**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:

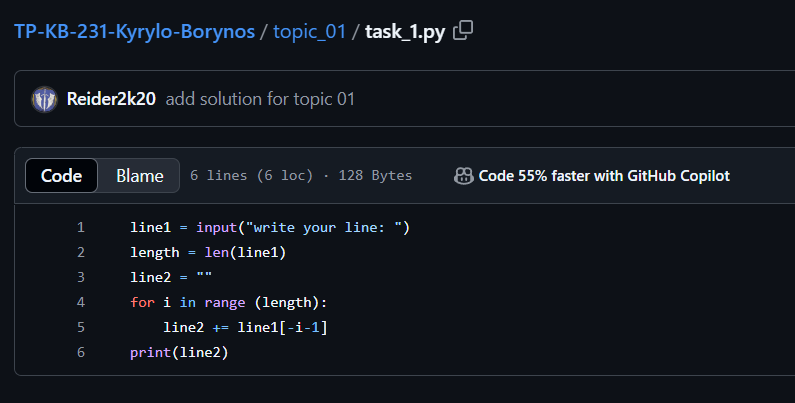
Спочатку користувач вводить свій рядок, після чого в змінну length записується його довжина та створюється пустий рядок, в який будуть переписуватись символи, далі йде цикл, який інкрементує змінну поки вона не стане рівною довжині рядка, на кожне значення цієї змінної, в другий рядок додається символ 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:

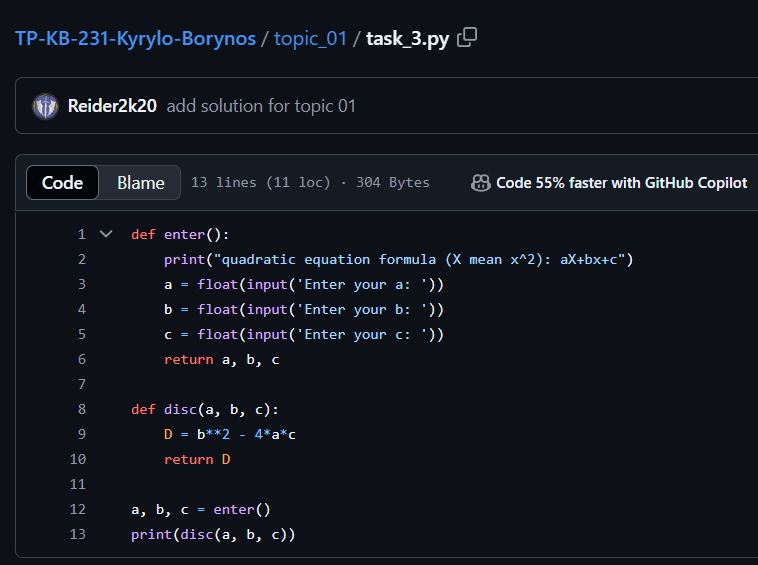
Користувач вводить три значення, які записуються в відповідні змінні, після чого ці змінні підставляються у формулу для обрахунку та результат виводиться в консоль.

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:

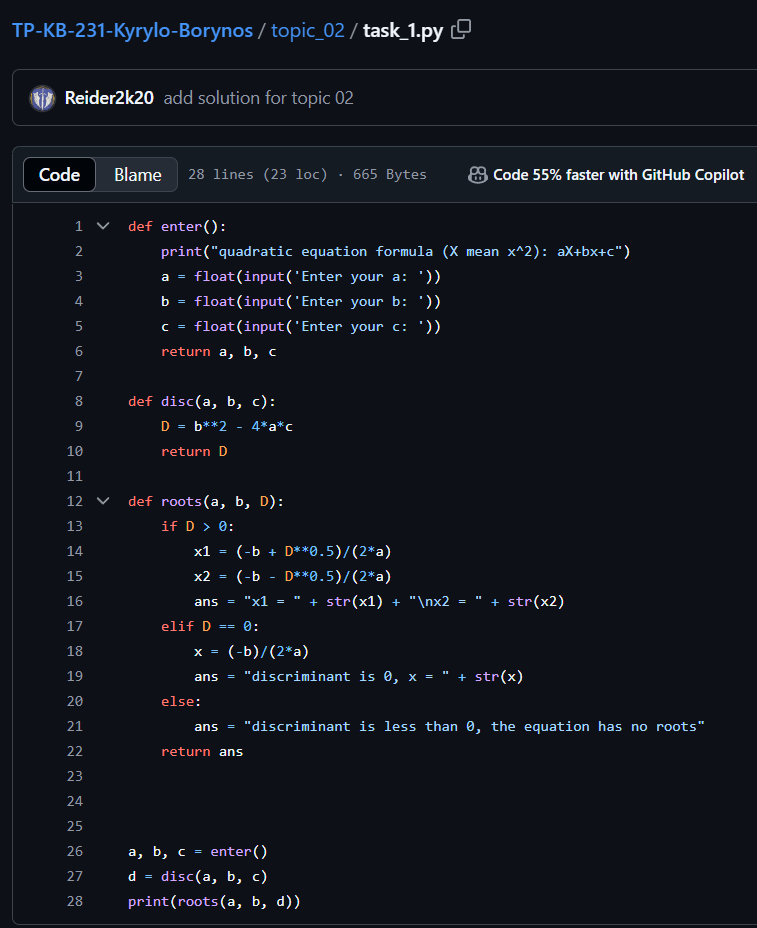
Розрахунок дискримінанту відбувається, як в 3 завданні Теми 1, після чого значення дискримінанту, “a” та “b” приймаються функцією для розрахунку коренів, якщо значення дискримінанту більше 0, то буде 2 кореня, рівне 0, 1 корінь, а якщо менше 0, то коренів не буде.

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:

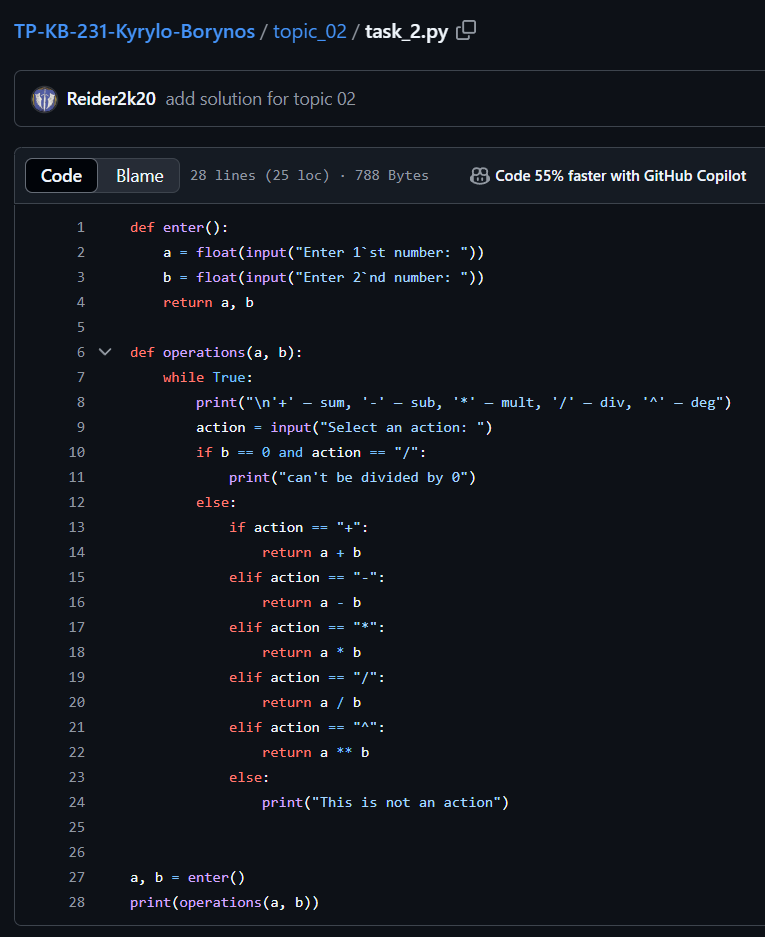
Програма приймає значення для 2 змінних від користувача після чого виконується нескінченний цикл допоки користувач не обере дію, яку можливо виконати та виводить результат цієї дії.

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:

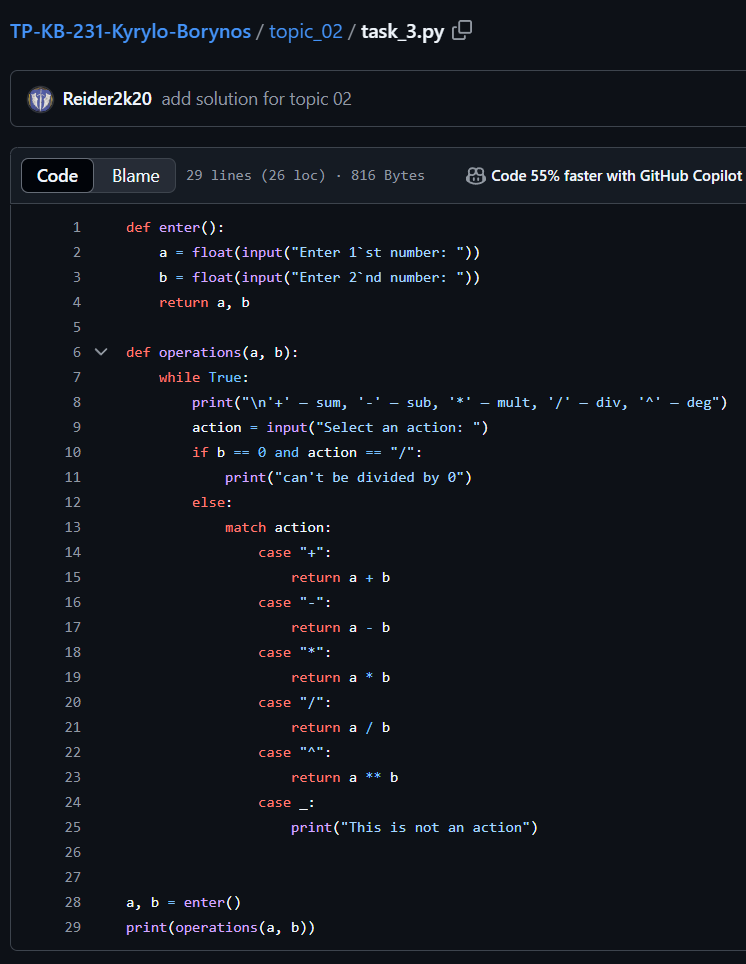
Ця програма ідентична програмі з 2 завдання Теми 2, але замість if else конструкції використовувати match конструкцію.

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

Цикли

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:

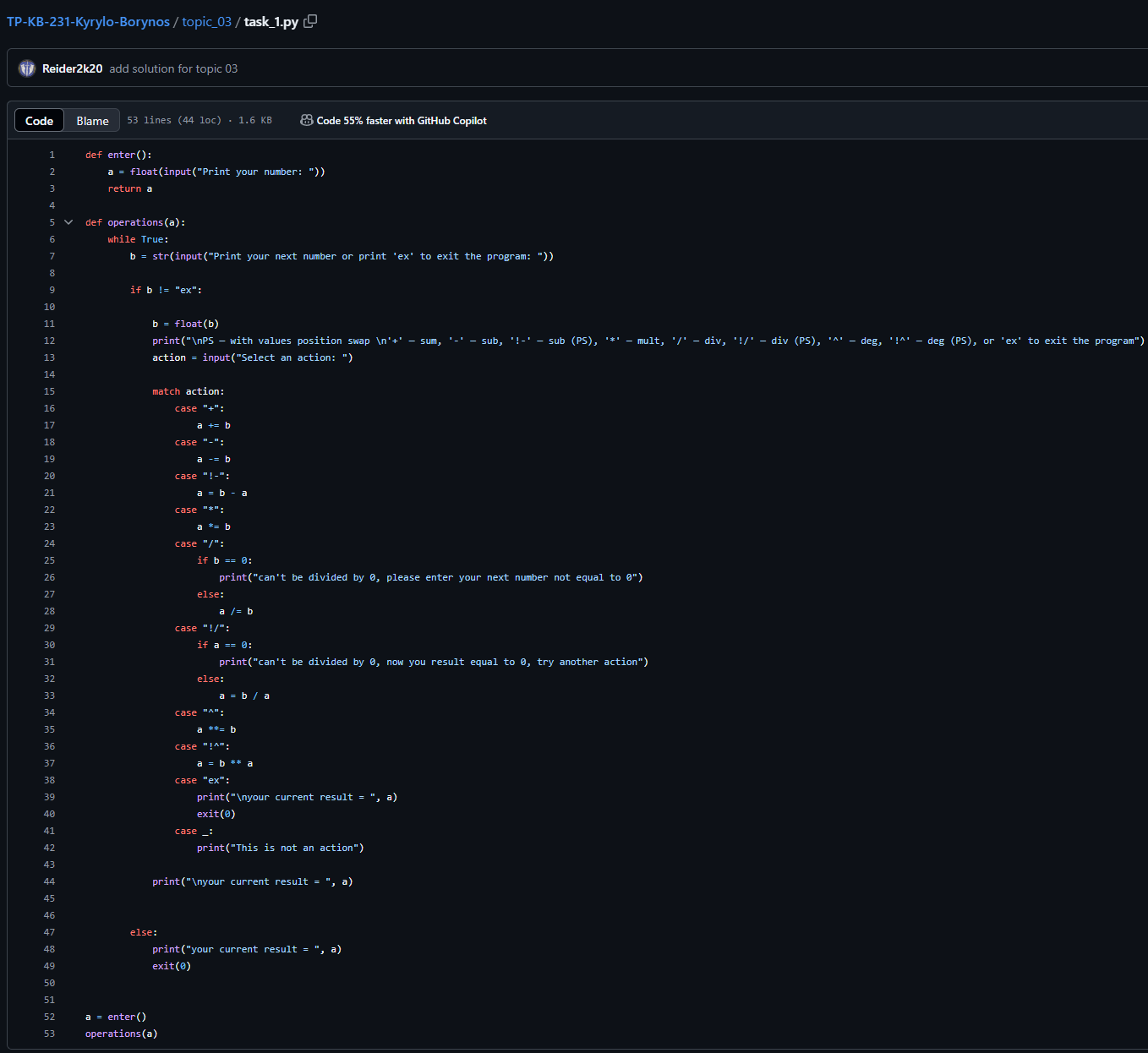
Програма запитує перше число, після чого друге число вводиться всередині циклу, на будь-якому етапі виконання програми можна ввести “ex” та завершити виконання програми, при введені дії можна поставити ! перед нею і при її виконанні значення будуть поміняні місцями, після виконання дії, результат записується 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:

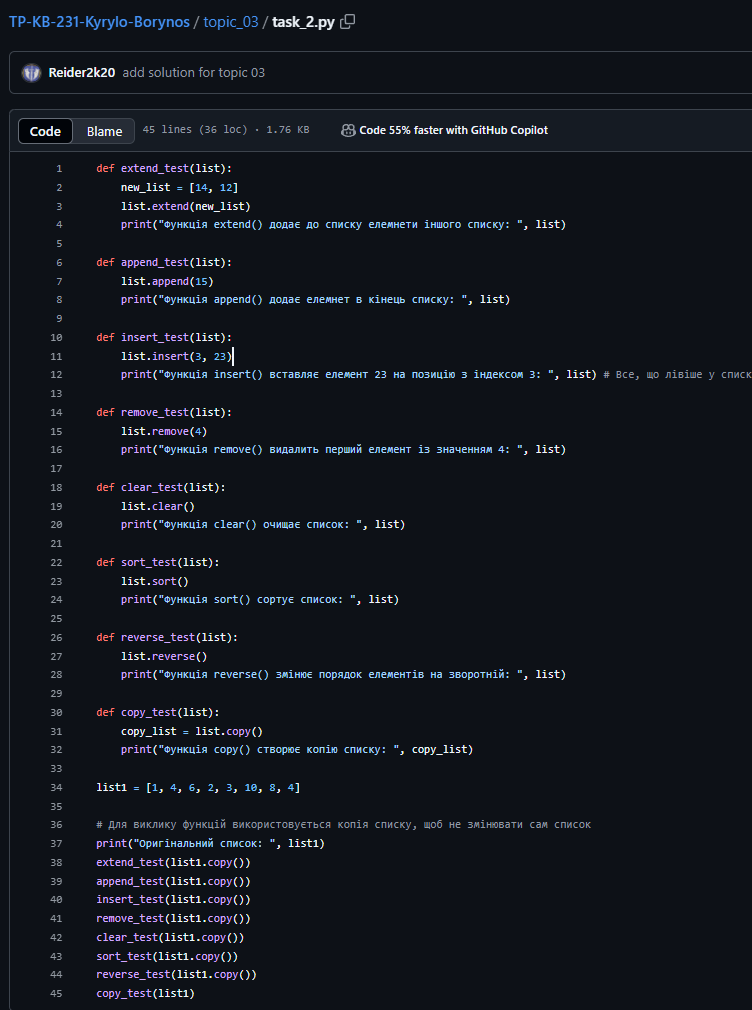
Програма просто демонструє можливості функцій списків.

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:

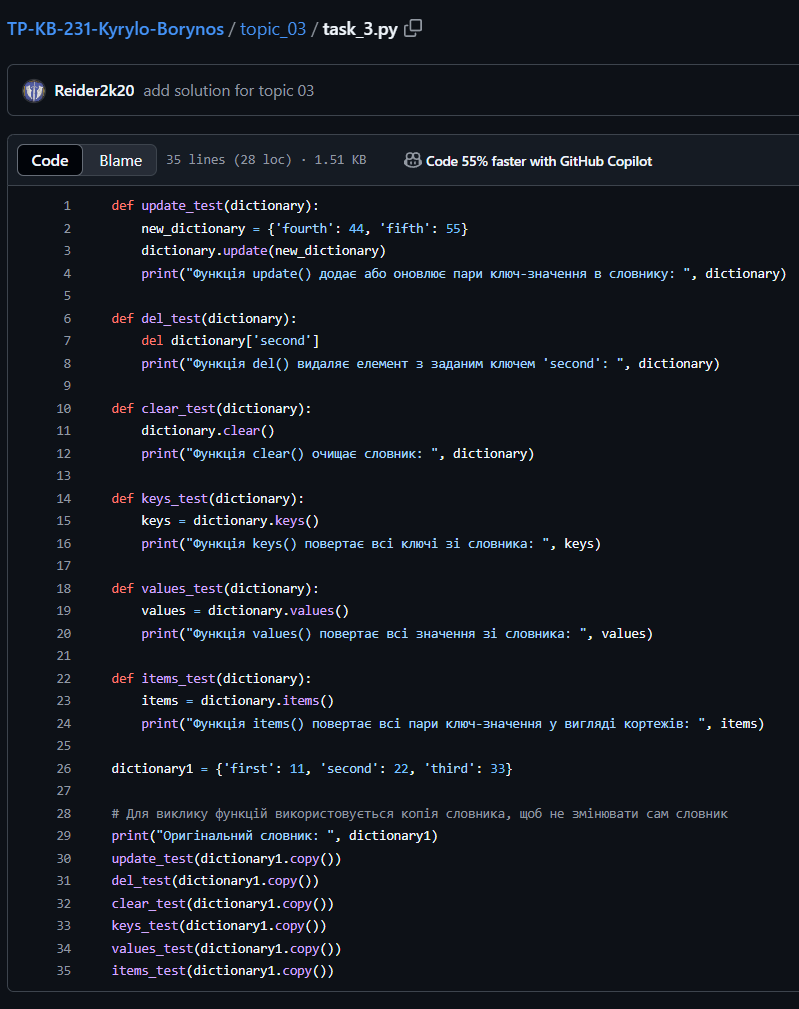
Програма демонструє можливості функцій списків.

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:

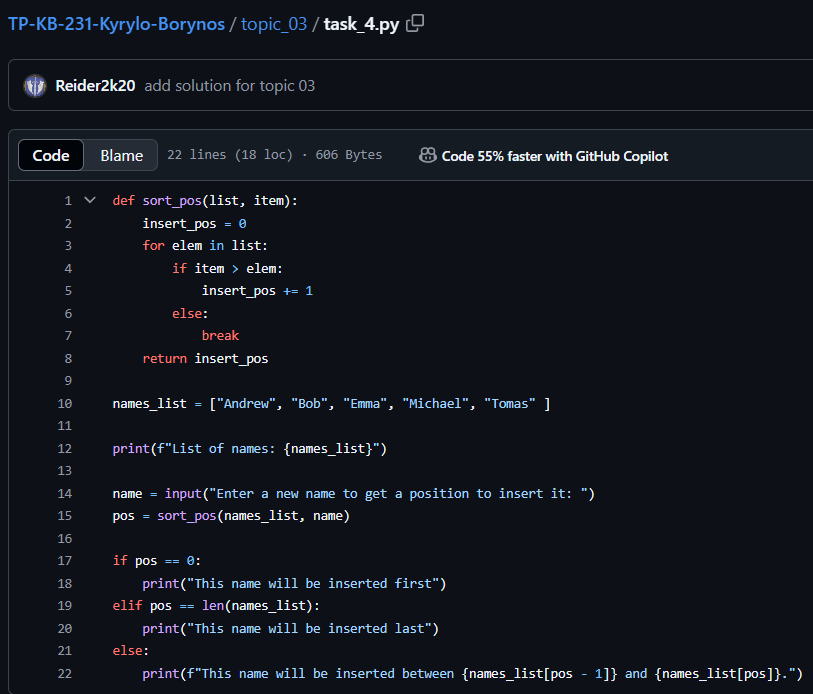
Програма виводить відсортований список, після чого користувач вводить нове ім’я для цього списку, потім функція сортування шукає позицію, на яку можна вставити це ім’я і повертає індекс позиції, потім в залежності від цього індекса залежить, що буде виведено користувачу, якщо це перша або остання (останній 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:



Звіт до Теми №4

Виняткові ситуації

Під час виконання практичного завдання до Теми №4 було надано варіанти рішення до наступних задач:

**Розширення програми калькулятора**

Необхідно розширити програму калькулятор додаванням обробки виняткових ситуацій.

Хід виконання завдання:

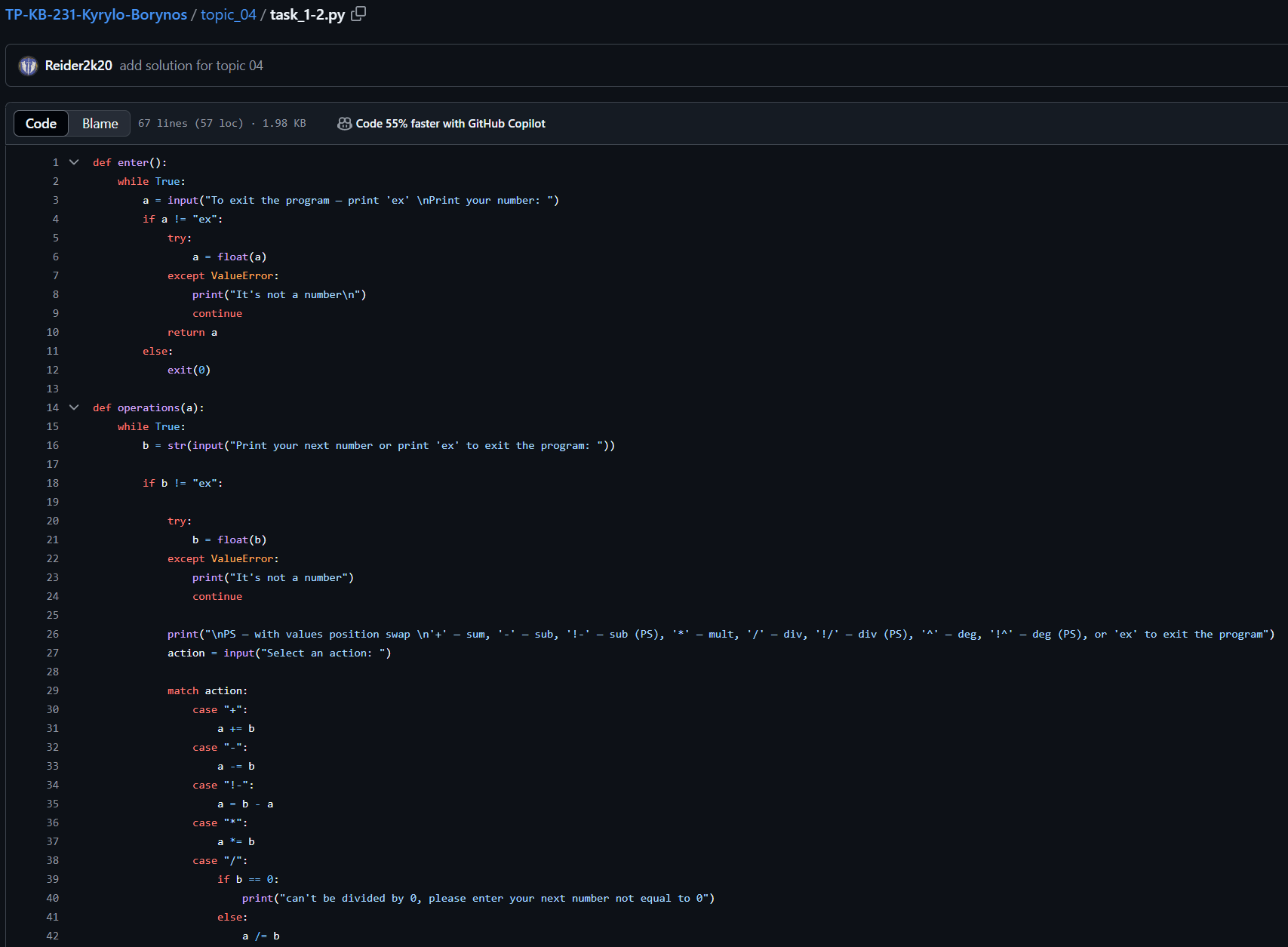
Ця програма відрізняється від попередньої тільки можливістю введення користувачем будь-яких даних для запису в змінні, це стало можливим завдяки додаванню структур try except, які дозволяють не зупиняту виконання програми при виникненні помилки з переведенням змінних в тип float з типу string, якщо введені користувачем дані неможливо було перевести в даний тип, через додавання цих структур, ще довелось додати нескінченний запит на введення першого значення, також при введені першого значення тепер можна завершити виконання програми.

Текст програми:

|  |
| --- |
| def enter():  while True:  a = input("To exit the program — print 'ex' \nPrint your number: ")  if a != "ex":  try:  a = float(a)  except ValueError:  print("It's not a number\n")  continue  return a  else:  exit(0)  def operations(a):  while True:  b = str(input("Print your next number or print 'ex' to exit the program: "))  if b != "ex":  try:  b = float(b)  except ValueError:  print("It's not a number")  continue    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\_04/task\_1-2.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**

Необхідно написати програму, яка буде запитувати в користувача його вибір, камінь, ножиці або папір, після чого випадково обирати 1 з 3 варіантів і порівнювати вибори, після чого виводити результат гри.

Хід виконання завдання:

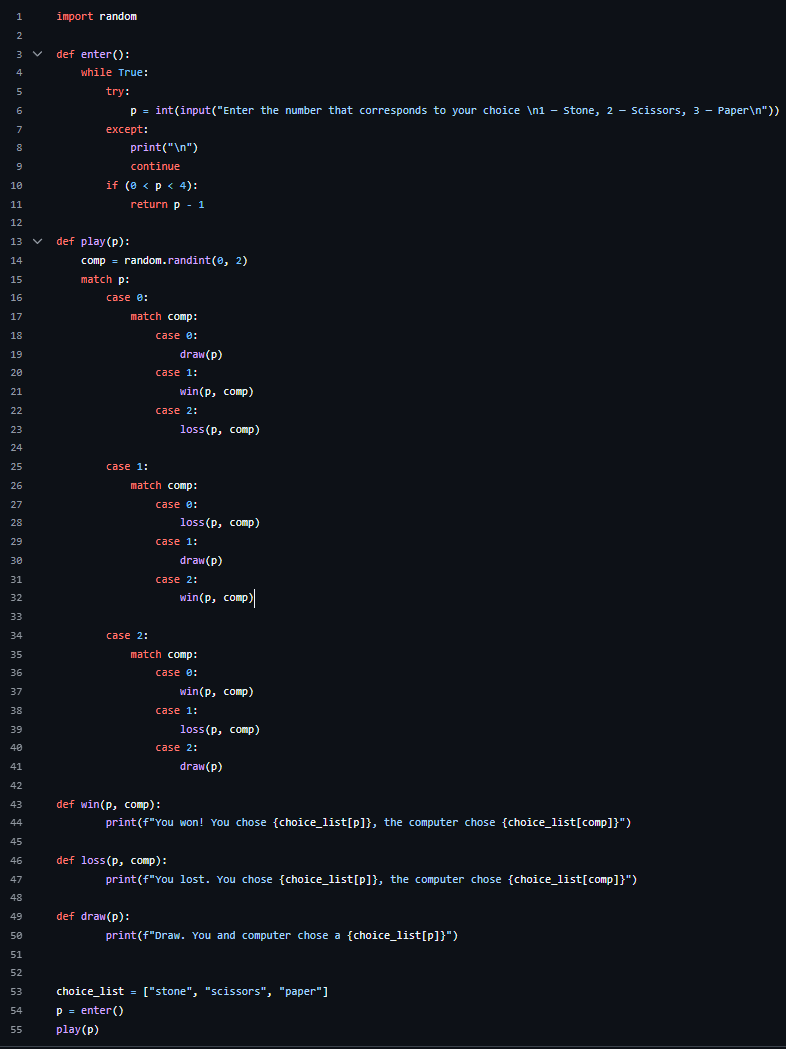
Програма запитує число, яке буде означати вибір каменя, ножиць або папіру, після чого за допомогою бібліотеки “random” сама випадково обираэ свій варіант, після чого порівнює його з вибором користувача та виводить результат, де вказує, що обрали обидві сторони.

Текст програми:

|  |
| --- |
| import random  def enter():  while True:  try:  p = int(input("Enter the number that corresponds to your choice \n1 — Stone, 2 — Scissors, 3 — Paper\n"))  except:  print("\n")  continue  if (0 < p < 4):  return p - 1  def play(p):  comp = random.randint(0, 2)  match p:  case 0:  match comp:  case 0:  draw(p)  case 1:  win(p, comp)  case 2:  loss(p, comp)  case 1:  match comp:  case 0:  loss(p, comp)  case 1:  draw(p)  case 2:  win(p, comp)    case 2:  match comp:  case 0:  win(p, comp)  case 1:  loss(p, comp)  case 2:  draw(p)  def win(p, comp):  print(f"You won! You chose {choice\_list[p]}, the computer chose {choice\_list[comp]}")  def loss(p, comp):  print(f"You lost. You chose {choice\_list[p]}, the computer chose {choice\_list[comp]}")  def draw(p):  print(f"Draw. You and computer chose a {choice\_list[p]}")  choice\_list = ["stone", "scissors", "paper"]  p = enter()  play(p) |

Посилання на github: https://github.com/Reider2k20/TP-KB-231-Kyrylo-Borynos/blob/main/topic\_05/task\_1.py

Знімок екрану з посилання на github:



**Конвертор валюти**

Необхідно написати програму, яка буде конвертувати іноземну валюту в українську гривню.

Хід виконання завдання:

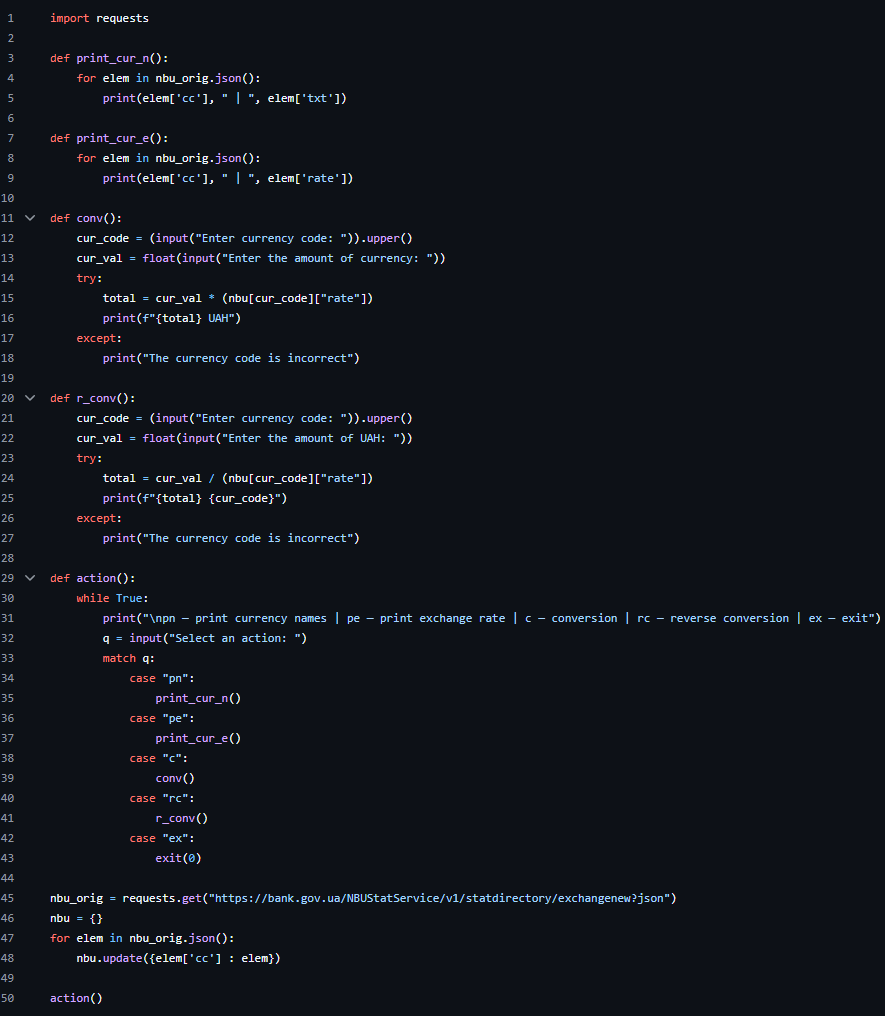
Програма спочатку бере з сайту НБУ список різних, який складається зі словників, які в свою чергу містять інформацію про актуальний курс цієї валюти до гривні, назву українською та код цієї валюти (це інформація, яка буде використовуватись програмою), після чого створює словник з цих словників, ключами до яких будуть коди цих валют, це потрібно для максимальної оптимізації пошуку потрібної валюти, для виведення інформації буде краще використовувати оригінальний список. На вибір користувача доступні наступні дії: виведення кодів валют разом із назвами, виведення кодів валют разом із курсами, конвертація, обернена конвертація та завершення роботи програми. Конвертація працює наступним чином, користувач вводить код валюти не зважаючи на регістр написаних ним літер, та кількість цієї валюти для конвертації в гривні, після чого отримує значення в гривнях, обернена конвертація працює так само, але замість переведення з іноземної валюти в гривні, переводить введену кількість гривень в іноземну валюту.

Текст програми:

|  |
| --- |
| import requests  def print\_cur\_n():  for elem in nbu\_orig.json():  print(elem['cc'], " | ", elem['txt'])  def print\_cur\_e():  for elem in nbu\_orig.json():  print(elem['cc'], " | ", elem['rate'])  def conv():  cur\_code = (input("Enter currency code: ")).upper()  cur\_val = float(input("Enter the amount of currency: "))  try:  total = cur\_val \* (nbu[cur\_code]["rate"])  print(f"{total} UAH")  except:  print("The currency code is incorrect")  def r\_conv():  cur\_code = (input("Enter currency code: ")).upper()  cur\_val = float(input("Enter the amount of UAH: "))  try:  total = cur\_val / (nbu[cur\_code]["rate"])  print(f"{total} {cur\_code}")  except:  print("The currency code is incorrect")  def action():  while True:  print("\npn — print currency names | pe — print exchange rate | с — conversion | rс — reverse conversion | ex — exit")  q = input("Select an action: ")  match q:  case "pn":  print\_cur\_n()  case "pe":  print\_cur\_e()  case "c":  conv()  case "rc":  r\_conv()  case "ex":  exit(0)  nbu\_orig = requests.get("https://bank.gov.ua/NBUStatService/v1/statdirectory/exchangenew?json")  nbu = {}  for elem in nbu\_orig.json():  nbu.update({elem['cc'] : elem})  action() |

Посилання на github: https://github.com/Reider2k20/TP-KB-231-Kyrylo-Borynos/blob/main/topic\_05/task\_2.py

Знімок екрану з посилання на github:



**Модульний калькулятор**

Необхідно розділити написану раніше програму-калькулятор на різні файли.

Хід виконання завдання:

Програма поділена на різні файли, функції з яких будуть використовуватись в основному файлі “calc.py”, тобто всі обчислення відбуваються в функціях файла “functions.py”, тоді як функції введення даних беруться з “operations.py”, нажаль в мене не вийшло повністю розділити код на ці файли через особливості структури саме цього калькулятора, але я намагався зробити це максимально наближено до поставленої задачі.

Текст програми:

calc.py:

|  |
| --- |
| from operations import enter, actions  from functions import sum, sub, rsub, mult, div, rdiv, deg, rdeg  def calculation(a):  while True:  b = enter()  act = actions(a)  match act:  case "+":  a = sum(a, b)  case "-":  a = sub(a, b)  case "!-":  a = rsub(a, b)  case "\*":  a = mult(a, b)  case "/":  a = div(a, b)  case "!/":  a = rdiv(a, b)  case "^":  a = deg(a, b)  case "!^":  a = rdeg(a, b)  print("\nyour current result = ", a)  calculation(enter()) |

functions.py:

|  |
| --- |
| def sum(a, b):  return a + b  def sub(a, b):  return a - b  def rsub(a, b):  return b - a  def mult(a, b):  return a \* b  def div(a, b):  if b == 0:  return "can't be divided by 0, please enter your next number not equal to 0"  else:  return a / b  def rdiv(a, b):  if a == 0:  return "can't be divided by 0, now you result equal to 0, try another action"  else:  return b / a  def deg(a, b):  return a \*\* b  def rdeg(a, b):  return b \*\* a |

operations.py:

|  |
| --- |
| def enter():  while True:  a = input("To exit the program — print 'ex' \nPrint your number: ")  if a != "ex":  try:  a = float(a)  except ValueError:  print("It's not a number\n")  continue  return a  else:  exit(0)  def actions(a):  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: ")  if action == "ex":  print("\nyour current result = ", a)  exit(0)  elif action in ["+", "-", "!-", "\*", "/", "!/", "^", "!^"]:  return action  else:  print("This is not an action") |

Посилання на github: https://github.com/Reider2k20/TP-KB-231-Kyrylo-Borynos/tree/main/topic\_05/task\_3

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 для модульного калькулятора**

Необхідно розробити механізм логування значень та дій, що виконуються програмою на основі виконаного раніше модульного калькулятора (3 завдання 5 теми).

Хід виконання завдання:

Відмінності від звичайного модульного калькулятора знаходяться лише в файлі calc.py, інші файли не змінювались, для виконання знадобилось під’єднати 2 бібліотеки, datetime для визначення поточного часу для написання рядка в логах, а os для написання росташування файла логів, “os.path.dirname(\_\_file\_\_)” бере шлях до директорії, в якій знаходиться цей файл, а “os.path.join()” об’єднує цей шлях разом із назвою потрібного файлу (Calc\_Log.txt), в циклі додатково записується старе значення “a”, бо це значення оновлюєтсья після кожної операції, після виконання операції перевіряється значення “act” яка визначає операцію, якщозначення для “act” відсутнє, то це означає, що при виборі дії не було обрано жодного з запропонованих варіантів, отже і операції не було, тоді в логи нічого не записується, якщо ж значення “act” це дія з перестановкою значень, то в логи записується цей вираз, але значення поміняні місцями, інакше це звичайна дія з звичайною розстановкою значень. На початку кожного запису пишеться дата та час виконаної операції.

Текст програми (calc.py):

|  |
| --- |
| from operations import enter, actions  from functions import sum, sub, mult, div, rdiv, deg  from datetime import datetime  import os  file\_path = os.path.join(os.path.dirname(\_\_file\_\_), "Calc\_Log.txt")  def calculation(a):  while True:  a\_old = a  b = enter()  act = actions(a)  match act:  case "+":  a = sum(a, b)  case "-":  a = sub(a, b)  case "!-":  a = sub(b, a)  case "\*":  a = mult(a, b)  case "/":  a = div(a, b)  case "!/":  a = rdiv(a, b)  case "^":  a = deg(a, b)  case "!^":  a = deg(b, a)  print("\nyour current result = ", a)  with open(file\_path, "a") as file:  log\_time = datetime.now().strftime("%Y-%m-%d %H:%M:%S")  if not act:  pass  elif act in ["!-", "!/","!^"]:  file.write(f"{log\_time} | {b} {act[1]} {a\_old} = {a}\n")  else:  file.write(f"{log\_time} | {a\_old} {act} {b} = {a}\n")  calculation(enter()) |

Посилання на github: https://github.com/Reider2k20/TP-KB-231-Kyrylo-Borynos/tree/main/topic\_06/task\_1

Знімок екрану з посилання на github:



**Сортування за допомогою lambda функції**

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:

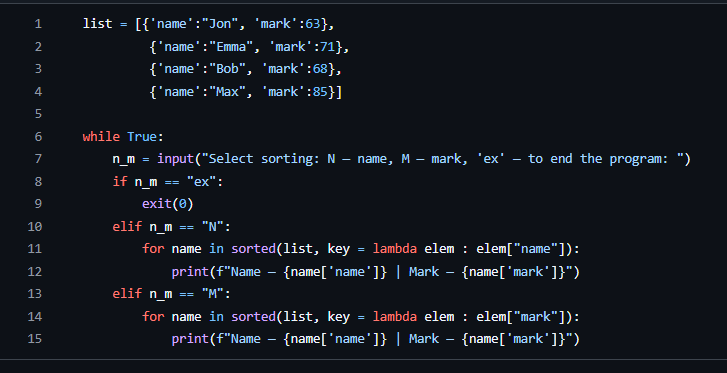
Програма являє собою нескінченний цикл із можливістю завершення написанням “ex”, якщо написати “N”, то сортування відбудеться за ім’ям, якщо ж написати “M”, то сортування відбудеться за оцінкою.

Текст програми:

|  |
| --- |
| list = [{'name':"Jon", 'mark':63},  {'name':"Emma", 'mark':71},  {'name':"Bob", 'mark':68},  {'name':"Max", 'mark':85}]  while True:  n\_m = input("Select sorting: N — name, M — mark, 'ex' — to end the program: ")  if n\_m == "ex":  exit(0)  elif n\_m == "N":  for name in sorted(list, key = lambda elem : elem["name"]):  print(f"Name — {name['name']} | Mark — {name['mark']}")  elif n\_m == "M":  for name in sorted(list, key = lambda elem : elem["mark"]):  print(f"Name — {name['name']} | Mark — {name['mark']}") |

Посилання на github: https://github.com/Reider2k20/TP-KB-231-Kyrylo-Borynos/blob/main/topic\_06/task\_2.py

Знімок екрану з посилання на github:



Звіт до Теми №7

ООП

Під час виконання практичного завдання до Теми №7 було надано варіанти рішення до наступних задач:

**Розробка класу Student та робота з ним**

Необхідно розробити клас **Student** атрибутами якого э два параметра **name** та **age**. Створити список елементами якого є об'єкти класу **Student**. Написати цикл який виводить на екран елементи списку у відсортованому порядку. Для сортування використати стандартну функцію **sorted**. Функція **sorted** має використовувати **lambda** функцію для визначення ключа сортування.

Хід виконання завдання:

Спочатку створюється список, елементами якого є об’єкти класу Student, які приймають 2 значення для своїх атрибутів, після цього, виконується сортування за ключем атрибуту name в кожного елемента, та запускається цикл, що перебирає всі елементи та друкує їх, при друку об’єктів класу Student виконується вміст функції \_\_str\_\_, яка виводить значення атрибутів об’єкта, потім відбувається теж саме, але з сортуванням за другим атрибутом.

Текст програми:

|  |
| --- |
| class Student:      def \_\_init\_\_(self, name, age):          self.name = name          self.age = age        def \_\_str\_\_(self):          return f"Student name — {self.name}, age — {self.age}"    students = [      Student("Bob", 23),      Student("Max", 25),      Student("Jon", 18),      Student("Emma", 20),  ]  print("Sort by name")  for student in sorted(students, key = lambda student : student.name):              print(student)  print("\nSort by age")  for student in sorted(students, key = lambda student : student.age):              print(student) |

Посилання на github: https://github.com/Reider2k20/TP-KB-231-Kyrylo-Borynos/blob/main/topic\_07/task\_2-3.py

Знімок екрану з посилання на github:



**Використання класів для модульного калькулятора**

Використовуючи принципи ООП переписати програму Калькулятор. Завдання має бути виконано використовуючи модульний підхід.

Хід виконання завдання:

Програма запускається з файлу main\_calc.py, в ній створюється об’єкт класу Calc, який знаходиться в файлі calc.py, після чого запускається основна функція цього класу, в якій виконуються основні дії, такі як порівняння введених даних та виклик функцій з різних класів, в ініціалізації створюються об’єкти всіх необхідних класів, бо без цього не можна буде викликати функції цих класів напряму, якщо в них вводиться self, в класі Functions э лише статичні методи, бо це клас, який містить лише функції з математичними операціями, і не потребує наявності атрибутів класу або інших функцій, так само і клас Operations. Після завершення обробки вибору користувача викликається функція класу Calc для логування, яка приймає 5 параметрів: self для звернення до атрибуту logger, який є об’єктом класу Logger, нове та старе значення відповіді, додаткове число та обрана дія, потім виконується логування операції.

Текст програми (main\_calc.py):

|  |
| --- |
| from calc import Calc  if \_\_name\_\_ == "\_\_main\_\_":      calc = Calc()      calc.calculation() |

Текст програми (calc.py):

|  |
| --- |
| from operations import Operations  from functions import Functions  from logging import Logger    class Calc:      def \_\_init\_\_(self):          self.functions = Functions()          self.operations = Operations()          self.logger = Logger()        def wr\_log(self, a, b, act, a\_old):          self.logger.log(a, b, act, a\_old)      def calculation(self):          a = self.operations.enter()          while True:              a\_old = a              b = self.operations.enter()              act = self.operations.actions(a)              match act:                  case "+":                      a = self.functions.sum(a, b)                  case "-":                      a = self.functions.sub(a, b)                  case "!-":                      a = self.functions.sub(b, a)                  case "\*":                      a = self.functions.mult(a, b)                  case "/":                      if b == 0:                          print("can't be divided by 0, please enter your next number not equal to 0\n")                          continue                      a = self.functions.div(a, b)                  case "!/":                      if a == 0:                          print("can't be divided by 0, now you result equal to 0, try another action\n")                          continue                      a = self.functions.div(b, a)                  case "^":                      a = self.functions.deg(a, b)                  case "!^":                      a = self.functions.deg(b, a)              print("\nyour current result = ", a)              self.wr\_log(a, b, act, a\_old) |

Текст програми (functions.py):

|  |
| --- |
| class Functions:      @staticmethod      def sum(a, b):          return a + b        @staticmethod      def sub(a, b):          return a - b        @staticmethod      def mult(a, b):          return a \* b        @staticmethod      def div(a, b):          return a / b        @staticmethod      def deg(a, b):          return a \*\* b |

Текст програми (logging.py):

|  |
| --- |
| import os  from datetime import datetime  class Logger:      def \_\_init\_\_(self, file\_name="Calc\_Log.txt"):          self.file\_path = os.path.join(os.path.dirname(\_\_file\_\_), file\_name)      def log(self, a, b, act, a\_old):          log\_time = datetime.now().strftime("%Y-%m-%d %H:%M:%S")          if not act:              pass          elif act in ["!-", "!/","!^"]:              with open(self.file\_path, "a") as file:                  file.write(f"{log\_time} | {b} {act[1]} {a\_old} = {a}\n")          else:              with open(self.file\_path, "a") as file:                  file.write(f"{log\_time} | {a\_old} {act} {b} = {a}\n") |

Текст програми (operations.py):

|  |
| --- |
| class Operations:      @staticmethod      def enter():          while True:              a = input("To exit the program — print 'ex' \nPrint your number: ")              if a != "ex":                  try:                      a = float(a)                  except ValueError:                      print("It's not a number\n")                      continue                  return a              else:                  exit(0)        @staticmethod      def actions(a):          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: ")          if action == "ex":              print("\nyour current result = ", a)              exit(0)          elif action in ["+", "-", "!-", "\*", "/", "!/", "^", "!^"]:              return action          else:              print("This is not an action") |

Посилання на github: https://github.com/Reider2k20/TP-KB-231-Kyrylo-Borynos/tree/main/topic\_07/task\_4

Знімок екрану з посилання на github:

