**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**

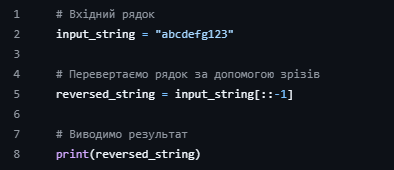
Необхідно рядок, що має вигляд "abcdefg123" перетворити наступним чином "321gfedcba", вважаючи сталою довжину рядку в 10 символів.

Хід виконання завдання:

1. Для початку я стоврит змінну input\_string в яку помістили текст "abcdefg123".
2. Потім з допомогою функції зрізу reversed\_string = input\_string[::-1] я перевернув рядок.
3. В кінці добавив функцію виводу print(reversed\_string), щоб побачити результат.

Текст програми:

|  |
| --- |
| # Вхідний рядок  input\_string = "abcdefg123"  # Перевертаємо рядок за допомогою зрізів  reversed\_string = input\_string[::-1]  # Виводимо результат  print(reversed\_string) |



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

Необхідно виконати тестування функцій, що працюють з рядками: strip, capitalize, title, upper, lower.

Хід виконання завдання:

1. Спочатку виводиться оригінальний рядок, що буде тестуватись. У цьому випадку рядок test\_str = " hello world! "
2. Надаю цьому рядку функцію print(f"Оригінал: '{test\_str}'"), щоб показати початковий вигляд тексту. Літера f перед рядком дозволяє безпосередньо вбудовувати значення змінних або виразів у рядок.
3. **Strip** - видаляє пробіли або інші вказані символи з початку і кінця рядка. Надаю цьому рядку функцію print(f"strip: '{test\_str.strip()}'") для виведення тексту.
4. **Capitalize** - робить першу літеру рядка великою, а всі інші — малими. Надаю цьому рядку функцію print(f"capitalize: '{test\_str.capitalize()}'") для виведення тексту.
5. **Title** - робить кожне слово в рядку з великої літери. Надаю цьому рядку функцію print(f"title: '{test\_str.title()}'") для виведення тексту.
6. **Upper** - переводить всі символи рядка в верхній регістр. Надаю цьому рядку функцію print(f"upper: '{test\_str.upper()}'") для виведення тексту.
7. **Lower** -переводить всі символи рядка в нижній регістр. Надаю цьому рядку функцію print(f"lower: '{test\_str.lower()}'") для виведення тексту.

Текст програми:

|  |
| --- |
| # Тестовий рядок  test\_str = " hello world! "  print(f"Оригінал: '{test\_str}'")  # 1. strip(видаляє пробіли або інші вказані символи з початку і кінця рядка)  print(f"strip: '{test\_str.strip()}'")  # 2. capitalize(робить першу літеру рядка великою, а всі інші — малими.)  print(f"capitalize: '{test\_str.capitalize()}'")  # 3. title(робить кожне слово в рядку з великої літери.)  print(f"title: '{test\_str.title()}'")  # 4. upper(переводить всі символи рядка в верхній регістр.)  print(f"upper: '{test\_str.upper()}'")  # 5. lower(переводить всі символи рядка в нижній регістр.)  print(f"lower: '{test\_str.lower()}'") |



**Написати функцію пошуку дискримінанту квадратного рівняння.**

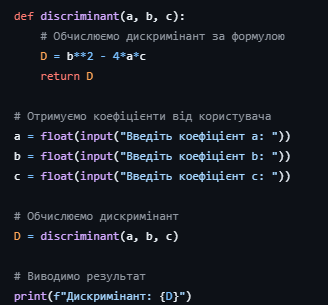
Треба написати функцію пошуку дискримінанту квадратного рівняння.

Хід виконання завдання:

1. Спочатку я створив функцію, яка буде обчислювати дискримінант на основі коефіцієнтів a, b і c. Дискримінант обчислюється за формулою D=b^2−4ac, тому ця формула буде основою для функції.
2. Для того, щоб можна було ввести коефіцієнти, використовуємо функцію input(). Важливо перетворити введені дані у числа (тип float), щоб обробляти не лише цілі, але й дробові значення.
3. Запитуємо кожен коефіцієнт окремо і зберігаємо їх у змінні.
4. Після того як користувач ввів значення коефіцієнтів, їх потрібно передати в нашу функцію discriminant(a,b,c). Результат обчислення дискримінанту зберігаємо у змінну D.
5. Використовуємо функцію print(), щоб вивести результат обчислення дискримінанту на екран. Використовуємо f-строку для зручного форматування виведення.

Текст програми:

|  |
| --- |
| def discriminant(a, b, c):      # Обчислюємо дискримінант за формулою      D = b\*\*2 - 4\*a\*c      return D  # Отримуємо коефіцієнти від користувача  a = float(input("Введіть коефіцієнт a: "))  b = float(input("Введіть коефіцієнт b: "))  c = float(input("Введіть коефіцієнт c: "))  # Обчислюємо дискримінант  D = discriminant(a, b, c)  # Виводимо результат  print(f"Дискримінант: {D}") |



Посилання на github:

<https://github.com/k1ryushaaa/TP-KB-231-Yakovenko-Kyrylo>

Звіт до Теми №2

Умовний перехід

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

**Написання функції пошуку коренів квадратного рівняння використовуючи функцію розрахунку дискримінанту з попередньої теми та умовні переходи.**

Необхідно написати функцію пошуку коренів квадратного рівняння використовуючи функцію розрахунку дискримінанту з попередньої теми та умовні переходи.  
  
Хід виконання завдання:

1. Спочатку викликав функцію «enter()», яка вивела підказку для користувача і попросила ввести коефіцієнти для рівняння вигляду ax^2 + bx + c. Функція «enter()» зчитала значення коефіцієнтів a, b і c від користувача та повернула їх.

2. Далі, з допомогою функції «discriminant(a, b, c)» я обчислив дискримінант за формулою D = b^2 - 4ac.

3. Потім викликав функцію «find\_roots(a, b, c)», яка: спочатку обчислила дискримінант D (використовуючи функцію «discriminant»).

- Якщо D > 0, обчислила два дійсні корені x\_1 і x\_2 за формулами:

x1 = (-b + D\*\*0.5) / (2 \* a)

x2 = (-b - D\*\*0.5) / (2 \* a)

і повернула їх як результат.

Якщо D = 0, обчислила один дійсний корінь x за формулою:

x = -b / (2 \* a)

і повернула його як результат.

Якщо D < 0, повернула повідомлення про відсутність дійсних коренів.

4. На завершення вивів результат виконання функції «find\_roots», щоб побачити корені рівняння або повідомлення про їхню відсутність.

5. Додав команду «input("Натисніть Enter, щоб завершити...")», щоб користувач зміг переглянути результат перед закриттям програми.  
Текст програми:

|  |
| --- |
| def enter():  print(" (X це x^2): aX+bx+c")  a = float(input('Напишіть a: '))  b = float(input('Напишіть b: '))  c = float(input('Напишіть c: '))  return a, b, c  def discriminant(a, b, c):  return b\*\*2 - 4 \* a \* c  def find\_roots(a, b, c):  D = discriminant(a, b, c)    if D > 0:  x1 = (-b + D\*\*0.5) / (2 \* a)  x2 = (-b - D\*\*0.5) / (2 \* a)  return f"Два дійсні корені: x1 = {x1}, x2 = {x2}"  elif D == 0:  x = -b / (2 \* a)  return f"Один дійсний корінь: x = {x}"  else:  return "Дійсних коренів немає"    a, b, c = enter()  d = discriminant(a, b, c)  print(find\_roots(a, b, d))  input("Натисніть Enter, щоб завершити...") |



**Написання програми калькулятора використовуючи if else конструкцію.**

Необхідно написати програму калькулятор використовуючи if else конструкцію. Кожна операція має бути виконана в окремій функції.  
Хід виконання завдання:  
Хід виконання завдання:

1. Спочатку визначив функції для кожної математичної операції:

* add(x, y) — функція для додавання, яка повертає суму чисел x та y .
* subtract(x, y) — функція для віднімання, яка повертає різницю чисел x та y .
* multiply(x, y) — функція для множення, яка повертає добуток чисел x та y .
* divide(x, y) — функція для ділення, яка перевіряє, чи y не дорівнює нулю, і якщо це так, виконує ділення x на y. Якщо y = 0, повертає повідомлення "Помилка: ділення на нуль".

2. Створив функцію calculator() , яка:

Просить користувача вибрати операцію (`+`, `-`, `\*`, `/`) і вводить її значення у змінну operation.

Запитує у користувача два числа (перше і друге) та зберігає їх у змінні num1 і num2.

3. Виконує перевірку вибраної операції:

* Якщо вибрано `+`, викликає функцію add(num1, num2) і виводить результат.
* Якщо вибрано `-`, викликає функцію subtract(num1, num2) і виводить результат.
* Якщо вибрано `\*`, викликає функцію multiply(num1, num2) і виводить результат.
* Якщо вибрано `/`, викликає функцію divide(num1, num2) і виводить результат.
* Якщо вибрано некоректну операцію, виводить повідомлення "Помилка: некоректний вибір операції".

4. Викликав функцію calculator() для запуску програми.

5. Додав команду input("Натисніть Enter, щоб завершити..."), щоб користувач міг побачити результат перед закриттям програми.

Текст програми:

|  |
| --- |
| def add(x, y):  return x + y  def subtract(x, y):  return x - y  def multiply(x, y):  return x \* y  def divide(x, y):  return x / y if y != 0 else "Помилка: ділення на нуль"  def calculator():  operation = input("Оберіть операцію (+, -, \*, /): ")  num1 = float(input("Введіть перше число: "))  num2 = float(input("Введіть друге число: "))    if operation == '+':  print("Результат:", add(num1, num2))  elif operation == '-':  print("Результат:", subtract(num1, num2))  elif operation == '\*':  print("Результат:", multiply(num1, num2))  elif operation == '/':  print("Результат:", divide(num1, num2))  else:  print("Помилка: некоректний вибір операції.")  calculator()  input("Натисніть Enter, щоб завершити...") |



**Написати програму калькулятор використовуючи match конструкцію.**

Необхідно написати програму калькулятор використовуючи match конструкцію. Кожна операція має бути виконана в окремій функції.  
  
Хід виконання завдання:

1. Спочатку визначив функції для кожної математичної операції:

* add(x, y) — функція для додавання, яка повертає суму чисел x та y .
* subtract(x, y) — функція для віднімання, яка повертає різницю чисел x та y .
* multiply(x, y) — функція для множення, яка повертає добуток чисел x та y .
* divide(x, y) — функція для ділення, яка перевіряє, чи y не дорівнює нулю, і якщо це так, виконує ділення x на y. Якщо y = 0, повертає повідомлення «Помилка: ділення на нуль».

2. Створив функцію calculator() , яка:

Просить користувача вибрати операцію (`+`, `-`, `\*`, `/`) і вводить її значення у змінну operation.

Запитує у користувача два числа (перше і друге) та зберігає їх у змінні num1 і num2.

1. Використав конструкцію match для вибору операції залежно від значення змінної operation:

Якщо операція `+`, викликає функцію add(num1, num2) і виводить результат.

Якщо операція `-`, викликає функцію subtract(num1, num2) і виводить результат.

Якщо операція `\*`, викликає функцію multiply(num1, num2) і виводить результат.

Якщо операція `/`, викликає функцію divide(num1, num2) і виводить результат.

Якщо введено некоректну операцію, виконує case \_ і виводить повідомлення «Помилка: некоректний вибір операції».

1. Викликав функцію calculator() для запуску калькулятора.
2. Додав команду input(«Натисніть Enter, щоб завершити»), щоб користувач міг переглянути результат перед завершенням програми.

Текст програми:

|  |
| --- |
| def add(x, y):  return x + y  def subtract(x, y):  return x – y  def multiply(x, y):  return x \* y  def divide(x, y):  return x / y if y != 0 else “Помилка: ділення на нуль”  def calculator():  operation = input(“Оберіть операцію (+, -, \*, /): “)  num1 = float(input(“Введіть перше число: “))  num2 = float(input(“Введіть друге число: “))    match operation:  case ‘+’:  print(“Результат:”, add(num1, num2))  case ‘-‘:  print(“Результат:”, subtract(num1, num2))  case ‘\*’:  print(“Результат:”, multiply(num1, num2))  case ‘/’:  print(“Результат:”, divide(num1, num2))  case \_:  print(«Помилка: некоректний вибір операції.»)  calculator()  input(«Натисніть Enter, щоб завершити») |

  
Посилання на github:

<https://github.com/k1ryushaaa/TP-KB-231-Yakovenko-Kyrylo>

Звіт до Теми №3

Цикли

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

**Написання програми калькулятор з постійними запитами на введення нових даних та операцій. За основу взяття програми калькулятор з попередньої теми. Реалізування механізму завершення програми після отримання відповідної команди.**

Необхідно 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 виконная завдання:

**Створення функції enter:**

* Ця функція запитує у користувача одне число, яке зберігається у змінній a, і повертає його.
* Введене число використовується як початковий результат для подальших операцій.

**Створення функції operations:**

Ця функція приймає число a, яке було введено раніше, і запускає нескінченний цикл для виконання обчислень.

* У циклі запитує:
* Нове число або команду ex для виходу.
* Тип математичної операції, яку слід виконати з поточним результатом aa та введеним числом b.
* Пояснення операцій:
* **+**: додає b до a.
* -: віднімає b від a.
* **!-**: виконує віднімання, змінюючи порядок операндів (b – a).
* **\***: множить a на b.
* **/**: ділить aa на b (з перевіркою, щоб b≠0).
* **!/**: виконує ділення b на aa (з перевіркою, щоб a≠0).
* **^**: підносить a до степеня b.
* **!^**: підносить b до степеня a.
* **ex**: завершує програму та виводить фінальний результат.
* Якщо введена некоректна операція, програма повідомляє про помилку.

1. **Логіка перевірок у функції operations:**

* Якщо користувач вводить ex, програма виводить поточний результат і завершує виконання.
* Для операцій ділення (/, !/), перевіряє, чи дільник не дорівнює нулю, щоб уникнути помилки.
* Для інших операцій просто виконує відповідну математичну дію.

1. **Виведення результату:**

Після кожної операції виводиться поточний результат aa, щоб користувач міг бачити проміжний підсумок.

1. **Запуск програми:**

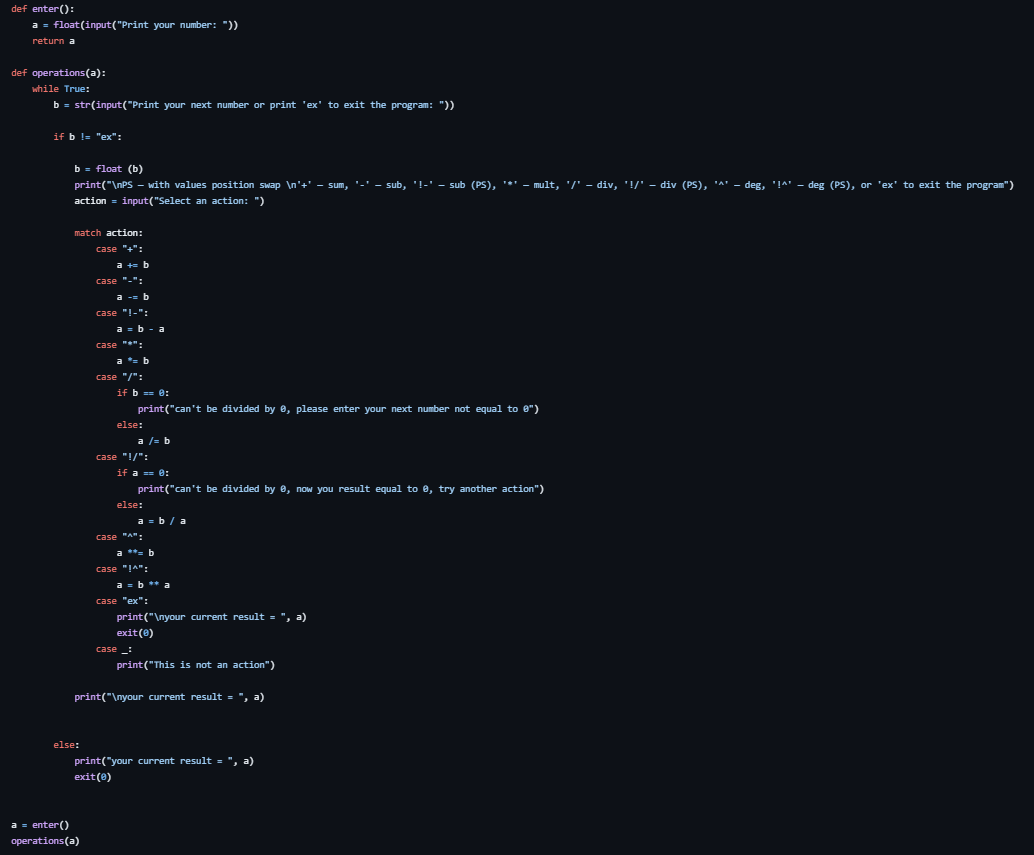
* Спочатку викликається функція enter(), яка отримує початкове число від користувача.
* Потім запускається функція operations(a) для виконання обчислень.

1. **Фінал:**

Програма працює доти, доки користувач не введе команду ex, після чого виводить остаточний результат і завершує виконання.

Текст програми:

|  |
| --- |
| def enter():      a = float(input(“Напишіть ваше число: “))      return a  def operations(a):      while True:          b = str(input(«Напишіть друге число або напишіть'ex' для виходу із програми: «))          if b != “ex”:              b = float (b)              print(“\nPS — зі значеннями зміни позиції \n’+’ — сумма, ‘-‘ — різниця, ‘!-‘ — різниця (PS), ‘\*’ — добуток, ‘/’ — ділення, ‘!/’ — ділення (PS), ‘^’ — піднесення в степінь, ‘!^’ — піднесення в степінь (PS), або ‘ex’ для того щоб вийти із програми”)              action = input(“Виберіть дію: “)              match action:                  case “+”:                      a += b                  case “-“:                      a -= b                  case “!-“:                      a = b – a                  case “\*”:                      a \*= b                  case «/»:                      if b == 0:                          print(«не ділиться на 0, будь ласка, введіть наступне число, яке не дорівнює 0.»)                      else:                          a /= b                  case “!/”:                      if a == 0:                          print(«не можна поділити на 0, результат дорівнює 0, спробуйте іншу дію»)                      else:                          a = b / a                  case “^”:                      a \*\*= b                  case “!^”:                      a = b \*\* a                  case “ex”:                      print(«\nваш поточний результат = «, a)                      exit(0)                  case \_:                      print(«Це не працює»)              print(«\nваш поточний результат = «, a)          else:              print(«ваш поточний результат = «, a)              exit(0)  a = enter()  operations(a) |

  
**Написання програми тестування функцій списків таких як: extend(), append(), insert(id, val), remove(val), clear(), sort(), reverse(), copy()**

Необхідно написати програму тестування функцій списків таких як: extend(), append(), insert(id, val), remove(val), clear(), sort(), reverse(), copy()  
  
Хід виконання завдання:

**Підготовка даних:**

* Створено список list1, який містить початкові значення: [1, 4, 6, 2, 3, 10, 8, 4].
* Оригінальний список виводиться на екран перед застосуванням будь-яких функцій.

**Опис функцій:**

* + **extend\_test(list)**:
    - Створює новий список new\_list = [14, 12].
    - Використовує метод extend(), який додає елементи new\_list до кінця переданого списку.
    - Виводить результат.
  + **append\_test(list)**:
    - Використовує метод append(), який додає значення 15 в кінець списку.
    - Виводить результат.
  + **insert\_test(list)**:
    - Використовує метод insert(3, 23), який вставляє значення 23 на позицію з індексом 3.
    - Виводить результат із поясненням, що елементи правіше індексу 3 зміщуються.
  + **remove\_test(list)**:
    - Використовує метод remove(4), який видаляє перший елемент із значенням 4.
    - Виводить результат.
  + **clear\_test(list)**:
    - Використовує метод clear(), який видаляє всі елементи списку, залишаючи його порожнім.
    - Виводить результат.
  + **sort\_test(list)**:
    - Використовує метод sort(), який сортує елементи списку в порядку зростання.
    - Виводить результат.
  + **reverse\_test(list)**:
    - Використовує метод reverse(), який змінює порядок елементів списку на зворотній.
    - Виводить результат.
  + **copy\_test(list)**:
    - Використовує метод copy(), який створює копію списку.
    - Виводить копію списку.

**Виконання функцій:**

* + Для кожної функції передається **копія оригінального списку** (list1.copy()), щоб зберегти початковий список незмінним під час демонстрації роботи функцій.
  + Результат кожної операції виводиться на екран з поясненням.

**Виклик функцій:**

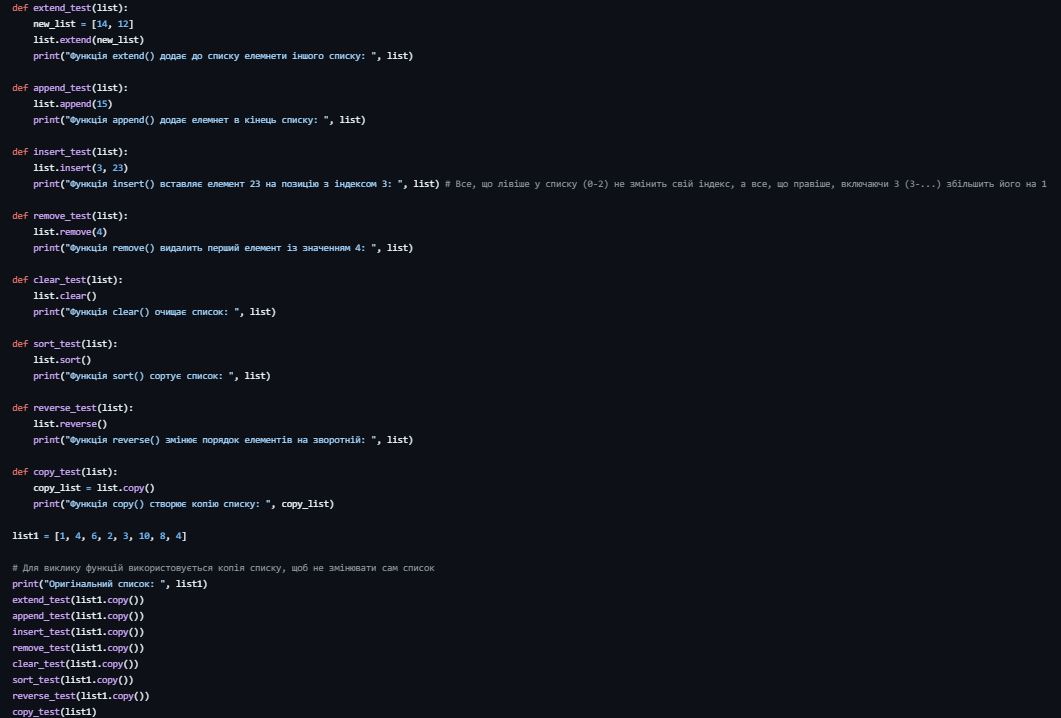
* + Кожна функція виконується послідовно з коментарями та прикладами.
  + Для функції copy\_test() використовується сам список list1, оскільки ця функція створює незалежну копію списку.

**Результат:**

Кожна функція демонструє свою роботу на списку, а початковий список list1 залишається незмінним, оскільки передаються його копії, окрім останнього тесту копіювання.

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) |



**Написання програми тестування функцій словників таких як: update(), del(), clear(), keys(), values(), items()**

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

Хід виконання завдання:

**Підготовка даних:**

* + Створено початковий словник dictionary1 з парами ключ-значення: {'first': 11, 'second': 22, 'third': 33}.
  + Оригінальний словник виводиться на екран перед виконанням будь-яких операцій.

**Опис функцій для роботи зі словниками:**

* + **update\_test(dictionary)**:
    - Створюється новий словник new\_dictionary = {'fourth': 44, 'fifth': 55}.
    - Використовується метод update(), який додає пари ключ-значення з new\_dictionary до існуючого словника. Якщо ключі вже існують, їх значення будуть оновлені.
    - Виводиться результат оновленого словника.
  + **del\_test(dictionary)**:
    - Використовується оператор del, щоб видалити елемент із ключем 'second'.
    - Виводиться результат після видалення.
  + **clear\_test(dictionary)**:
    - Використовується метод clear(), який очищає словник, залишаючи його порожнім.
    - Виводиться результат.
  + **keys\_test(dictionary)**:
    - Використовується метод keys(), який повертає список усіх ключів у словнику.
    - Виводиться список ключів.
  + **values\_test(dictionary)**:
    - Використовується метод values(), який повертає список усіх значень у словнику.
    - Виводиться список значень.
  + **items\_test(dictionary)**:
    - Використовується метод items(), який повертає всі пари ключ-значення у вигляді кортежів.
    - Виводиться результат у вигляді списку кортежів.

1. **Виконання функцій:**

* Для кожної функції передається **копія оригінального словника** (dictionary1.copy()), щоб зберегти початковий словник незмінним під час демонстрації роботи.
* Результати виконання кожної операції виводяться на екран із поясненням.

1. **Виклик функцій:**

Кожна функція викликається послідовно:

* + - update\_test(): демонструє додавання нових пар ключ-значення.
    - del\_test(): демонструє видалення елемента з ключем 'second'.
    - clear\_test(): очищає словник.
    - keys\_test(): виводить усі ключі.
    - values\_test(): виводить усі значення.
    - items\_test(): виводить усі пари ключ-значення.

1. **Результат:**

Кожна функція демонструє відповідну операцію над копією словника, а початковий словник dictionary1 залишається незмінним.

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()) |



**Написати функцію пошуку позиції для вставки нового елементу в список.**

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

Хід виконання завдання:

**Підготовка даних:**

* Створено список імен names\_list, який містить початкові значення: ["Андрій", "Олександр", "Кирило", "Микола", "Дмитро"].
* Виводиться список імен перед початком роботи програми.

**Опис функції sort\_pos:**

* + **Призначення:** Знаходить позицію, куди потрібно вставити новий елемент у впорядкованому списку.
  + **Аргументи:**
    - list: список, у якому шукається позиція.
    - item: новий елемент, для якого визначається позиція вставлення.
  + **Алгоритм роботи:**
    - Ініціалізується змінна insert\_pos зі значенням 0.
    - Програма перебирає елементи списку:
      * Якщо новий елемент item більший за поточний елемент списку elem, позиція збільшується на 1.
      * Якщо значення менше або дорівнює, перебір завершується (знайдено позицію).
    - Повертається позиція insert\_pos.

**Отримання нового імені:**

* + Використовується функція input() для введення нового імені користувачем.
  + Введене ім’я зберігається в змінній name.

**Використання функції sort\_pos:**

Викликається функція sort\_pos, яка визначає позицію для вставлення нового імені в список.

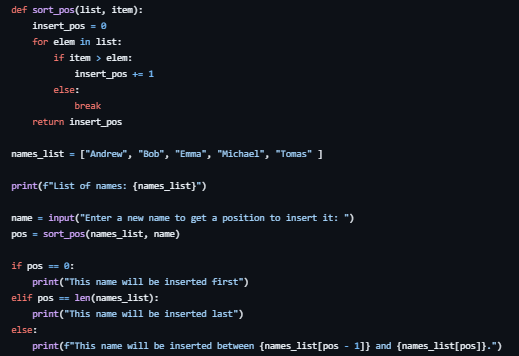
**Результат:**

* + Якщо позиція pos == 0, ім’я буде вставлено першим.
  + Якщо позиція дорівнює довжині списку len(names\_list), ім’я буде вставлено останнім.
  + В іншому випадку виводиться повідомлення, що ім’я буде вставлено між двома існуючими іменами, вказаними за індексами pos - 1 і pos.

**Приклад роботи програми:**

* + Початковий список: ["Андрій", "Олександр", "Кирило", "Микола", "Дмитро"].
  + Введення нового імені, наприклад, "Віталій".
  + Функція визначає позицію для вставлення (між "Андрій" і "Олександр").
  + Результат: "Ця назва буде вставлена між Андрій і Олександр."

|  |
| --- |
| 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 = ["Андрій", "Олександр", "Кирило", "Микола", "Дмитро" ]  print(f"Останні імена: {names\_list}")  name = input("Введіть нове ім’я, щоб отримати позицію для його вставлення: ")  pos = sort\_pos(names\_list, name)  if pos == 0:      print("Це ім'я буде вставлено першим")  elif pos == len(names\_list):      print("Це ім'я буде вставлено останнім")  else:      print(f"Ця назва буде вставлена ​​між {names\_list[pos - 1]} і {names\_list[pos]}.") |



Посилання на github:

<https://github.com/k1ryushaaa/TP-KB-231-Yakovenko-Kyrylo>

Звіт до Теми №4

Виняткові ситуації

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

**Попередні умови**: реалізована програма калькулятор, що використовує метод нескінченного введення даних для обробки. Всі дії (додавання, віднімання, множення, ділення) реалізовані як окремі функції та використовуються у відповідних місцях.

1. Розширити програму калькулятор функцією запитів даних для виконання операцій від користувача, що обробляє виняткові ситуації.
2. Розширити функцію ділення обробкою виняткової ситуації ділення но нуль

Хід виконання завдання:

**Підготовка та вхідні дані:**

* + Реалізована функція **enter()**, яка дозволяє користувачеві ввести початкове число.
  + Введення завершується валідацією:
    - Якщо користувач вводить "ex", програма завершує роботу.
    - Якщо введене значення не є числом, користувач отримує повідомлення про помилку та повторний запит.
  + Після успішного введення повертається числове значення.

**Опис функції operations:**

**Призначення:** Виконує арифметичні операції над введеним числом із подальшими числами.

**Алгоритм роботи:**

Вхід у безкінечний цикл для отримання наступного числа або виходу.

Перевірка введення:

* + - * Якщо введено "ex", програма завершує роботу, виводячи поточний результат.
      * Якщо введене значення не є числом, користувач отримує повідомлення про помилку та повторний запит.

Вибір операції:

* + - * Користувач вводить одну з доступних операцій:
        + '+' — додавання.
        + '-' — віднімання.
        + '!-' — віднімання з позиційною зміною (результат = друге число - перше).
        + '\*' — множення.
        + '/' — ділення (з перевіркою на ділення на 0).
        + '!/' — ділення з позиційною зміною (результат = друге число / перше, перевірка на 0).
        + '^' — піднесення до степеня.
        + '!^' — піднесення до степеня з позиційною зміною (друге число в степені першого).
        + 'ex' — вихід із програми.
      * Якщо введена некоректна операція, виводиться повідомлення про помилку.

Результат обчислення зберігається у змінній a, і виводиться оновлений результат.

Програма повертається до пункту 1, поки користувач не введе "ex".

**Основна програма:**

* + Виклик функції enter() для отримання першого числа.
  + Виклик функції operations() для виконання обчислень із подальшими числами.

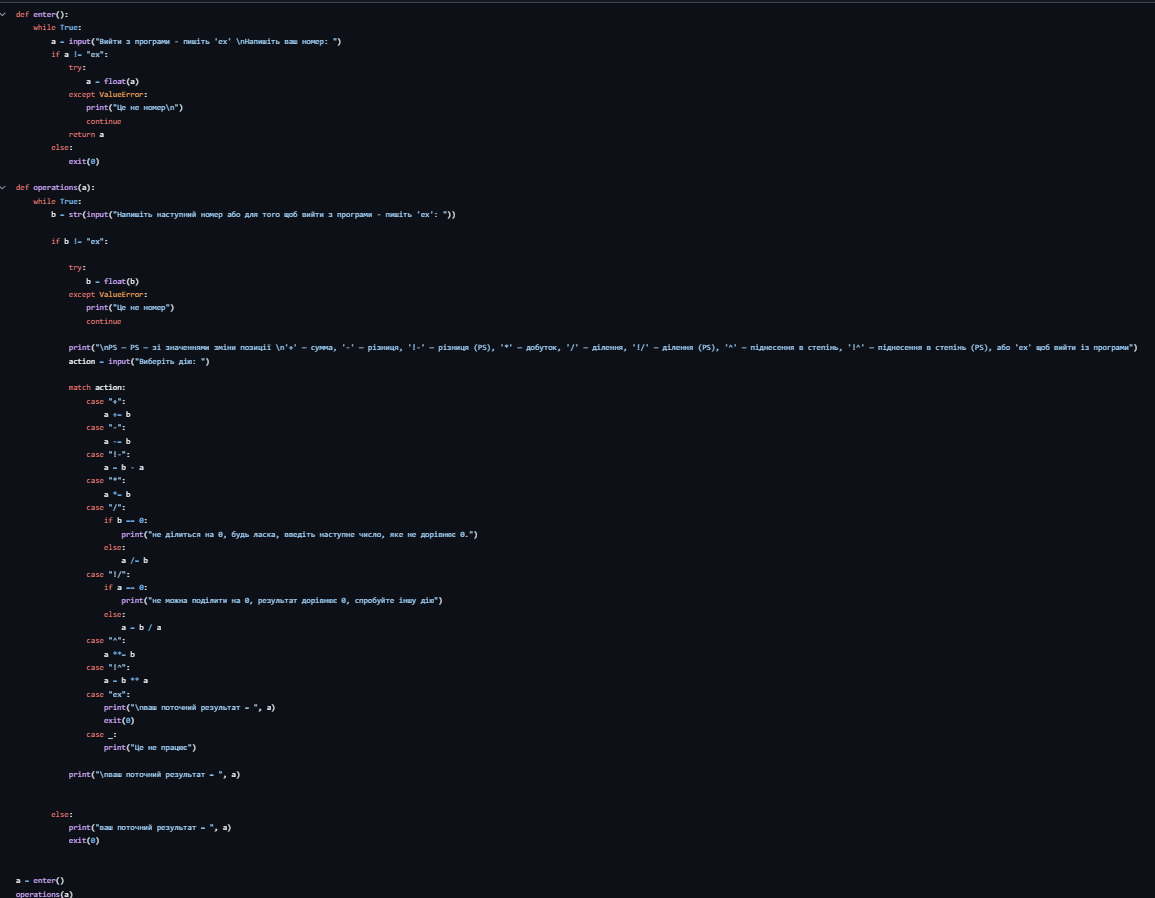
**Результат роботи програми:**

* + Програма дозволяє послідовно виконувати арифметичні дії над числами з можливістю виходу в будь-який момент.
  + У випадку помилок введення (некоректне число чи дія) користувач отримує повідомлення й може повторити спробу.

**Приклад роботи:**

* + Початковий запит: "Напишіть ваш номер:". Введено 5.
  + Наступний запит: "Напишіть наступний номер або для того щоб вийти з програми - пишіть 'ex':". Введено 3.
  + Обрано дію "+".
  + Вивід результату: "ваш поточний результат = 8".
  + Цикл продовжується, доки користувач не введе "ex".

|  |
| --- |
| def enter():      while True:          a = input("Вийти з програми - пишіть 'ex' \nНапишіть ваш номер: ")          if a != "ex":              try:                  a = float(a)              except ValueError:                  print("Це не номер\n")                  continue              return a          else:              exit(0)  def operations(a):      while True:          b = str(input("Напишіть наступний номер або для того щоб вийти з програми - пишіть 'ex': "))          if b != "ex":              try:                  b = float(b)              except ValueError:                  print("Це не номер")                  continue                print("\nPS — PS — зі значеннями зміни позиції \n'+' — сумма, '-' — різниця, '!-' — різниця (PS), '\*' — добуток, '/' — ділення, '!/' — ділення (PS), '^' — піднесення в степінь, '!^' — піднесення в степінь (PS), або 'ex' щоб вийти із програми")              action = input("Виберіть дію: ")              match action:                  case "+":                      a += b                  case "-":                      a -= b                  case "!-":                      a = b - a                  case "\*":                      a \*= b                  case "/":                      if b == 0:                          print("не ділиться на 0, будь ласка, введіть наступне число, яке не дорівнює 0.")                      else:                          a /= b                  case "!/":                      if a == 0:                          print("не можна поділити на 0, результат дорівнює 0, спробуйте іншу дію")                      else:                          a = b / a                  case "^":                      a \*\*= b                  case "!^":                      a = b \*\* a                  case "ex":                      print("\nваш поточний результат = ", a)                      exit(0)                  case \_:                      print("Це не працює")              print("\nваш поточний результат = ", a)          else:              print("ваш поточний результат = ", a)              exit(0)  a = enter()  operations(a) |

  
  
Посилання на github:

<https://github.com/k1ryushaaa/TP-KB-231-Yakovenko-Kyrylo>

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:

**Імпорт бібліотеки:**

Імпортується модуль random, щоб використовувати функцію random.randint() для випадкового вибору дій комп'ютера.

**Функція enter():**

Призначення: отримання вибору гравця.

**Алгоритм:**

* + - Користувачу пропонується вибрати один із варіантів:
      * 1 — камінь (stone),
      * 2 — ножиці (scissors),
      * 3 — папір (paper).
    - Перевіряється, чи введене значення є числом від 1 до 3:
      * Якщо так, повертається вибір гравця у вигляді індексу (0, 1 або 2) для подальшої обробки.
      * Інакше пропонується повторити введення.

**Функція play(p):**

Призначення: визначення результату гри на основі вибору гравця та комп'ютера.

**Алгоритм:**

* + - Комп'ютер робить випадковий вибір (comp) за допомогою random.randint(0, 2).
    - Використовується конструкція match-case для визначення результату:
      * Якщо вибори гравця та комп'ютера однакові, викликається функція draw().
      * У разі переваги одного вибору над іншим викликаються функції win() або loss().

**Функції для результатів:**

* + **win(p, comp):**
    - Виводить повідомлення про виграш, зазначаючи вибір гравця та комп'ютера.
  + **loss(p, comp):**
    - Виводить повідомлення про програш, зазначаючи вибір гравця та комп'ютера.
  + **draw(p):**
    - Виводить повідомлення про нічию, зазначаючи вибір гравця та комп'ютера.

**Список choice\_list:**

* + Містить текстові позначення варіантів: ["stone", "scissors", "paper"].
  + Використовується для відображення назв виборів у функціях win(), loss(), та draw().

**Основна програма:**

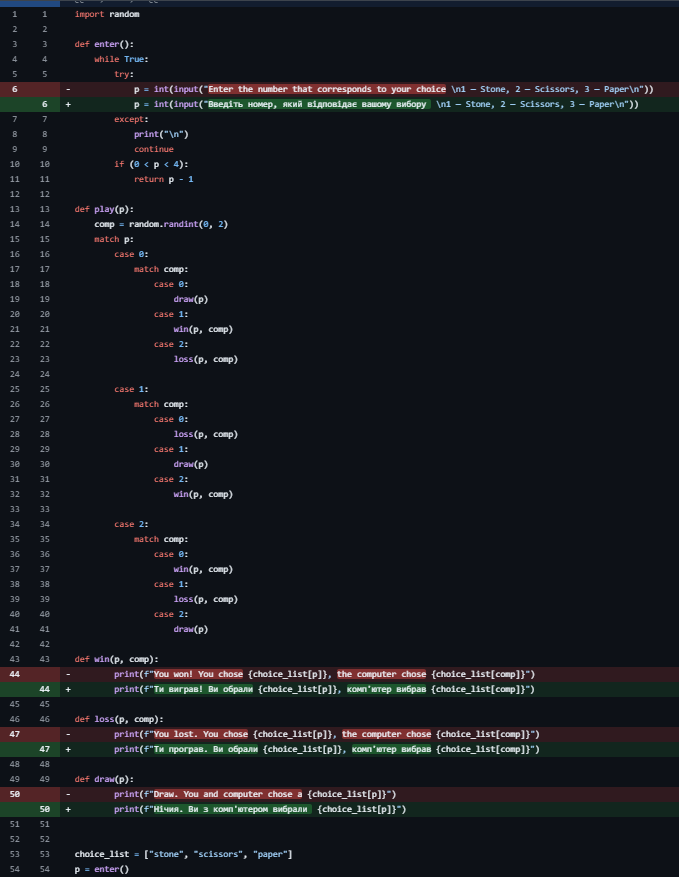
* + Викликається enter() для отримання вибору гравця.
  + Викликається play(p), яка обробляє логіку гри, враховуючи вибори гравця та комп'ютера.

**Приклад роботи:**

* + Гравець вводить 1 (камінь).
  + Комп'ютер випадково вибирає 2 (ножиці).
  + Викликається функція win(), яка виводить: Ти виграв! Ви обрали stone, комп'ютер вибрав scissors

Текст програми

|  |
| --- |
| import random  def enter():  while True:  try:  p = int(input("Введіть номер, який відповідає вашому вибору \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"Ти виграв! Ви обрали {choice\_list[p]}, комп'ютер вибрав {choice\_list[comp]}")  def loss(p, comp):  print(f"Ти програв. Ви обрали {choice\_list[p]}, комп'ютер вибрав {choice\_list[comp]}")  def draw(p):  print(f"Нічия. Ви з комп'ютером вибрали {choice\_list[p]}")  choice\_list = ["stone", "scissors", "paper"]  p = enter()  play(p) |



**Конвертор валюти**

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

Хід виконання завдання:

**Імпорт бібліотеки requests:**

Використовується для отримання актуальних даних про обмінний курс із сайту Національного банку України (НБУ).

**Запит до API НБУ:**

* Функція requests.get завантажує JSON-дані з API НБУ:https://bank.gov.ua/NBUStatService/v1/statdirectory/exchangenew?json
* Дані зберігаються в змінній nbu\_orig.

**Формування словника nbu:**

Перебираються отримані JSON-дані, щоб створити зручний словник, де:

* + - Ключі — коди валют (наприклад, USD, EUR).
    - Значення — словники з детальною інформацією про валюту (назва, курс тощо).

**Функції для роботи з валютами:**

* + **print\_cur\_n():**
    - Виводить список кодів валют та їхніх назв.
    - Наприклад: USD | Долар EUR | Євро
  + **print\_cur\_e():**
    - Виводить список кодів валют та їхніх курсів відносно гривні.
    - Наприклад: USD | 36.56 EUR | 40.12
  + **conv():**
    - Приймає код валюти та суму.
    - Переводить задану кількість вибраної валюти у гривні (UAH).
    - Використовує курс валюти з nbu.
    - У разі помилки (наприклад, неправильний код валюти) виводить повідомлення:
    - Неправильно вказано код валюти
  + **r\_conv():**
    - Приймає код валюти та суму в гривнях (UAH).
    - Переводить задану кількість гривень у вибрану валюту.
    - У разі помилки також повідомляє про неправильний код валюти.

**Функція action():**

Забезпечує вибір дій користувача:

* + - **pn** — виведення назв валют (print\_cur\_n()).
    - **pe** — виведення курсів валют (print\_cur\_e()).
    - **c** — конвертація валюти в гривні (conv()).
    - **rc** — зворотна конвертація (з гривні в іншу валюту, r\_conv()).
    - **ex** — вихід із програми.

**Робота основної програми:**

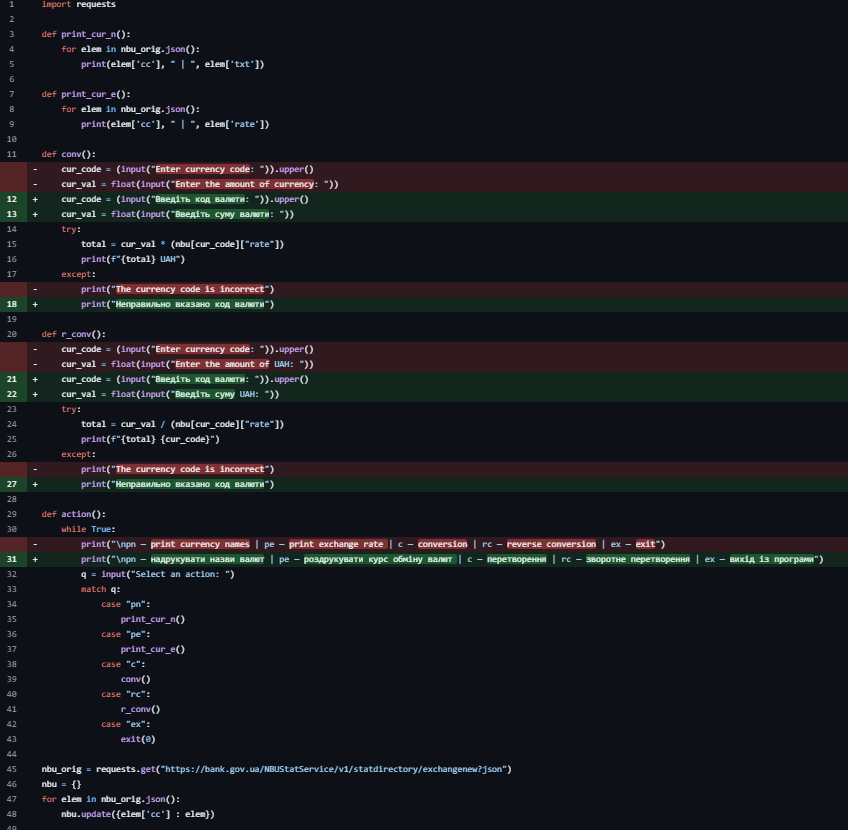
* + Після отримання даних про курси валют запускається функція action().
  + Програма пропонує користувачу вибрати дію та працює у циклі до вибору команди ex.

**Приклад роботи:**

* + Користувач вводить pn:
  + USD | Долар США
  + EUR | Євро
  + Користувач вводить c, обирає USD і вводить 100:
  + Введіть код валюти: USD
  + Введіть суму валюти: 100
  + 3656.0 UAH

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("Введіть код валюти: ")).upper()  cur\_val = float(input("Введіть суму валюти: "))  try:  total = cur\_val \* (nbu[cur\_code]["rate"])  print(f"{total} UAH")  except:  print("Неправильно вказано код валюти")  def r\_conv():  cur\_code = (input("Введіть код валюти: ")).upper()  cur\_val = float(input("Введіть суму UAH: "))  try:  total = cur\_val / (nbu[cur\_code]["rate"])  print(f"{total} {cur\_code}")  except:  print("Неправильно вказано код валюти")  def action():  while True:  print("\npn — надрукувати назви валют | pe — роздрукувати курс обміну валют | с — перетворення | rс — зворотне перетворення | ex — вихід із програми")  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() |



**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:

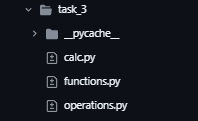
|  |
| --- |
| from operations import enter, actions  from functions import sum, sub, rsub, mult, div, rdiv, deg, rdeg  def calculation(a):      while True:          b = enter()          act = actions(a)          match act:              case "+":                  a = sum(a, b)              case "-":                  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("\nваш поточний результат = ", 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 "не ділиться на 0, будь ласка, введіть наступне число, яке не дорівнює 0."      else:          return a / b  def rdiv(a, b):      if a == 0:          return "не можна поділити на 0, тепер результат дорівнює 0, спробуйте іншу дію"      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("Для виходу із програми — напишіть 'ex' \nНапишіть ваш номер: ")          if a != "ex":              try:                  a = float(a)              except ValueError:                  print("Це не цифра\n")                  continue              return a          else:              exit(0)  def actions(a):      print("\nPS — з обміном позицій значень \n'+' — сумма, '-' — різниця, '!-' — різниця (PS), '\*' — добуток, '/' — ділення, '!/' — ділення (PS), '^' — піднесення в степінь, '!^' — піднесення в степінь (PS), або 'ex' для виходу із програми")      action = input("Виберіть дію: ")      if action == "ex":          print("\nваш поточний результат = ", a)          exit(0)      elif action in ["+", "-", "!-", "\*", "/", "!/", "^", "!^"]:          return action      else:          print("Це не працює") |

  
Посилання на github:

<https://github.com/k1ryushaaa/TP-KB-231-Yakovenko-Kyrylo>