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

**Topic\_01**

**Функції та змінні**

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

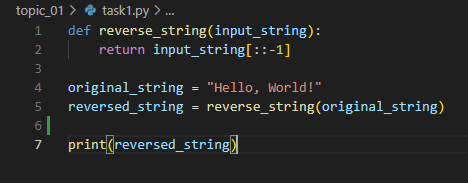
Завдання 1:

Треба за допомогою коду написати “Hello World!” навпаки.

Пояснення коду:

1. **def reverse\_string(input\_string):** - Це ключове слово **def** визначає початок функції. **reverse\_string** - це ім'я функції, яка приймає один аргумент **input\_string**.
2. **return input\_string[::-1]** - Це тіло функції. **input\_string** - це рядок, переданий функції як аргумент. Вираз **[:: -1]** використовує зріз, який повертає реверсовану версію рядка. Коли ви використовуєте **[:: -1]**, ви отримуєте копію рядка, але в зворотному порядку. Фактично, цей вираз забезпечує реверсування рядка.
3. **original\_string = "Hello, World!"** - Це створення змінної **original\_string**, якій присвоєне значення рядка "Hello, World!". Цей рядок буде використаний для виклику функції **reverse\_string**.
4. **reversed\_string = reverse\_string(original\_string)** - Це виклик функції **reverse\_string** з аргументом **original\_string**. Вона приймає рядок **original\_string**, виконує реверсування цього рядка за допомогою визначеної функції **reverse\_string** і зберігає результат у змінній **reversed\_string**.
5. **print(reversed\_string)** - Цей рядок виводить реверсований рядок **reversed\_string** на екран консолі за допомогою функції **print()**.

Скріншот коду:



Завдання 2:

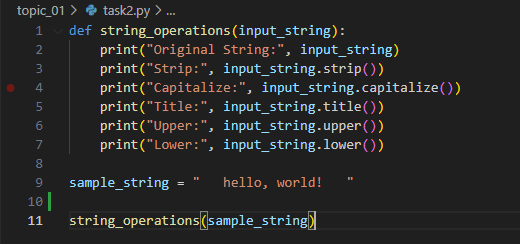
Виконати тестування функцій, що працюють з рядками: strip(), capitalize(), title(), upper(), lower().

Пояснення коду:

1. **def string\_operations(input\_string):** - Це оголошення функції **string\_operations**, яка приймає один аргумент **input\_string**.
2. **print("Original String:", input\_string)** - Це виведення на екран оригінального рядка, переданого у функцію.
3. **print("Strip:", input\_string.strip())** - Це виведення на екран результату виклику методу **strip()**, який видаляє всі пробіли та символи переносу рядка з початку та кінця вхідного рядка **input\_string**.
4. **print("Capitalize:", input\_string.capitalize())** - Це виведення на екран рядка, у якому перший символ перетворюється на верхній регістр, а всі інші - на нижній за допомогою методу **capitalize()**.
5. **print("Title:", input\_string.title())** - Це виведення на екран рядка, у якому кожне слово починається з великої літери за допомогою методу **title()**.
6. **print("Upper:", input\_string.upper())** - Це виведення на екран рядка, у якому всі символи переведені до верхнього регістру за допомогою методу **upper()**.
7. **print("Lower:", input\_string.lower())** - Це виведення на екран рядка, у якому всі символи переведені до нижнього регістру за допомогою методу **lower()**.

Отже, ця функція **string\_operations** виконує різні операції над вхідним рядком **input\_string** та виводить результати цих операцій на екран консолі.

Скріншот коду:



Завдання 3:

Написати функцію пошуку дискримінанту квадратного рівняння.

Пояснення коду:

Цей код має функцію `quadratic\_discriminant`, яка обчислює дискримінант квадратного рівняння за формулою. Дискримінант обчислюється для квадратного рівняння виду \(ax^2 + bx + c = 0\), де \(a\), \(b\) і \(c\) є коефіцієнтами рівняння.

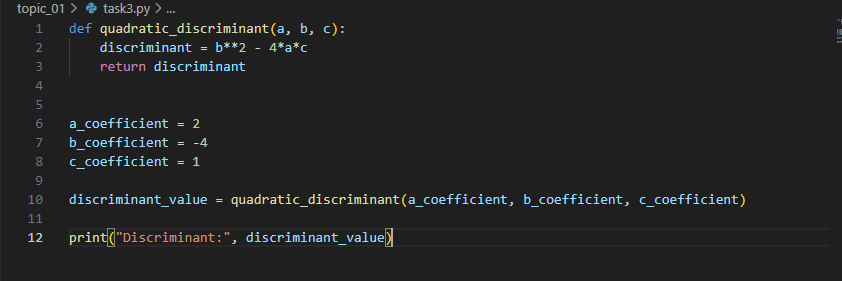
1. У функції `quadratic\_discriminant(a, b, c)`, квадрат дискримінанта обчислюється за формулою \(b^2 - 4ac\) та повертається як результат виконання функції.

2. У вашому коді ви використовуєте цю функцію, передаючи коефіцієнти квадратного рівняння \(a = 2\), \(b = -4\) і \(c = 1\) у якості аргументів для обчислення дискримінанта. Результат обчислення дискримінанта зберігається у змінній `discriminant\_value`.

3. На останньому рядку коду виводиться значення дискримінанта на екран консолі за допомогою функції `print()`.

Отже, цей код використовує функцію `quadratic\_discriminant` для обчислення дискримінанта квадратного рівняння з визначеними коефіцієнтами \(a\), \(b\) і \(c\), а потім виводить результат обчислення на екран.

Скріншот коду:



**Topic\_02**

Завдання 1:

Написати функцію пошуку коренів квадратного рівняння використовуючи функцію розрахунку дискримінанту з попередньої теми та умовні переходи.

Пояснення коду:

def quadratic\_roots(a, b, c):

# Вычисляем дискриминант

discriminant = b\*\*2 - 4\*a\*c

# Проверяем условия для различных случаев

if discriminant > 0: # Если дискриминант больше нуля, у уравнения два действительных корня

root1 = (-b + (discriminant)\*\*0.5) / (2\*a) # Вычисляем первый корень

root2 = (-b - (discriminant)\*\*0.5) / (2\*a) # Вычисляем второй корень

return root1, root2 # Возвращаем оба корня в виде кортежа

elif discriminant == 0: # Если дискриминант равен нулю, у уравнения один корень

root = -b / (2\*a) # Вычисляем корень

return root, # Возвращаем корень в виде кортежа с одним элементом

else: # Если дискриминант меньше нуля, у уравнения нет действительных корней

return "No real roots" # Возвращаем сообщение об отсутствии действительных корней

a = 1

b = -3

c = 2

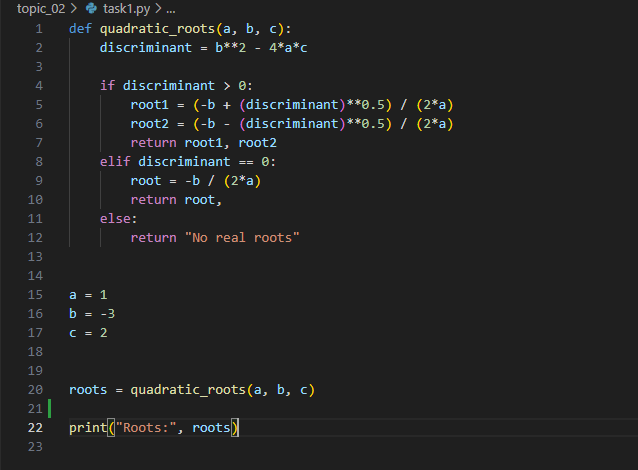
# Вызываем функцию quadratic\_roots с заданными коэффициентами

roots = quadratic\_roots(a, b, c)

# Выводим корни на экран

print("Roots:", roots)

Скріншот коду:



Завдання 2:

Написати програму калькулятор використовуючи **if else** конструкцію. Кожна операція має бути виконана в окремій функції.

Пояснення коду:

# Функція для додавання двох чисел

def add(x, y):

return x + y

**#** Функція для віднімання одного числа від іншого

def subtract(x, y):

return x - y

# Функція для множення двох чисел

def multiply(x, y):

return x \* y

# Функція для ділення одного числа на інше

def divide(x, y):

# Перевірка на нуль, щоб уникнути ділення на нуль

if y != 0:

return x / y

else:

return "Ділення на нуль не визначено" # Повідомлення про помилку при діленні на нуль

# Отримання введення користувача для першого числа

num1 = float(input("Введіть перше число: "))

# Отримання введення користувача для оператора (+, -, \*, /)

operator = input("Введіть оператор (+, -, \*, /): ")

# Отримання введення користувача для другого числа

num2 = float(input("Введіть друге число: "))

# Перевірка оператора та виконання відповідної операції

if operator == '+':

result = add(num1, num2) # Виклик функції додавання

elif operator == '-':

result = subtract(num1, num2) # Виклик функції віднімання

elif operator == '\*':

result = multiply(num1, num2) # Виклик функції множення

elif operator == '/':

result = divide(num1, num2) # Виклик функції ділення

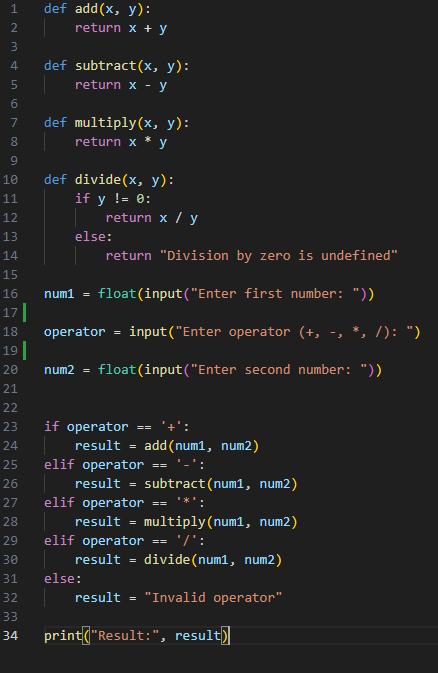
else:

result = "Невірний оператор" # Повідомлення про помилку при некоректному операторі

# Виведення результату операції на екран

print("Результат:", result)

Скріншот коду:



Завдання 3:

Написати програму калькулятор використовуючи **match** конструкцію. Кожна операція має бути виконана в окремій функції.

Пояснення коду:

def add(x, y):

return x + y

# Функція add(x, y) приймає два аргументи (x та y) і повертає їх суму.

def subtract(x, y):

return x - y

# Функція subtract(x, y) приймає два аргументи (x та y) і повертає їх різницю.

def multiply(x, y):

return x \* y

# Функція multiply(x, y) приймає два аргументи (x та y) і повертає їх добуток.

def divide(x, y):

if y != 0:

return x / y

else:

return "Division by zero is undefined"

# Функція divide(x, y) приймає два аргументи (x та y) і повертає їх частку,

# але перевіряє, чи друге число (y) не є нулем, щоб уникнути ділення на нуль.

def calculator():

print("Simple Calculator")

print("1. Add") # Вивід опції додавання

print("2. Subtract") # Вивід опції віднімання

print("3. Multiply") # Вивід опції множення

print("4. Divide") # Вивід опції ділення

choice = input("Enter choice (1/2/3/4): ")

# Запит на вибір операції від користувача.

num1 = float(input("Enter first number: "))

# Запит на введення першого числа від користувача

num2 = float(input("Enter second number: "))

# Запит на введення другого числа від користувача.

if choice == '1':

result = add(num1, num2)

print(f"{num1} + {num2} = {result}") # Вивід результату додавання

elif choice == '2':

result = subtract(num1, num2)

print(f"{num1} - {num2} = {result}") # Вивід результату віднімання

elif choice == '3':

result = multiply(num1, num2)

print(f"{num1} \* {num2} = {result}") # Вивід результату множення

elif choice == '4':

result = divide(num1, num2)

print(f"{num1} / {num2} = {result}") # Вивід результату ділення

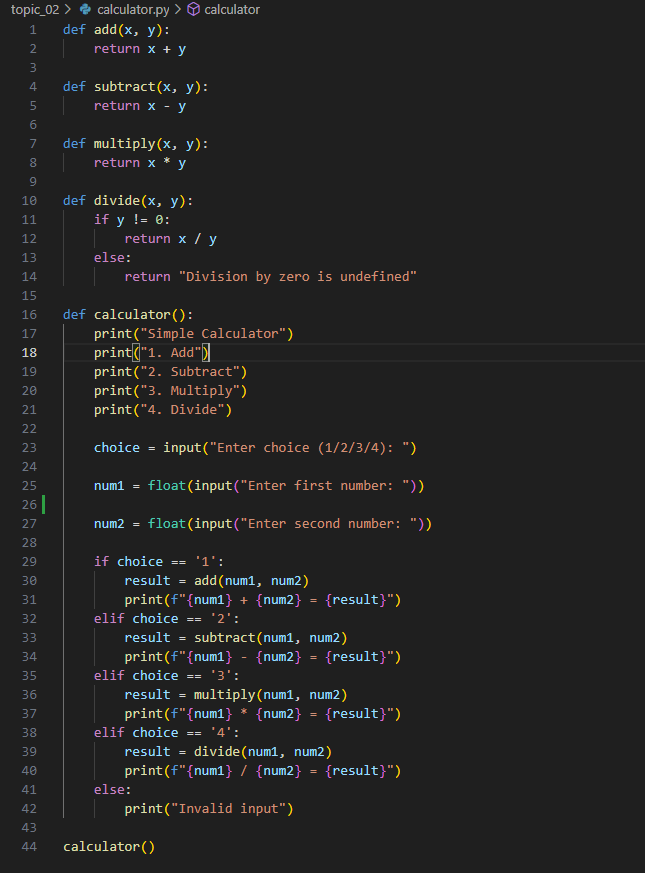
else:

print("Invalid input") # Повідомлення про невірний ввід

calculator()

# Виклик функції калькулятора для початку виконання програми.

Скріншот коду:



**Topic\_03**

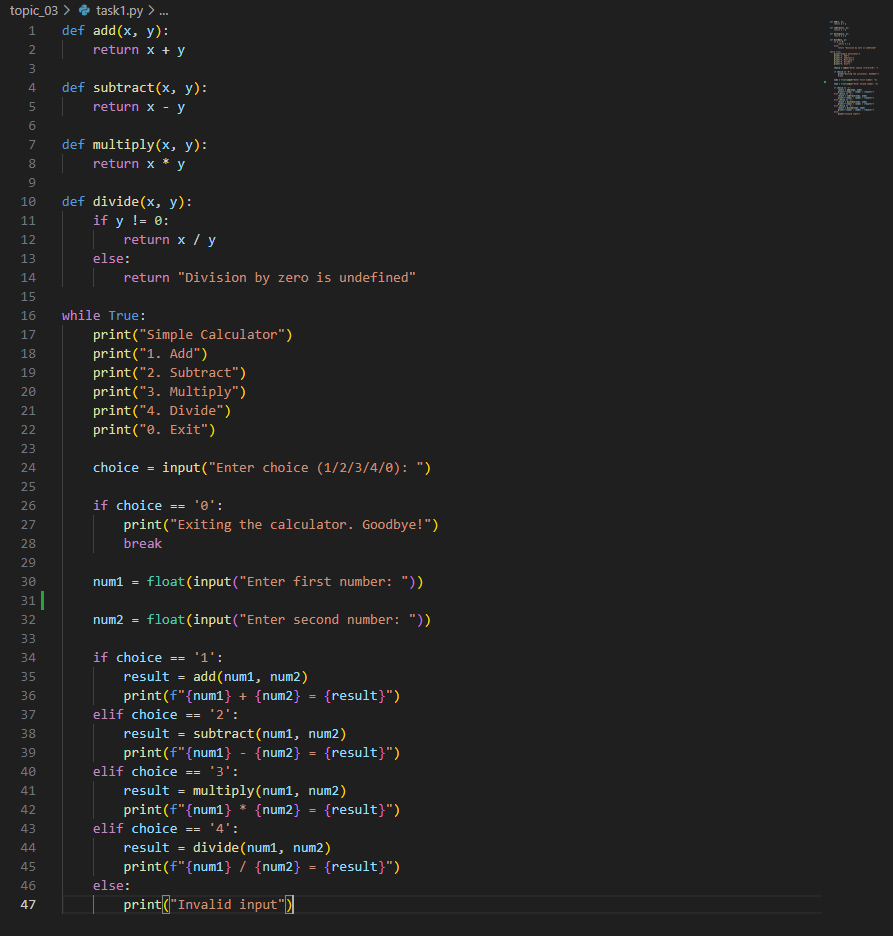
Завдання 1:

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.

Пояснення коду:

1. **Функції для математичних операцій**: **add**, **subtract**, **multiply**, **divide** виконують відповідні математичні операції над двома числами, які передаються їм у вигляді аргументів.
2. **Цикл while True**: Цей цикл ніколи не припиняється самостійно (без break). Він виводить меню калькулятора з опціями додавання, віднімання, множення та ділення, а також опцію виходу.
3. Користувач обирає операцію введенням відповідного числа: 1 для додавання, 2 для віднімання, 3 для множення, 4 для ділення, 0 для виходу.
4. Після вибору операції користувачем, програма просить ввести два числа.
5. Залежно від вибору користувача викликається відповідна функція для обраної операції. Результат обчислення виводиться на екран.
6. Якщо користувач вибере опцію виходу (0), програма виводить повідомлення "Exiting the calculator. Goodbye!" та завершує свою роботу.

Скріншот коду:



Завдання 2:

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

Пояснення коду:

my\_list = [1, 2, 3]

# Створення списку my\_list з елементами 1, 2, 3.

my\_list.extend([4, 5])

print(f"extend(): {my\_list}")

# Метод extend() використовується для розширення списку my\_list іншим списком [4, 5].

# Результат: [1, 2, 3, 4, 5]. Список my\_list містить ці п'ять елементів.

my\_list.append(6)

print(f"append(): {my\_list}")

# Метод append() додає елемент 6 в кінець списку my\_list.

# Результат: [1, 2, 3, 4, 5, 6].

my\_list.insert(1, 10)

print(f"insert(1, 10): {my\_list}")

# Метод insert() вставляє елемент 10 на позицію з індексом 1 (другий елемент) у списку my\_list.

# Результат: [1, 10, 2, 3, 4, 5, 6].

my\_list.remove(3)

print(f"remove(3): {my\_list}")

# Метод remove() видаляє перше входження значення 3 із списку my\_list.

# Результат: [1, 10, 2, 4, 5, 6].

my\_list.clear()

print(f"clear(): {my\_list}")

# Метод clear() видаляє всі елементи зі списку my\_list, залишаючи його порожнім.

# Результат: [].

my\_list = [4, 2, 1, 3]

my\_list.sort()

print(f"sort(): {my\_list}")

# Метод sort() сортує елементи у списку my\_list за зростанням.

# Результат: [1, 2, 3, 4].

my\_list.reverse()

print(f"reverse(): {my\_list}")

# Метод reverse() реверсує порядок елементів у списку my\_list.

# Результат: [4, 3, 2, 1].

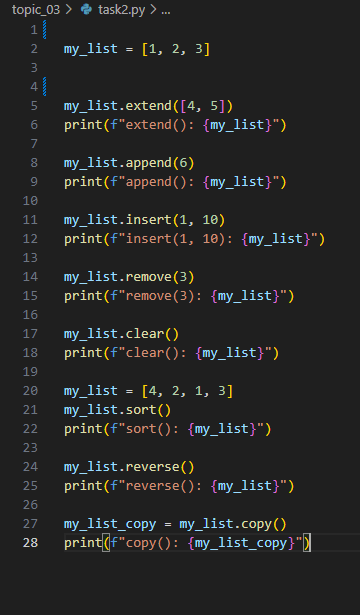
my\_list\_copy = my\_list.copy()

print(f"copy(): {my\_list\_copy}")

# Метод copy() створює поверхневу копію списку my\_list і присвоює його новій змінній my\_list\_copy.

# Результат: [4, 3, 2, 1]. Змінна my\_list\_copy тепер містить копію списку my\_list.

Скріншот коду:



Завдання 3:

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

Пояснення коду:

my\_dict = {'a': 1, 'b': 2, 'c': 3}

# Створення словника my\_dict з ключами 'a', 'b', 'c' та відповідними значеннями 1, 2, 3.

my\_dict.update({'d': 4, 'e': 5})

print(f"update(): {my\_dict}")

# Метод update() додає нові пари ключ-значення в словник my\_dict: 'd': 4, 'e': 5.

# Результат: {'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5}.

del my\_dict['a']

print(f"del('a'): {my\_dict}")

# Оператор del видаляє ключ 'a' та йому відповідне значення зі словника my\_dict.

# Результат: {'b': 2, 'c': 3, 'd': 4, 'e': 5}.

my\_dict.clear()

print(f"clear(): {my\_dict}")

# Метод clear() видаляє всі пари ключ-значення зі словника my\_dict, залишаючи його порожнім.

# Результат: {}.

my\_dict = {'a': 1, 'b': 2, 'c': 3}

print(f"keys(): {my\_dict.keys()}")

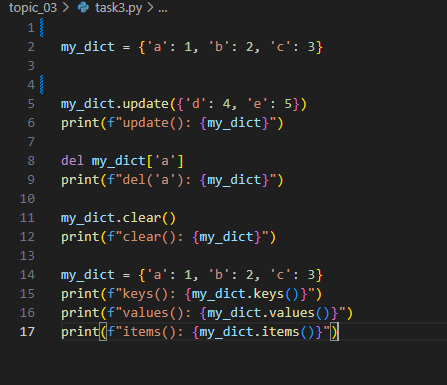
print(f"values(): {my\_dict.values()}")

print(f"items(): {my\_dict.items()}")

# Методи keys(), values() та items() повертають відповідно ключі, значення та пари ключ-значення

# словника my\_dict у вигляді об'єктів dict\_keys, dict\_values та dict\_items відповідно.

Скріншот коду:

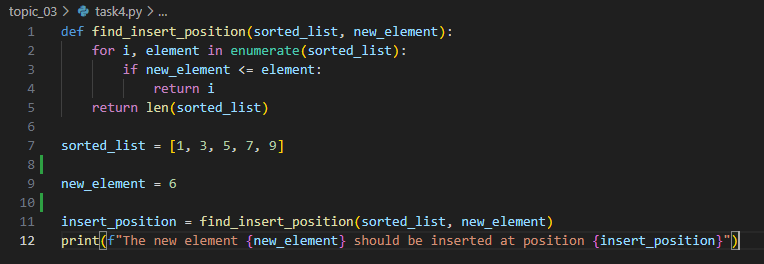


Завдання 4:

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

Пояснення коду:

* **find\_insert\_position(sorted\_list, new\_element)**: Це визначення функції **find\_insert\_position**, яка отримує відсортований список **sorted\_list** та новий елемент **new\_element**.
* **for i, element in enumerate(sorted\_list):**: Цикл перебирає елементи відсортованого списку **sorted\_list** разом з їхніми індексами **i**.
* **if new\_element <= element:**: Перевіряється, чи новий елемент менше або дорівнює поточному елементу списку.
* **return i**: Якщо новий елемент менше або дорівнює поточному елементу, повертається його позиція **i** у списку, де потрібно вставити новий елемент.
* **return len(sorted\_list)**: Якщо новий елемент більший за всі інші елементи у відсортованому списку, повертається довжина списку, що означає, що новий елемент потрібно вставити в кінець списку.
* **insert\_position = find\_insert\_position(sorted\_list, new\_element)**: Виклик функції **find\_insert\_position** для визначення позиції вставки нового елементу **new\_element** у відсортований список **sorted\_list**.
* **print(f"The new element {new\_element} should be inserted at position {insert\_position}")**: Виведення результату - позиції, на яку потрібно вставити новий елемент **new\_element** у відсортованому списку.

Скріншот коду:

**Topic\_04**

Завдання 1:

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

Пояснення коду:

while True:

print("Simple Calculator")

# Виведення меню операцій для користувача

choice = input("Enter choice (1/2/3/4/0): ")

# Зчитування вибору операції від користувача.

if choice == '0':

print("Exiting the calculator. Goodbye!")

break

# Якщо вибрано опцію виходу (0), програма завершує роботу.

try:

num1 = float(input("Enter first number: "))

num2 = float(input("Enter second number: "))

except ValueError:

print("Error: Invalid input. Please enter valid numbers.")

continue

# Зчитування двох чисел від користувача та перевірка на правильність введення.

if choice == '1':

result = add(num1, num2)

print(f"{num1} + {num2} = {result}")

elif choice == '2':

result = subtract(num1, num2)

print(f"{num1} - {num2} = {result}")

elif choice == '3':

result = multiply(num1, num2)

print(f"{num1} \* {num2} = {result}")

elif choice == '4':

result = divide(num1, num2)

if result is not None:

print(f"{num1} / {num2} = {result}")

else:

print("Invalid input")

* Цикл **while True** виконується безкінечно, доки користувач не вибере опцію виходу (0).
* Внутрішній блок **try-except** перевіряє правильність введення чисел. Якщо користувач введе нечислові дані, виводиться повідомлення про помилку.
* Після введення чисел та обрання операції, програма виконує відповідну операцію (додавання, віднімання, множення або ділення) та виводить результат. Якщо введена операція ділення, перевіряється, чи друге число не є нулем перед виконанням операції, щоб уникнути помилки ділення на нуль.

Скріншот коду:



**Topic\_05**

Завдання 1:

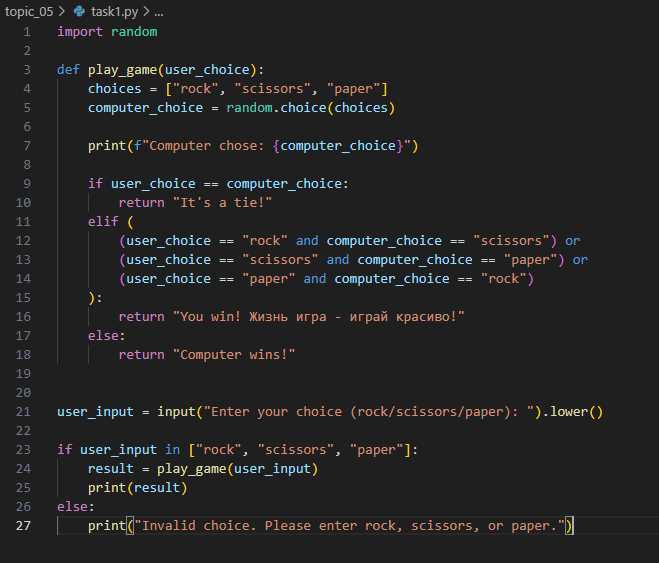
Гра з комп’ютером: камінь, ножиці, папір. Програма виконує запит від користувача на введення одного із значень ["rock", "scissor", "paper"]. Наступним кроком, використовуючи модуль random, програма у випадковому порядку вибирає одне із значень ["rock", "scissor", "paper"]. В залежності від умови, що камінь перемагає ножиці, ножиці перемагають папір, а папір перемагає камінь визначити переможця.

Пояснення коду:

Ця функція play\_game отримує вибір користувача у вигляді аргументу user\_choice. Комп'ютер також обирає випадковий варіант зі списку choices ("rock", "scissors", "paper"). Виконується порівняння вибору користувача та комп'ютера для визначення переможця чи ні.

Цей код запитує користувача ввести свій вибір (камінь, ножиці, папір), конвертує введення до нижнього регістру за допомогою lower(). Після того, як користувач введе допустимий варіант, викликається функція play\_game() з вибором користувача як аргументу. Якщо введено недопустимий варіант, виводиться повідомлення про недійсний вибір.

Скріншот коду:



Завдання 2:

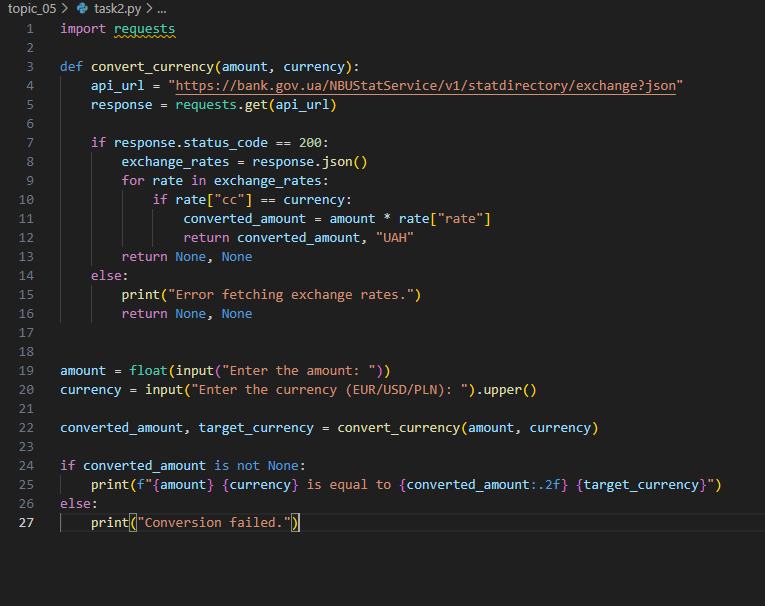
1. Програма конвертування іноземної валюти в українську гривню. Для отримання актуальних курсів валют необхідно використовувати API НБУ та модуль, що надає можливість виконувати запити до сторонніх сервісів requests. Достатня умова роботи – можливість конвертації для трьох іноземних валют EUR, USD, PLN. Користувачу надається можливість введення кількості та типу валюти, результат роботи програми – конвертоване значення в українських гривнях.

Пояснення коду:

Перевіряється статус відповіді: якщо статус 200 (OK), дані отримані успішно. Далі отримані дані у форматі JSON зберігаються у змінній exchange\_rates. Потім виконується пошук потрібного курсу валюти в цих даних. Якщо валюту знайдено, виконується обчислення конвертованої суми та повертається в результаті кортеж (converted\_amount, "UAH"). У випадку невдачі отримання даних або відсутності валюти у списку обмінних курсів, повертається None, None.

Користувач вводить суму та валюту. Функція convert\_currency викликається з введеними значеннями, і результат конвертації відображається у вигляді рядка. Якщо конвертація пройшла успішно, виводиться результат конвертації. У протилежному випадку виводиться повідомлення про невдачу конвертації.

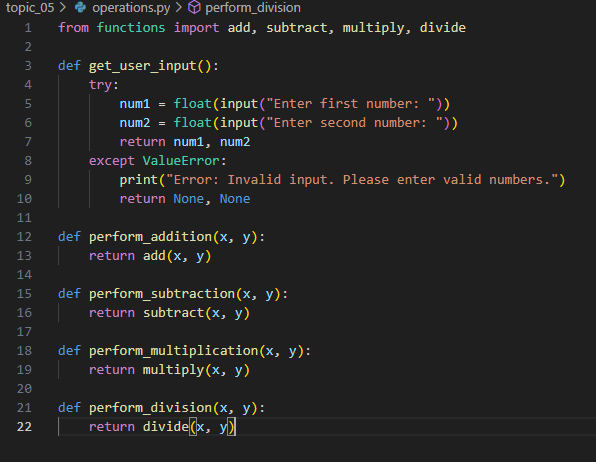
Скріншот коду:

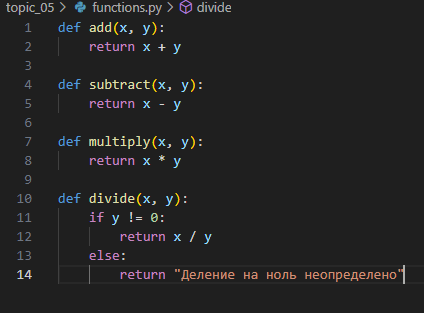


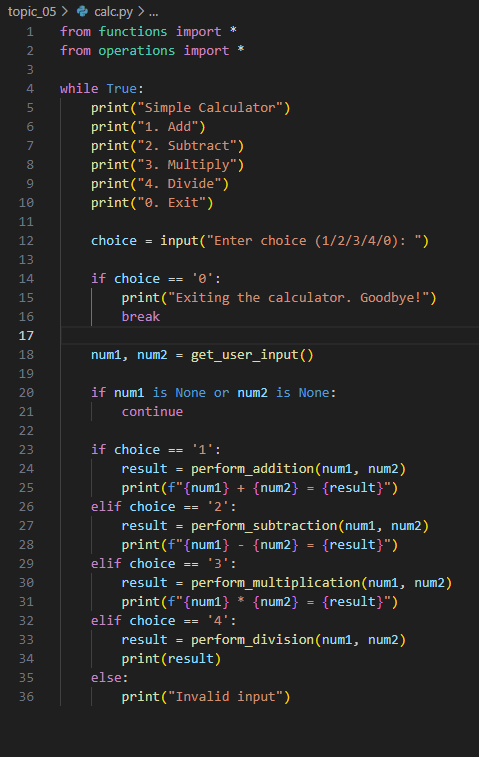
Завдання 3:

1. 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.

Скріншот коду:





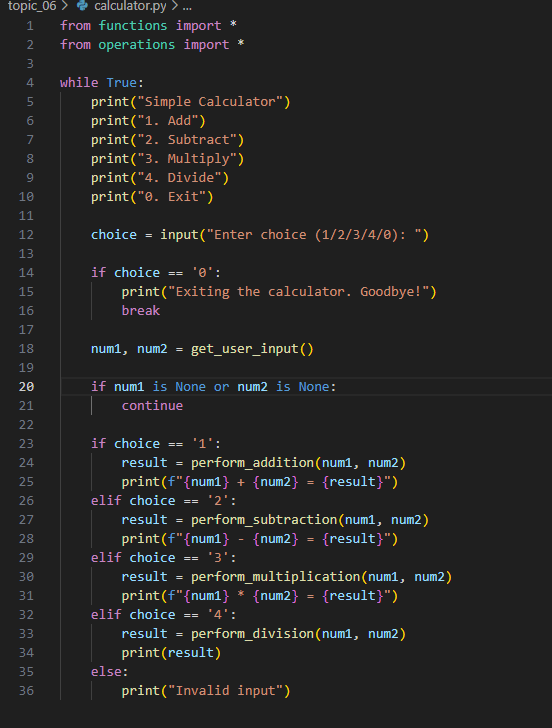


**Topic\_06**

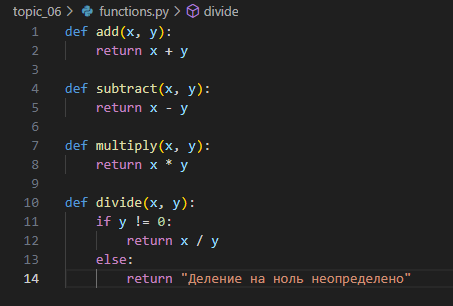
Завдання 1:

Розробити механізм логування всіх дій, що виконує програма. Забезпечити зберігання інформації про введені данні, виконану операцію та результат виконання операції над даними.

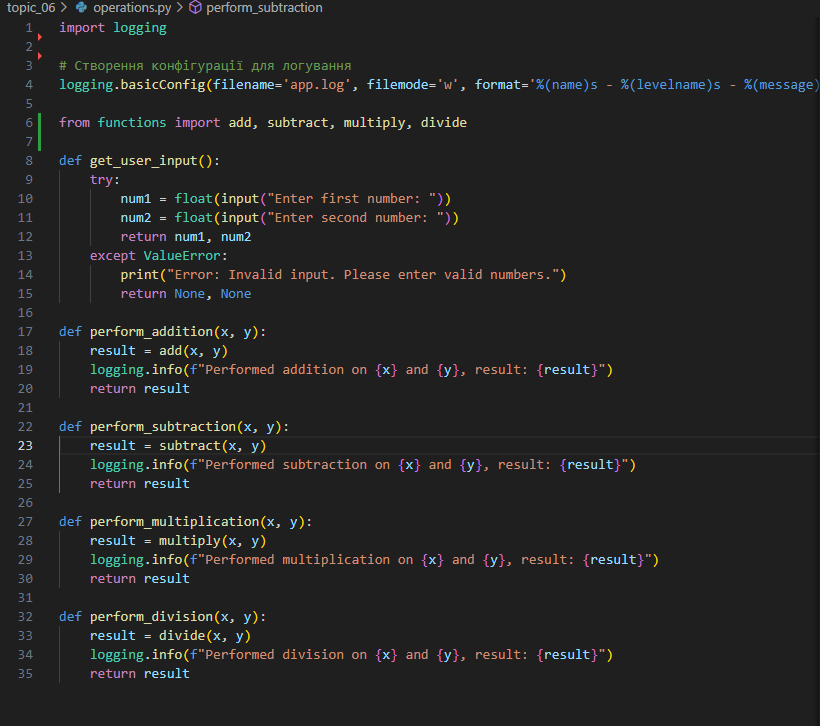
Calculator.py



Functions.py



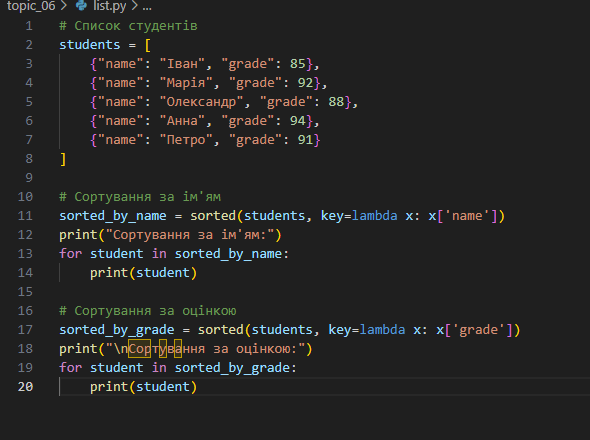
Operations.py



Завдання 2:

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.

Скріншот коду:



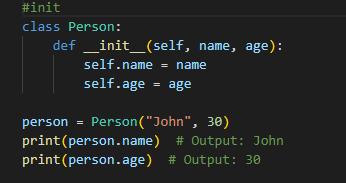
Висновок: В процесі виконання цього конкретного завдання я освоїв техніку реєстрації змін за допомогою лог-файлу і отримав розуміння суті лямбда-функцій, здобувши навички їх використання.

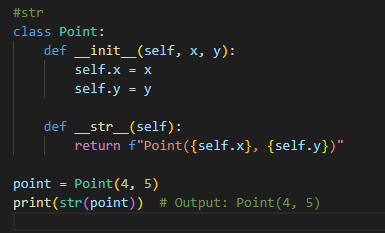
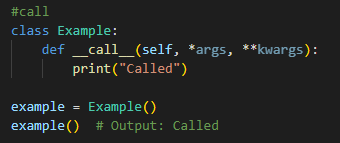
**Topic\_07**

Завдання 1:

У даному завданні потрібно було розібратись з декількома методами класу. Я розібрав три. Це init, str, call.

Ви згадали методи `init`, `str`, `call`, і я розповім вам, як вони працюють та як їх використовувати.

1.Class“init”  
 - Цей метод є конструктором класу і викликається при створенні нового об'єкта цього класу.  


2.Class “str”  
- Цей метод повертає рядок, який представляє об'єкт у зручному для читання форматі.  
  
  
3.Class “Call”  
-Завдяки цьому методу об'єкт можна викликати, якщо викликати його з дужками.  
-Він викликається у разі, коли ви викликаєте об'єкт, використовуючи дужки.  


Завдання 2: Створити клас "Студент", що містить два атрибути: " **name** " та " **age** ". Створити список, елементи якого є об'єктами класу "Студент". Написати цикл, який виводить на екран елементи списку у відсортованому порядку. Для сортування використати стандартну функцію "sorted". Під час використання функції "sorted" додати лямбда-функцію для визначення ключа сортування.

Скріншот коду:



Завдання 3: Використовуючи принципи ООП переписати програму Калькулятор. Завдання має бути виконано використовуючи модульний підхід.

Скріншот коду:

