**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:

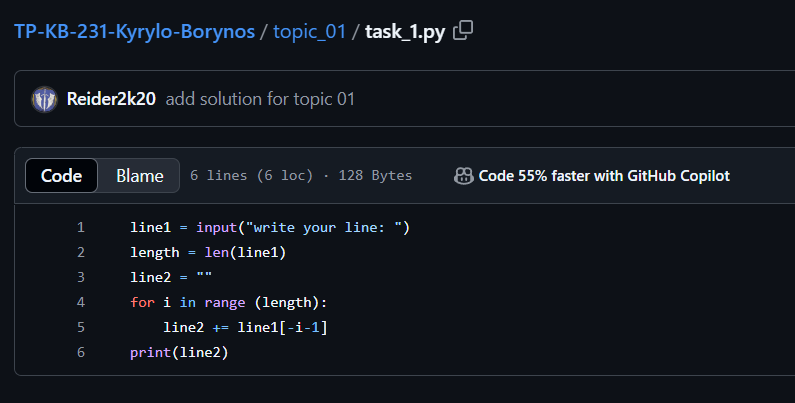
Спочатку користувач вводить свій рядок, після чого в змінну length записується його довжина та створюється пустий рядок, в який будуть переписуватись символи, далі йде цикл, який інкрементує змінну поки вона не стане рівною довжині рядка, на кожне значення цієї змінної, в другий рядок додається символ з індексом “-i-1”, -1 тут тому що для індексації елементів не з початку, а з кінця використовуються від’ємні значення, а – перед “i” для того, щоб значення зменшувалось, а не збільшувалось

Текст програми:

|  |
| --- |
| line1 = input("write your line: ")  length = len(line1)  line2 = ""  for i in range (length):  line2 += line1[-i-1]  print(line2) |

Посилання на github: https://github.com/Reider2k20/TP-KB-231-Kyrylo-Borynos/blob/main/topic\_01/task\_1.py

Знімок екрану з посилання на github:



**Тестування функцій, що працюють з рядками**

Необхідно написати програму, який буде демонструвати роботу наступних функцій: strip(), capitalize(), title(), upper(), lower().

Хід виконання завдання:

Програма демонструє можливості різних функцій, які працюють з рядками.

Текст програми:

|  |
| --- |
| string1 = " hello, worlD "  print("Функція strip() прибирає пробіли на початку та в кінці рядка:", string1.strip())  print("Функція capitalize() переводить перший символ рядка в верхній регістр:", string1.strip().capitalize()) #для демонстрації пробіли на початку було прибрано  print("Функція title() переводить першу літеру кожного слова в верхній регістр:", string1.title())  print("Функція upper() переводить всі літери рядка в верхній регістр:", string1.upper())  print("Функція lower() переводить всі літери рядка в нижній регістр:", string1.lower()) |

Посилання на github: https://github.com/Reider2k20/TP-KB-231-Kyrylo-Borynos/blob/main/topic\_01/task\_2.py

Знімок екрану з посилання на github:



**Пошук дискримінанту квадратного рівняння**

Необхідно маючи 3 значення з квадратного рівняння обчислити дискримінант.

Хід виконання завдання:

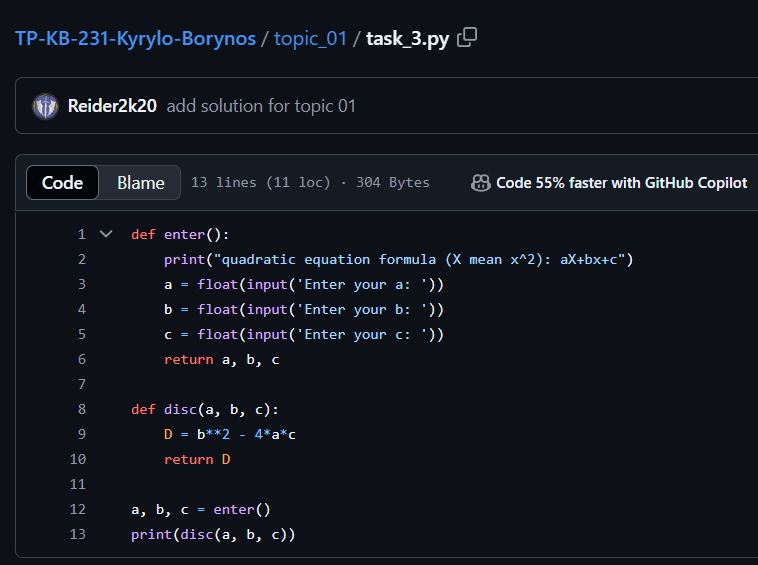
Користувач вводить три значення, які записуються в відповідні змінні, після чого ці змінні підставляються у формулу для обрахунку та результат виводиться в консоль.

Текст програми:

|  |
| --- |
| def enter():  print("quadratic equation formula (X mean x^2): aX+bx+c")  a = float(input('Enter your a: '))  b = float(input('Enter your b: '))  c = float(input('Enter your c: '))  return a, b, c  def disc(a, b, c):  D = b\*\*2 - 4\*a\*c  return D  a, b, c = enter()  print(disc(a, b, c)) |

Посилання на github: https://github.com/Reider2k20/TP-KB-231-Kyrylo-Borynos/blob/main/topic\_01/task\_3.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 маючи 3 значення з квадратного рівняння обчислити значення коренів цього рівняння, якщо вони є.

Хід виконання завдання:

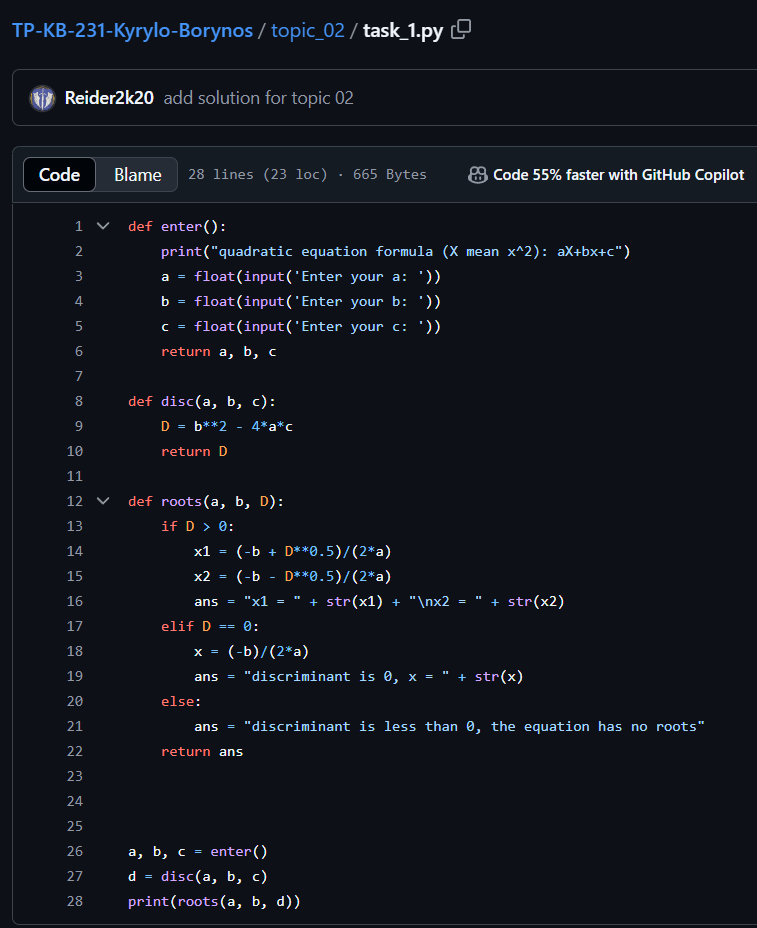
Розрахунок дискримінанту відбувається, як в 3 завданні Теми 1, після чого значення дискримінанту, “a” та “b” приймаються функцією для розрахунку коренів, якщо значення дискримінанту більше 0, то буде 2 кореня, рівне 0, 1 корінь, а якщо менше 0, то коренів не буде.

Текст програми:

|  |
| --- |
| def enter():  print("quadratic equation formula (X mean x^2): aX+bx+c")  a = float(input('Enter your a: '))  b = float(input('Enter your b: '))  c = float(input('Enter your c: '))  return a, b, c  def disc(a, b, c):  D = b\*\*2 - 4\*a\*c  return D  def roots(a, b, D):  if D > 0:  x1 = (-b + D\*\*0.5)/(2\*a)  x2 = (-b - D\*\*0.5)/(2\*a)  ans = "x1 = " + str(x1) + "\nx2 = " + str(x2)  elif D == 0:  x = (-b)/(2\*a)  ans = "discriminant is 0, x = " + str(x)  else:  ans = "discriminant is less than 0, the equation has no roots"  return ans  a, b, c = enter()  d = disc(a, b, c)  print(roots(a, b, d)) |

Посилання на github: https://github.com/Reider2k20/TP-KB-231-Kyrylo-Borynos/blob/main/topic\_02/task\_1.py

Знімок екрану з посилання на github:



**Програма калькулятор на основі if else**

Необхідно написати програму калькулятор для виконання операцій над двома введеними користувачем числами.

Хід виконання завдання:

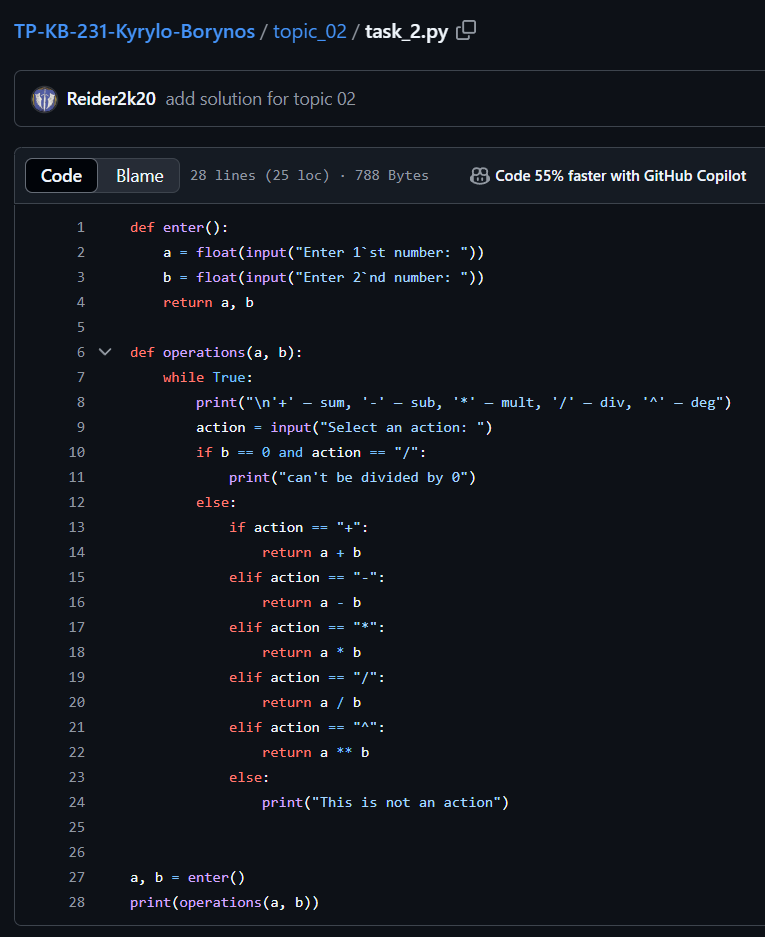
Програма приймає значення для 2 змінних від користувача після чого виконується нескінченний цикл допоки користувач не обере дію, яку можливо виконати та виводить результат цієї дії.

Текст програми:

|  |
| --- |
| def enter():  a = float(input("Enter 1`st number: "))  b = float(input("Enter 2`nd number: "))  return a, b  def operations(a, b):  while True:  print("\n'+' — sum, '-' — sub, '\*' — mult, '/' — div, '^' — deg")  action = input("Select an action: ")  if b == 0 and action == "/":  print("can't be divided by 0")  else:  if action == "+":  return a + b  elif action == "-":  return a - b  elif action == "\*":  return a \* b  elif action == "/":  return a / b  elif action == "^":  return a \*\* b  else:  print("This is not an action")  a, b = enter()  print(operations(a, b)) |

Посилання на github: https://github.com/Reider2k20/TP-KB-231-Kyrylo-Borynos/blob/main/topic\_02/task\_2.py

Знімок екрану з посилання на github:



**Програма калькулятор на основі match**

Необхідно написати програму ідентичну програмі з 2 завдання цієї теми.

Хід виконання завдання:

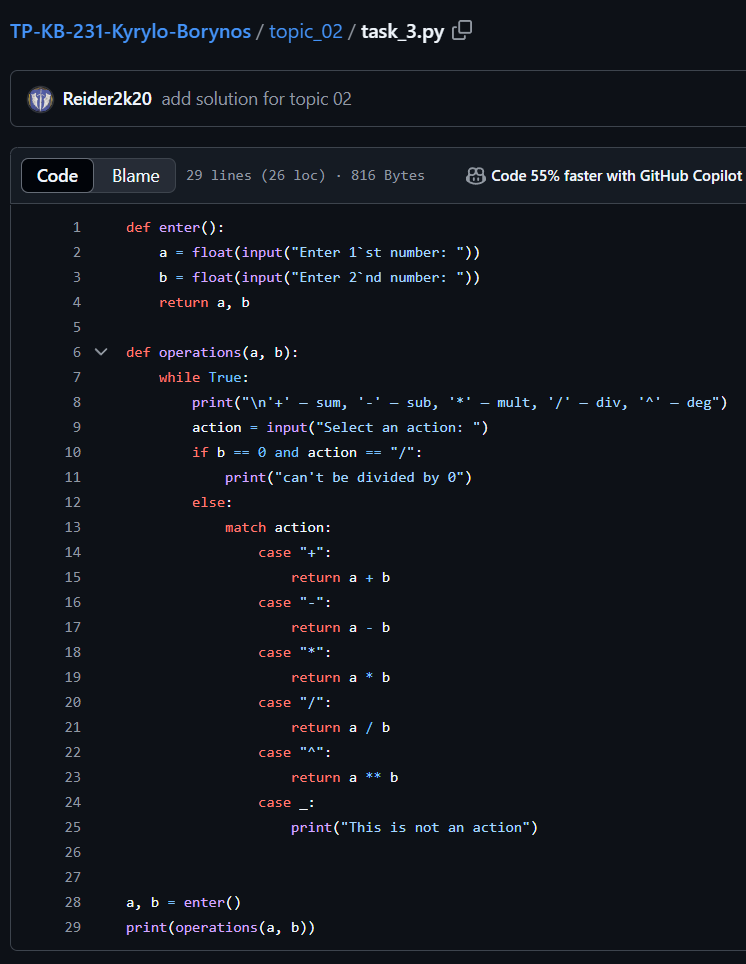
Ця програма ідентична програмі з 2 завдання Теми 2, але замість if else конструкції використовувати match конструкцію.

Текст програми:

|  |
| --- |
| def enter():  a = float(input("Enter 1`st number: "))  b = float(input("Enter 2`nd number: "))  return a, b  def operations(a, b):  while True:  print("\n'+' — sum, '-' — sub, '\*' — mult, '/' — div, '^' — deg")  action = input("Select an action: ")  if b == 0 and action == "/":  print("can't be divided by 0")  else:  match action:  case "+":  return a + b  case "-":  return a - b  case "\*":  return a \* b  case "/":  return a / b  case "^":  return a \*\* b  case \_:  print("This is not an action")  a, b = enter()  print(operations(a, b)) |

Посилання на github: https://github.com/Reider2k20/TP-KB-231-Kyrylo-Borynos/blob/main/topic\_02/task\_3.py

Знімок екрану з посилання на github:



Звіт до Теми №3

Умовний перехід

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

**Програма калькулятор з постійним запитом нових даних**

Необхідно написати програму калькулятор з постійним запитом нових ввідних даних.

Хід виконання завдання:

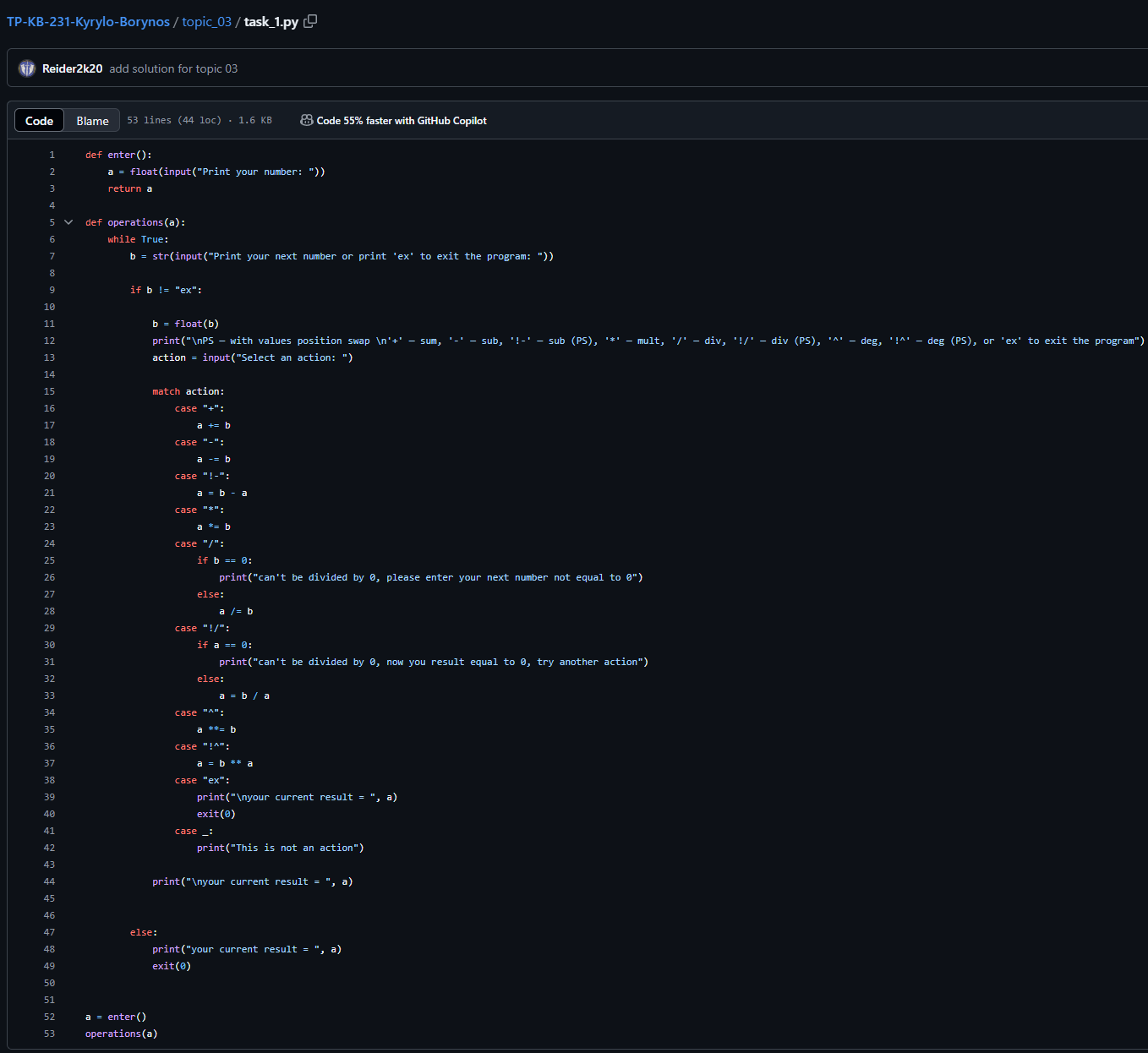
Програма запитує перше число, після чого друге число вводиться всередині циклу, на будь-якому етапі виконання програми можна ввести “ex” та завершити виконання програми, при введені дії можна поставити ! перед нею і при її виконанні значення будуть поміняні місцями, після виконання дії, результат записується в першу змінну, потім цикл знову питає значення для 2 числа і так допоки користувач сам не вийде із циклу.

Текст програми:

|  |
| --- |
| def enter():  a = float(input("Print your number: "))  return a  def operations(a):  while True:  b = str(input("Print your next number or print 'ex' to exit the program: "))  if b != "ex":  b = float(b)  print("\nPS — with values position swap \n'+' — sum, '-' — sub, '!-' — sub (PS), '\*' — mult, '/' — div, '!/' — div (PS), '^' — deg, '!^' — deg (PS), or 'ex' to exit the program")  action = input("Select an action: ")  match action:  case "+":  a += b  case "-":  a -= b  case "!-":  a = b - a  case "\*":  a \*= b  case "/":  if b == 0:  print("can't be divided by 0, please enter your next number not equal to 0")  else:  a /= b  case "!/":  if a == 0:  print("can't be divided by 0, now you result equal to 0, try another action")  else:  a = b / a  case "^":  a \*\*= b  case "!^":  a = b \*\* a  case "ex":  print("\nyour current result = ", a)  exit(0)  case \_:  print("This is not an action")  print("\nyour current result = ", a)  else:  print("your current result = ", a)  exit(0)  a = enter()  operations(a) |

Посилання на github: https://github.com/Reider2k20/TP-KB-231-Kyrylo-Borynos/blob/main/topic\_03/task\_1.py

Знімок екрану з посилання на github:



**Тестування функцій списків**

Необхідно протестувати наступні функції списків: extend(), append(), insert(id, val), remove(val), clear(), sort(), reverse(), copy().

Хід виконання завдання:

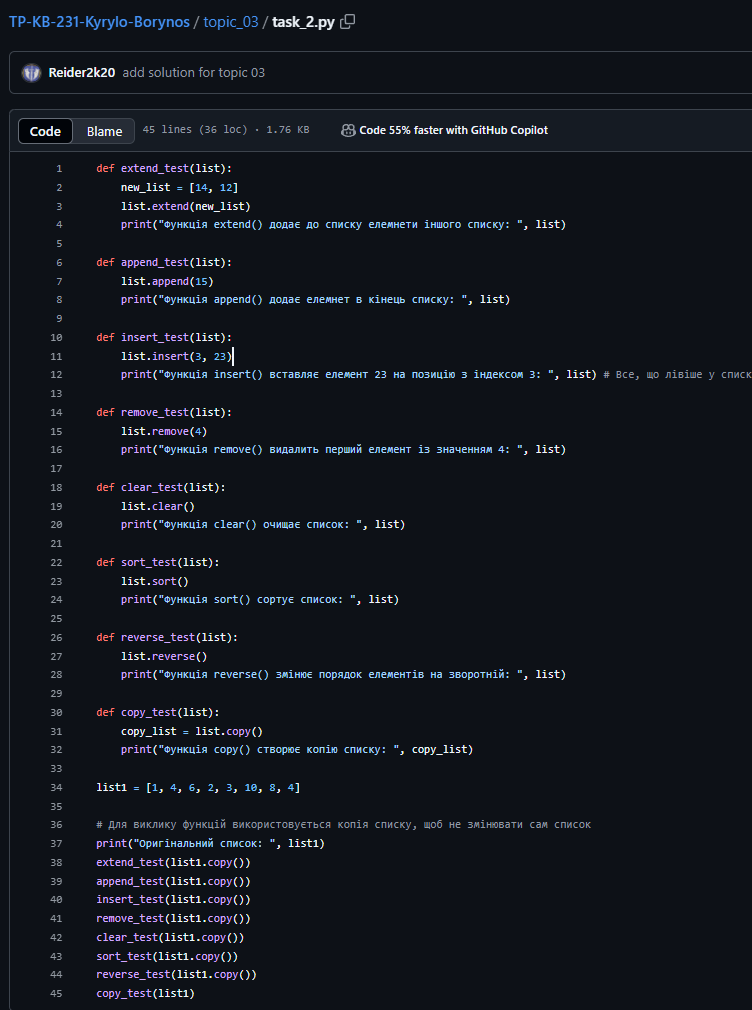
Програма просто демонструє можливості функцій списків.

Текст програми:

|  |
| --- |
| def extend\_test(list):  new\_list = [14, 12]  list.extend(new\_list)  print("Функція extend() додає до списку елемнети іншого списку: ", list)  def append\_test(list):  list.append(15)  print("Функція append() додає елемнет в кінець списку: ", list)  def insert\_test(list):  list.insert(3, 23)  print("Функція insert() вставляє елемент 23 на позицію з індексом 3: ", list) # Все, що лівіше у списку (0-2) не змінить свій індекс, а все, що правіше, включаючи 3 (3-...) збільшить його на 1  def remove\_test(list):  list.remove(4)  print("Функція remove() видалить перший елемент із значенням 4: ", list)  def clear\_test(list):  list.clear()  print("Функція clear() очищає список: ", list)  def sort\_test(list):  list.sort()  print("Функція sort() сортує список: ", list)  def reverse\_test(list):  list.reverse()  print("Функція reverse() змінює порядок елементів на зворотній: ", list)  def copy\_test(list):  copy\_list = list.copy()  print("Функція copy() створює копію списку: ", copy\_list)  list1 = [1, 4, 6, 2, 3, 10, 8, 4]  # Для виклику функцій використовується копія списку, щоб не змінювати сам список  print("Оригінальний список: ", list1)  extend\_test(list1.copy())  append\_test(list1.copy())  insert\_test(list1.copy())  remove\_test(list1.copy())  clear\_test(list1.copy())  sort\_test(list1.copy())  reverse\_test(list1.copy())  copy\_test(list1) |

Посилання на github: https://github.com/Reider2k20/TP-KB-231-Kyrylo-Borynos/blob/main/topic\_03/task\_2.py

Знімок екрану з посилання на github:



**Тестування функцій словників**

Необхідно виконати тестування наступних функцій словників: update(), del(), clear(), keys(), values(), items().

Хід виконання завдання:

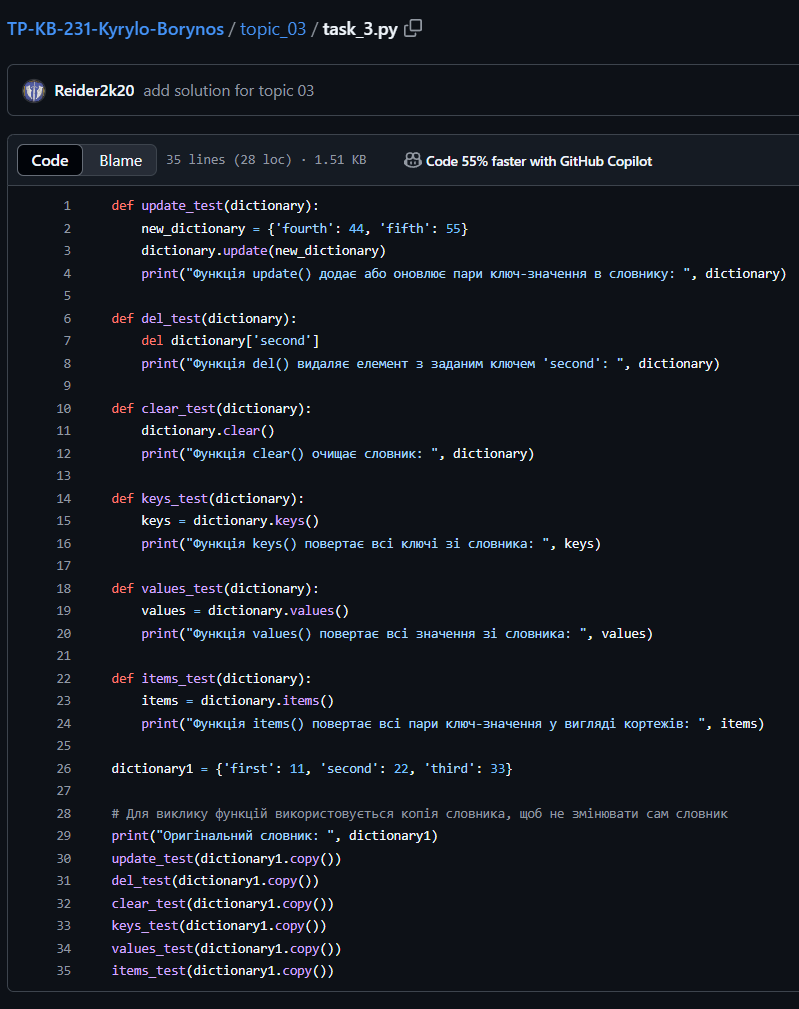
Програма демонструє можливості функцій списків.

Текст програми:

|  |
| --- |
| def update\_test(dictionary):  new\_dictionary = {'fourth': 44, 'fifth': 55}  dictionary.update(new\_dictionary)  print("Функція update() додає або оновлює пари ключ-значення в словнику: ", dictionary)  def del\_test(dictionary):  del dictionary['second']  print("Функція del() видаляє елемент з заданим ключем 'second': ", dictionary)  def clear\_test(dictionary):  dictionary.clear()  print("Функція clear() очищає словник: ", dictionary)  def keys\_test(dictionary):  keys = dictionary.keys()  print("Функція keys() повертає всі ключі зі словника: ", keys)  def values\_test(dictionary):  values = dictionary.values()  print("Функція values() повертає всі значення зі словника: ", values)  def items\_test(dictionary):  items = dictionary.items()  print("Функція items() повертає всі пари ключ-значення у вигляді кортежів: ", items)  dictionary1 = {'first': 11, 'second': 22, 'third': 33}  # Для виклику функцій використовується копія словника, щоб не змінювати сам словник  print("Оригінальний словник: ", dictionary1)  update\_test(dictionary1.copy())  del\_test(dictionary1.copy())  clear\_test(dictionary1.copy())  keys\_test(dictionary1.copy())  values\_test(dictionary1.copy())  items\_test(dictionary1.copy()) |

Посилання на github: https://github.com/Reider2k20/TP-KB-231-Kyrylo-Borynos/blob/main/topic\_03/task\_3.py

Знімок екрану з посилання на github:



**Вставка елемента у відсортований список**

Маючи відсортований список необхідно знайти позицію для вставки нового елемента в нього.

Хід виконання завдання:

Програма виводить відсортований список, після чого користувач вводить нове ім’я для цього списку, потім функція сортування шукає позицію, на яку можна вставити це ім’я і повертає індекс позиції, потім в залежності від цього індекса залежить, що буде виведено користувачу, якщо це перша або остання (останній елемент + 1) позиція, то це і буде виведено, якщо ж ні, то виведеться між якими ім’ями вставиться введене.

Текст програми:

|  |
| --- |
| def sort\_pos(list, item):  insert\_pos = 0  for elem in list:  if item > elem:  insert\_pos += 1  else:  break  return insert\_pos  names\_list = ["Andrew", "Bob", "Emma", "Michael", "Tomas" ]  print(f"List of names: {names\_list}")  name = input("Enter a new name to get a position to insert it: ")  pos = sort\_pos(names\_list, name)  if pos == 0:  print("This name will be inserted first")  elif pos == len(names\_list):  print("This name will be inserted last")  else:  print(f"This name will be inserted between {names\_list[pos - 1]} and {names\_list[pos]}.") |

Посилання на github: https://github.com/Reider2k20/TP-KB-231-Kyrylo-Borynos/blob/main/topic\_03/task\_4.py

Знімок екрану з посилання на github:

