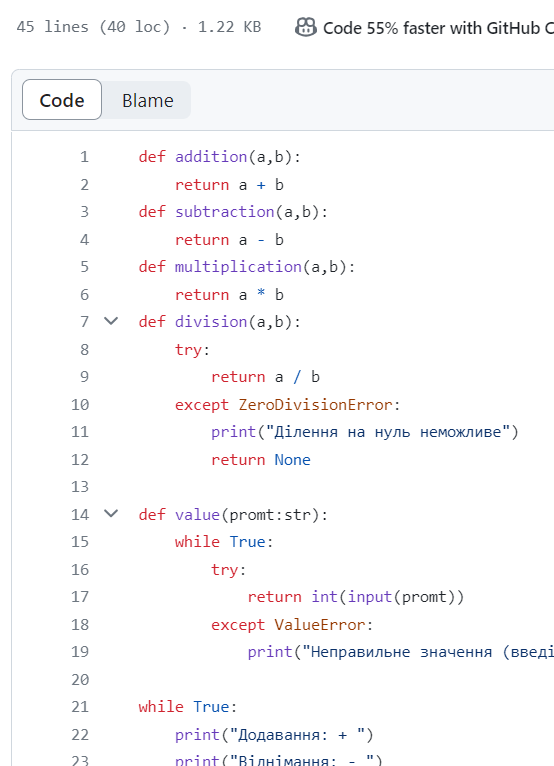
Текст програми:

|  |
| --- |
| def addition(a,b):  return a + b  def subtraction(a,b):  return a - b  def multiplication(a,b):  return a \* b  def division(a,b):  try:  return a / b  except ZeroDivisionError:  print("Ділення на нуль неможливе")  return None    def value(promt:str):  while True:  try:  return int(input(promt))  except ValueError:  print("Неправильне значення (введіть число)")    while True:  print("Додавання: + ")  print("Віднімання: - ")  print("Множення: \* ")  print("Ділення: / ")  print("Вихід: 0 ")  a = value("Введіть значення a: ")  b = value("Введіть значення b: ")  op = (input("Операція, яку треба виконати: "))  if op == "0":  break  match op:  case "+":  print(addition(a,b))  case "-":  print(subtraction(a,b))  case "\*":  print(multiplication(a,b))  case "/":  print(division(a,b))  case \_:  print("Неправильно введена операція, спробуйте \*,/,-,+") |

Посилання на github:

[TP-KB-222-Anna-Dryzhak/topic\_04/task2.py at main · aanutadrzhk/TP-KB-222-Anna-Dryzhak (github.com)](https://github.com/aanutadrzhk/TP-KB-222-Anna-Dryzhak/blob/main/topic_04/task2.py)

Знімок екрану з посилання на github:



**Різноманітні виняткові ситуації**

Ознайомитись зі списком виняткових ситуацій за посиланням <https://docs.python.org/3/library/exceptions.html>.

Хід виконання завдання:

* 1. Перейшла за наданим посиланням.
  2. Ознайомилась з винятковими ситуаціями.
  3. Реалізувала деякі з них:
     + ValueError - функція отримує аргумент правильного типу, але з недійсним значенням.
     + FileNotFoundError - намагаєтеся відкрити файл, який не існує.
     + KeyError - намагаєтеся отримати значення за ключем, який не існує в словнику.
     + IndexError - намагаєтеся звернутися до неіснуючого індексу в списку чи кортежі.
     + TypeError - викликаєте функцію з аргументом невірного типу.
     + NameError - намагаєтеся використовувати змінну, яка не була раніше оголошена.
     + KeyboardInterrupt - прериваєте виконання програми натисканням клавіші Ctrl+C.

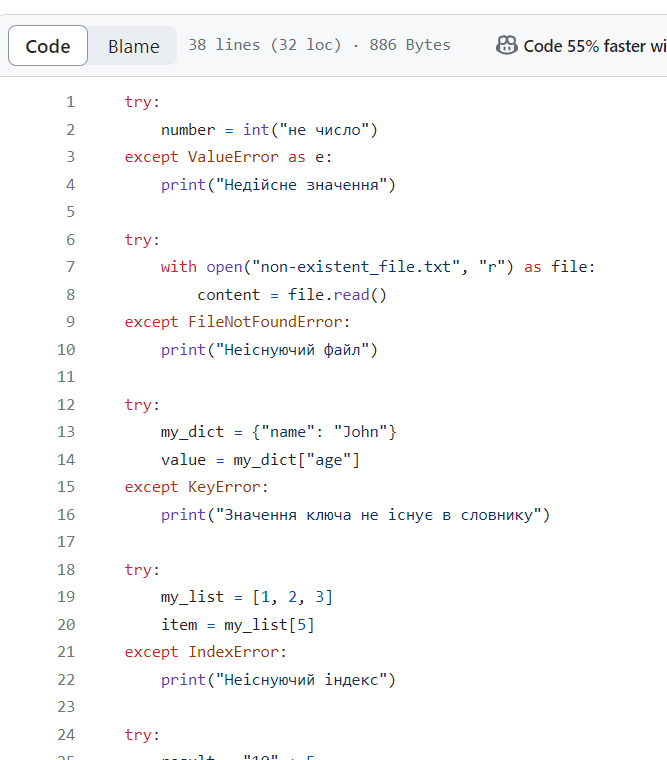
Текст програми:

|  |
| --- |
| try:  number = int("не число")  except ValueError as e:  print("Недійсне значення")  try:  with open("non-existent\_file.txt", "r") as file:  content = file.read()  except FileNotFoundError:  print("Неіснуючий файл")  try:  my\_dict = {"name": "John"}  value = my\_dict["age"]  except KeyError:  print("Значення ключа не існує в словнику")  try:  my\_list = [1, 2, 3]  item = my\_list[5]  except IndexError:  print("Неіснуючий індекс")  try:  result = "10" + 5  except TypeError:  print("Аргумент невірного типу")  try:  print(undefined\_variable)  except NameError as e:  print("Неіснюча змінна")  try:  while True:  pass  except KeyboardInterrupt:  print("Програму перервано користувачем") |

Посилання на github:

[TP-KB-222-Anna-Dryzhak/topic\_04/task3.py at main · aanutadrzhk/TP-KB-222-Anna-Dryzhak (github.com)](https://github.com/aanutadrzhk/TP-KB-222-Anna-Dryzhak/blob/main/topic_04/task3.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 ["rock", "scissor", "paper"]. Наступним кроком, використовуючи модуль random, програма у випадковому порядку вибирає одне із значень ["rock", "scissor", "paper"]. В залежності від умови, що камінь перемагає ножиці, ножиці перемагають папір, а папір перемагає камінь визначити переможця.

Хід виконання завдання:

* 1. import random: додає модуль random, який допоможе генерувати випадкові вибори для комп'ютера.
  2. get\_user\_choice(): ця функція запитує користувача вибрати "камінь (r), ножиці (s), папір (p)" і повертає вибір користувача.
  3. get\_computer\_choice(): ця функція вибирає випадковий вибір для комп'ютера зі списку ['r', 's', 'p'] і повертає його.
  4. winner(user\_choice, computer\_choice): ця функція приймає вибір користувача та вибір комп'ютера, порівнює їх і визначає переможця гри. Якщо вибори однакові, повертає "Нічия!".
  5. main(): основна функція гри. У циклі while гравець вибирає свій варіант, а потім відбувається порівняння з вибором комп'ютера. Результат гри виводиться на екран, і гравця запитується чи бажає він грати ще раз.

Текст програми:

|  |
| --- |
| import random  def get\_user\_choice():  user\_choice = input("Камінь (r), ножиці (s), папір (p). Оберіть:")  return user\_choice  def get\_computer\_choice():  options = ['r', 's', 'p']  computer\_choice = random.choice(options)  return computer\_choice  def winner(user\_choice, computer\_choice):  if user\_choice == computer\_choice:  return "Нічия!"  elif (user\_choice == 'r' and computer\_choice == 's') or (user\_choice == 's' and computer\_choice == 'p') or (user\_choice == 'p' and computer\_choice == 'r'):  return "Ви перемогли!"  else:  return "Комп'ютер переміг."  def main():  while True:  user\_choice = get\_user\_choice()  if user\_choice not in ['r', 's', 'p']:  print("Введено некоректний варіант. Спробуйте ще раз.")  continue  computer\_choice = get\_computer\_choice()  print(f"Ваш вибір: {user\_choice}")  print(f"Вибір комп'ютера: {computer\_choice}")  result = winner(user\_choice, computer\_choice)  print(result)  play\_again = input("Зіграти ще раз? (y/n): ")  if play\_again.lower() != 'y':  break  if \_\_name\_\_ == "\_\_main\_\_":  main() |

Посилання на github:

[TP-KB-222-Anna-Dryzhak/topic\_05/task1.py at main · aanutadrzhk/TP-KB-222-Anna-Dryzhak (github.com)](https://github.com/aanutadrzhk/TP-KB-222-Anna-Dryzhak/blob/main/topic_05/task1.py)

Знімок екрану з посилання на github:



**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.

Хід виконання завдання:

* 1. import requests: імпортуємо бібліотеку requests, яка дозволить виконувати запити до API НБУ для отримання курсів валют.
  2. get\_exchange\_rate(currency\_code): ця функція призначена для отримання курсу обміну для вказаної валюти. currency\_code - аргумент функції, який вказує на код валюти, для якої ми хочемо отримати курс обміну. Формуємо URL-адресу запиту до API НБУ, додавши currency\_code до URL. Використовуємо рядок для побудови URL, де {currency\_code} замінюється на фактичний код валюти. Використовуємо requests.get(), щоб отримати дані з API НБУ. Потім перетворюємо отриману відповідь у форматі JSON в об'єкт Python і повертаємо курс обміну для валюти, якщо він доступний. Якщо немає даних або якщо виникла помилка, функція повертає None.
  3. convert\_currency(amount, from\_currency): ця функція конвертує введену користувачем суму з валюти from\_currency в українські гривні (UAH). Вона використовує функцію get\_exchange\_rate для отримання курсу обміну валюти from\_currency. Якщо курс обміну доступний, вона обчислює конвертовану суму і повертає її. Якщо курс обміну недоступний, функція повертає None.
  4. main(): запитує користувача про суму та валюту, з якої потрібно конвертувати, а потім викликає функцію convert\_currency для обчислення конвертованої суми. Результат виводиться на екран. Якщо не вдалося отримати курси валют, виводиться відповідне повідомлення.

Текст програми:

|  |
| --- |
| import requests  def get\_exchange\_rate(currency\_code):  response = requests.get("https://bank.gov.ua/NBUStatService/v1/statdirectory/exchange?valcode={currency\_code}&json")  data = response.json()  return data[0]['rate'] if data else None  def convert\_currency(amount, from\_currency):  from\_rate = get\_exchange\_rate(from\_currency)  if from\_rate is None:  return None  converted\_amount = amount \* from\_rate  return converted\_amount  def main():  print("Конвертер валют: EUR, USD, PLN в UAH")  amount = float(input("Введіть суму: "))  from\_currency = input("Введіть валюту (EUR, USD, PLN): ").upper()  result = convert\_currency(amount, from\_currency)  if result is not None:  print(f"{amount} {from\_currency} = {result} UAH")  else:  print("Не вдалося отримати курси валют. Спробуйте пізніше.")  if \_\_name\_\_ == "\_\_main\_\_":  main() |

Посилання на github:

[TP-KB-222-Anna-Dryzhak/topic\_05/task2.py at main · aanutadrzhk/TP-KB-222-Anna-Dryzhak (github.com)](https://github.com/aanutadrzhk/TP-KB-222-Anna-Dryzhak/blob/main/topic_05/task2.py)

Знімок екрану з посилання на github:



**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.

Хід виконання завдання:

* 1. Створила три файли: functions.py, operations.py, і task3.py.
  2. Перенесла функції додавання, віднімання, множення та ділення до файлу functions.py.
  3. Перенесла функції для операцій та введення даних до файлу operations.py та підключила сюди файл functions.py.
  4. Іншу частину коду програми “Калькулятор” залишила в файлі task3.py та підключила файл operations.py.

Текст програми:

Файл functions.py:

|  |
| --- |
| def addition(a,b):  return a + b  def subtraction(a,b):  return a - b  def multiplication(a,b):  return a \* b  def division(a,b):  try:  return a / b  except ZeroDivisionError:  print("Ділення на нуль неможливе")  return None |

Файл operations.py:

|  |
| --- |
| from functions import addition, subtraction, multiplication, division  def value(promt:str):  while True:  try:  return int(input(promt))  except ValueError:  print("Неправильне значення (введіть число)")    def operation(a, b, op):  match op:  case "+":  print(addition(a,b))  case "-":  print(subtraction(a,b))  case "\*":  print(multiplication(a,b))  case "/":  print(division(a,b))  case \_:  print("Неправильно введена операція, спробуйте \*,/,-,+") |

Файл task3.py:

|  |
| --- |
| from operations import value, operation  while True:  print("Додавання: + ")  print("Віднімання: - ")  print("Множення: \* ")  print("Ділення: / ")  print("Вихід: 0 ")  a = value("Введіть значення a: ")  b = value("Введіть значення b: ")  op = (input("Операція, яку треба виконати: "))  if op == "0":  break  result = operation(a, b, op)  if result is not None:  print(result) |

Посилання на github:

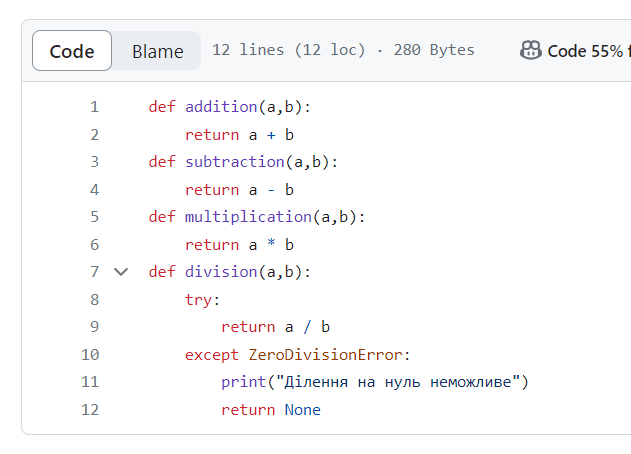
[TP-KB-222-Anna-Dryzhak/topic\_05/task3.py at main · aanutadrzhk/TP-KB-222-Anna-Dryzhak (github.com)](https://github.com/aanutadrzhk/TP-KB-222-Anna-Dryzhak/blob/main/topic_05/task3.py)

[TP-KB-222-Anna-Dryzhak/topic\_05/functions.py at main · aanutadrzhk/TP-KB-222-Anna-Dryzhak (github.com)](https://github.com/aanutadrzhk/TP-KB-222-Anna-Dryzhak/blob/main/topic_05/functions.py)

[TP-KB-222-Anna-Dryzhak/topic\_05/operations.py at main · aanutadrzhk/TP-KB-222-Anna-Dryzhak (github.com)](https://github.com/aanutadrzhk/TP-KB-222-Anna-Dryzhak/blob/main/topic_05/operations.py)

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:

**Логування всіх дій в застосунку 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:

* 1. За основу беремо код з завдання 3 теми №5.
  2. У файл calc.py та operation.py імпортуємо бібліотеку logging та у файл calc.py бібліотеку os.
  3. Файл calc.py:
     + За допомогою os.path.dirname(...) ми отримуємо поточну директорію файла калькулятору, а потім використовуємо os.path.join() для створення повного шляху до файлу логів у цій же директорії.
     + За допомогою logging.basicConfig() встановлюємо параметри для системи логування, а саме filename='calculator.log' - вказує ім'я файлу, в який будуть записуватися логи. level=logging.INFO - будуть записуватися повідомлення з рівнем INFO і вище. format='%(asctime)s - %(message)s' - формат записів у файлі логу, %(asctime)s буде замінено на дату та час запису, і %(message)s буде замінено на текстове повідомлення логу. datefmt='%Y-%m-%d %H:%M:%S' - формат для дати та часу, які включаються у лог.
  4. Файл operation.py:
     + logging.info() - запис інформації рівня "INFO" у файл логів.
     + "entered: a={a}, b={b}, operation={op}, result={result}" - повідомлення, яке буде записане у файл логів.

Текст програми:

*Файл calc.py:*

|  |
| --- |
| import logging  import os  from operations import value, operation  current\_directory = os.path.dirname(os.path.abspath(\_\_file\_\_))  log\_file\_path = os.path.join(current\_directory, 'calculator.log')  logging.basicConfig(filename=log\_file\_path, level=logging.INFO, format='%(asctime)s - %(message)s', datefmt='%Y-%m-%d %H:%M:%S')  while True:  print("Додавання: + ")  print("Віднімання: - ")  print("Множення: \* ")  print("Ділення: / ")  print("Вихід: 0 ")  a = value("Введіть значення a: ")  b = value("Введіть значення b: ")  op = (input("Операція, яку треба виконати: "))  if op == "0":  break  result = operation(a, b, op)  if result is not None:  print(result)  logging.shutdown() |

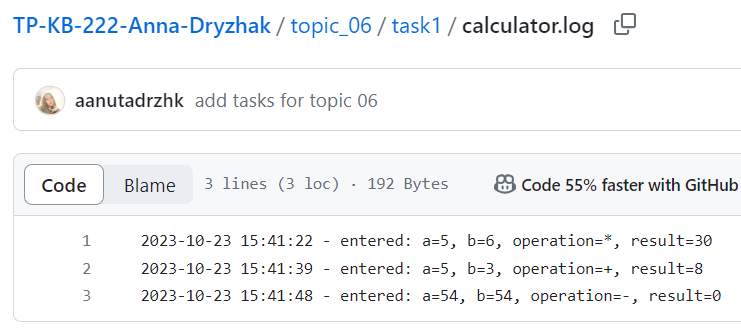
*Файл operation.py:*

|  |
| --- |
| import logging  from functions import addition, subtraction, multiplication, division  def value(promt:str):  while True:  try:  return int(input(promt))  except ValueError:  print("Неправильне значення (введіть число)")    def operation(a, b, op):  result = None  if op == "+":  result = addition(a, b)  elif op == "-":  result = subtraction(a, b)  elif op == "\*":  result = multiplication(a, b)  elif op == "/":  result = division(a, b)  if result is not None:  logging.info(f"entered: a={a}, b={b}, operation={op}, result={result}")  return result |

Посилання на github:

[TP-KB-222-Anna-Dryzhak/topic\_06/task1 at main · aanutadrzhk/TP-KB-222-Anna-Dryzhak (github.com)](https://github.com/aanutadrzhk/TP-KB-222-Anna-Dryzhak/tree/main/topic_06/task1)

Знімок екрану з посилання на github:



**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:

* 1. students - це список, що містить словники з іменами студентів і їх оцінками.
  2. reverse\_sort - користувачу пропонується ввести "так" або "ні" для визначення, чи потрібно сортувати від меншого до більшого.
  3. Reverse - якщо користувач вибрав сортувати від більшого до меншого, змінна reverse встановлюється на True,.
  4. sorted\_by\_name - застосовується функція sorted() для сортування списку students за полем "name" кожного словника. Використовується аргумент key=lambda x: x['name'] для вказання, яке поле використовувати для сортування. За допомогою reverse визначається напрям сортування.
  5. Кожен елемент відсортованого списку виводиться в форматі "Name: {ім'я}, Grade: {оцінка}" за допомогою циклу for.
  6. Аналогічно реалізовано сортування за оцінкою.

Текст програми:

|  |
| --- |
| students = [{'name': 'Ann', 'grade': 92},  {'name': 'Pavlo', 'grade': 85},  {'name': 'Mark', 'grade': 95},  {'name': 'Maria', 'grade': 78}]  reverse\_sort = input("Сортувати від меншого до більшого? (так/ні): ").lower()  reverse = False  if reverse\_sort == 'ні':  reverse = True  sorted\_by\_name = sorted(students, key=lambda x: x['name'], reverse=reverse)  print("Сортування за ім'ям:")  for student in sorted\_by\_name:  print(f"Name: {student['name']}, Grade: {student['grade']}")  sorted\_by\_grade = sorted(students, key=lambda x: x['grade'], reverse=reverse)  print("Сортування за оцінкою:")  for student in sorted\_by\_grade:  print(f"Name: {student['name']}, Grade: {student['grade']}") |

Посилання на github:

[TP-KB-222-Anna-Dryzhak/topic\_06/task2.py at main · aanutadrzhk/TP-KB-222-Anna-Dryzhak (github.com)](https://github.com/aanutadrzhk/TP-KB-222-Anna-Dryzhak/blob/main/topic_06/task2.py)

Знімок екрану з посилання на github:

