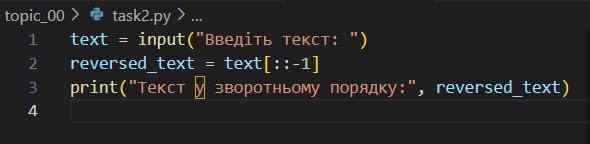
**Звіт про виконання практичних завдань до лекцій з курсу Технології програмування**

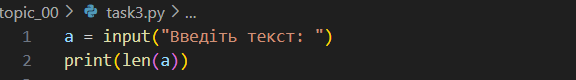
1. **Функції та змінні**

* Повернути рядок в зворотному порядку.

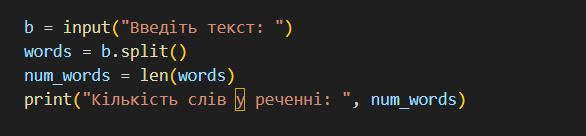


Змінна *text* запитує у користувача текст, який треба повернути в зворотному порядку. Далі створюємо функцію *reversed\_text*, яка буде обертати наш текст у зворотному порядку за допомогою *[:: -1],* що буде обирати елементи в зворотному порядку. За допомогою *print()* виводимо результат.

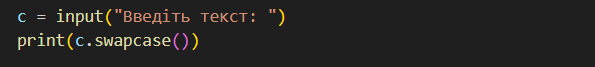
* Протестувати базові функції для рядків.



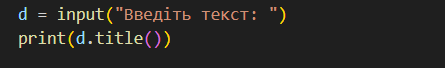
Змінна *а* запитує текст, функція *len()* рахує кількість літер в рядку і за допомогою функції *print()* виводить кількість літер, які ввів користувач.



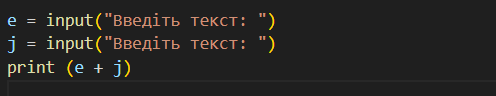
Змінна *b* запитує текст, вводимо змінну words, де функція *split()* буде розбивати введений текст на слова, використовуючи пробіл як спосіб їх розділяти. Змінна *num\_words()* буде викликати words щоб підрахувати їх кількість функцією *len()*.За допомогою *print* виводимо результат.



Змінна *с* запитує у користувача текст, функція *swapcase()* змінює кожну літеру в рядку, великі літери на малі, а малі - на великі, функцією *print()* виводимо результат.

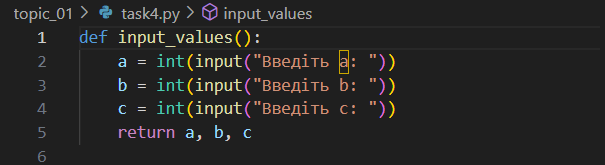


Змінна *d* запитує текст, функція *title()* змінює першу літеру в реченні на велику, функцією *print()* виводимо результат.

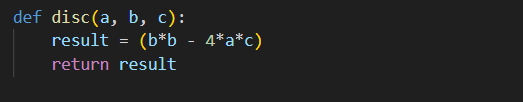


Змінні *e* та *j* запитують текст і в функції та в функції *print()* об’єднуються.

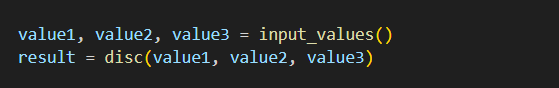
* Написати функцію пошуку Дискримінанту.



У функції *input\_values()* запитуємо числові змінні *a, b, c* та повертаємо значення.



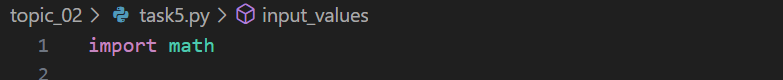
Функція *disc()* працюватиме з введеними значеннями та буде обчислювати дискримінант за формулою та повертати результат.



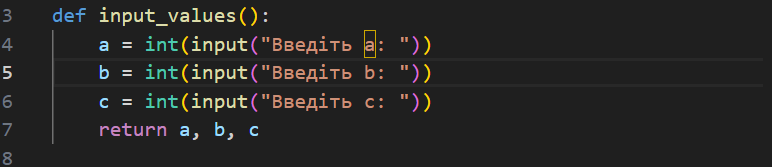
Змінні *value1, value2, value3* отримують значення *a, b, c.* Змінна *result* викликає функцію *disc()* для обчислення дискримінанту з отриманими значеннями *value1, value2, value3*.

1. **Умовні переходи.**

* Виконати пошук коренів квадратного рівняння враховуючи значення дискримінанту.



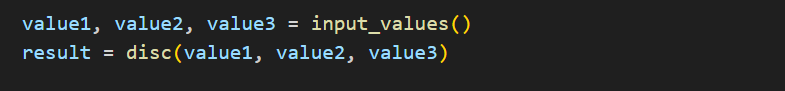
Додамо модуль *math*, він потрібен щоб використовувати знак кореню в обчисленнях.

\

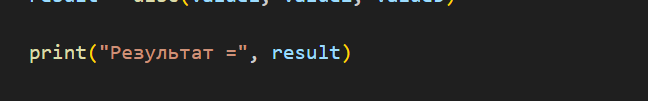
У функції *input\_values()* запитуємо числові змінні *a, b, c* та повертаємо значення.



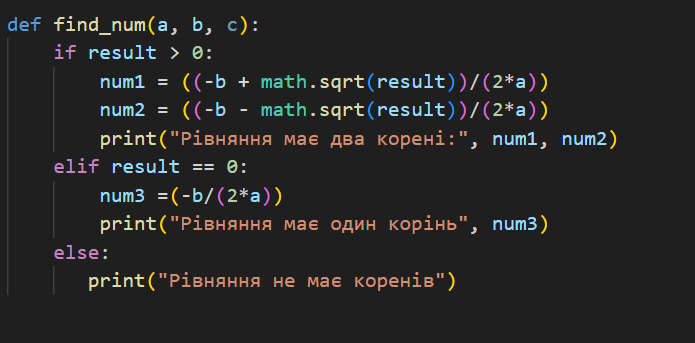
Функція *disc()* працюватиме з введеними значеннями та буде обчислювати дискримінант за формулою та повертати результат.



Змінні *value1, value2, value3* отримують значення a, b, c. Змінна *result* викликає функцію *disc()* для обчислення дискримінанту з отриманими значеннями *value1, value2, value3*.

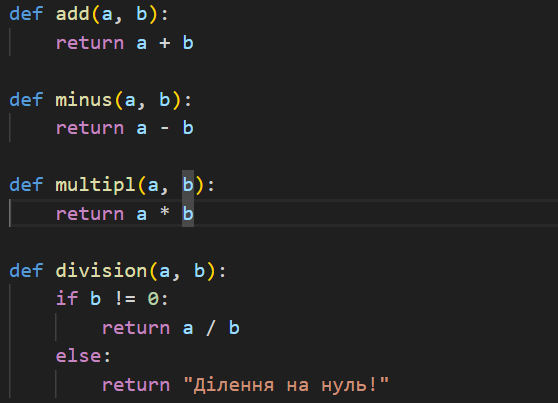


Функція *print()* виводитиме результат.



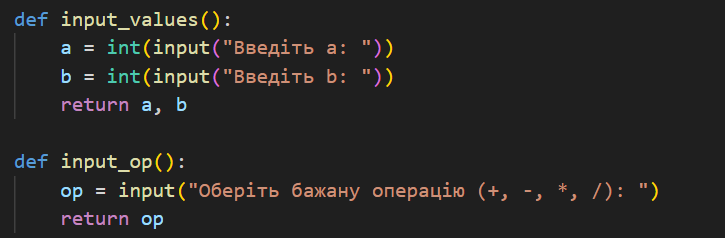
Створимо умову згідно теоремам, якщо результат більше 0, то корені 2 і задамо формулу для їхнього обчислення, якщо результат 0, то корінь 1, запишемо формулу для нього, в усіх інших випадках рівняння не має коренів.

* Програма калькулятор на основі *if elif else*.

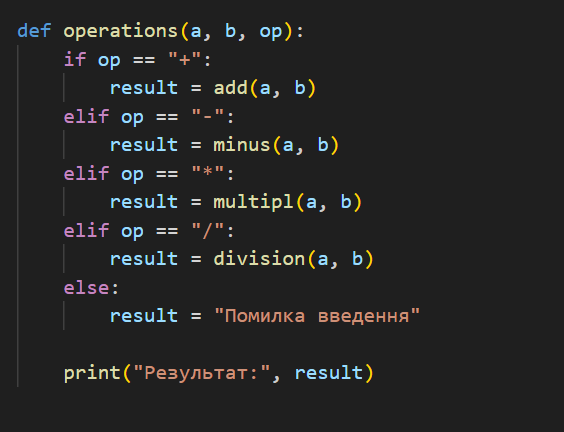


Задамо функціїї, які будуть означати дії з числами, які ми будемо викликати в процесі, додавання *(a + b)*, віднімання *(a – b)*, множення *(a \* b)*, ділення *(a / b)*.

Задамо через *if else*, що якщо *b* не дорівнює нулю, то виконується дія ділення, а якщо дорівнює, то буде виводитись попередження і результату не буде.

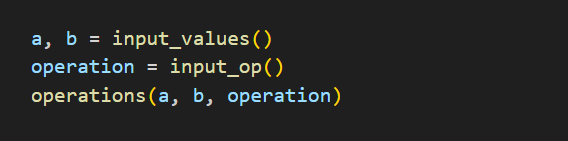


Введемо функцію, яка буде запитувати у користувача числові значення та бажану операцію з ними.



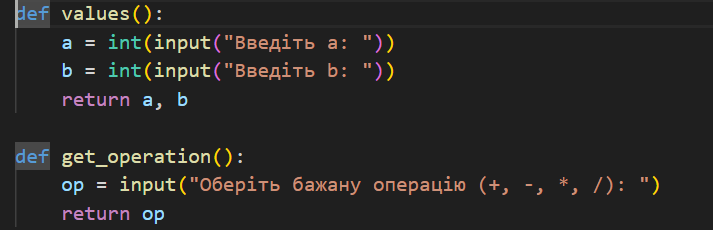
Перейдемо до умови *if elif* та *else*.

Задамо якщо операція +, то викликається функція з додаванням, через elif задамо операцію – з викликом функції віднімання, операцію \* з викликом функції множення та операцію / з викликом функції ділення. В усіх інших випадках буде помилка введення операції. Виведення результату.

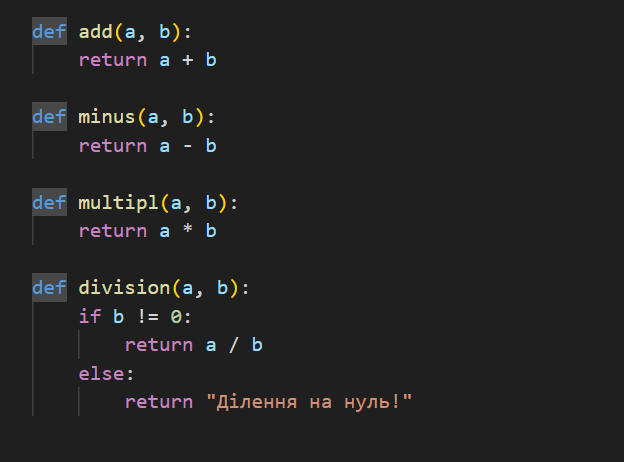


Виклик функцій.

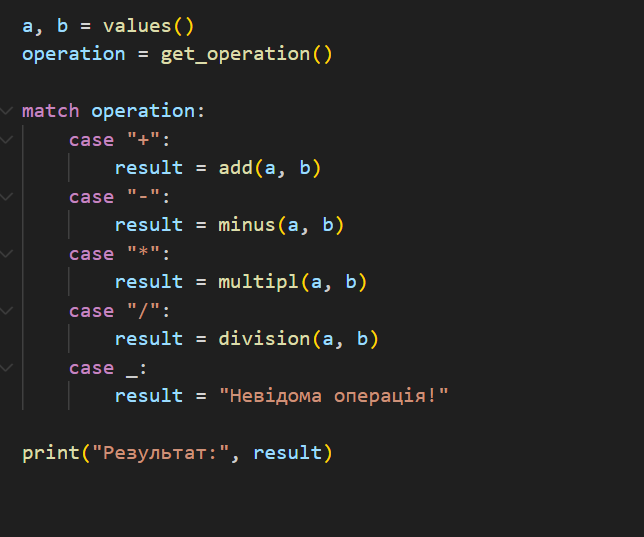
* Та сама програма калькулятор на основі оператора match.



Початок той самий що й в попередньому завданні, вводимо значення та операцію.



Функції дій з числами.

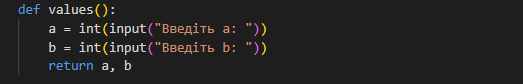


Викликаємо функцію *values()*, яка повертає значення змінних та функцію *get\_operation()*, яка повертає обрану користувачем операцію. Вводимо оператора match, яка перевірятиме значення змінної operation та відповідно якомусь з випадків код йтиме у підходящу для себе умову.

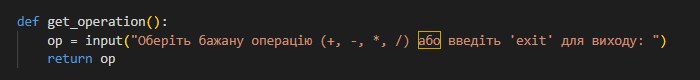
Якщо значення кейсу +, то викликається функція з додаванням, якщо кейс з – - з викликом функції віднімання, кейс з \* з викликом функції множення та кейс з / - з викликом функції ділення. В усіх інших випадках буде помилка введення операції. Виведення результату.

1. **Цикли**

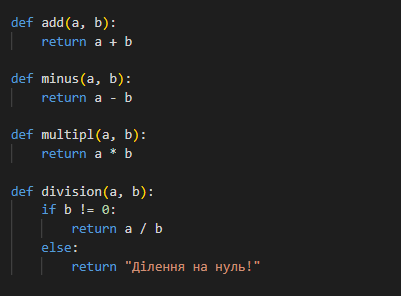
* Написати програму калькулятор з постійними запитами на введення нових даних та операцій.



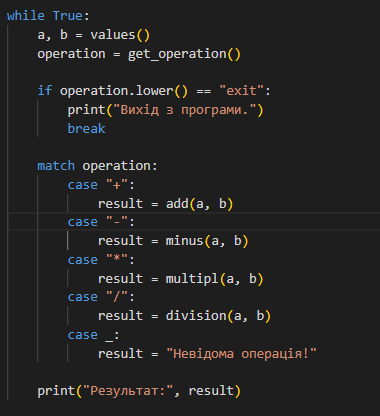
1. Функція def values() буде запитувати у користувача значення  a та b та повертатиме їх щоб далі з ними працювати.



1. Функція def get\_operation запитує одну з можливих операцій та повертатиме її

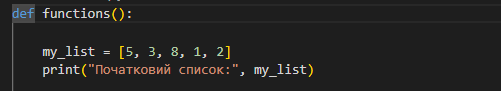


1. Функції обчислення, маємо 4 функції кожна з яких обчислюватиме свій знак обраний користувачем.



1. Цей цикл безкінечно запитує користувача для введення значень і операцій, доки не буде введено команду exit. Функція values має бути визначена, щоб отримати два числа, які користувач хоче використовувати для математичної операції. Функція get\_operation повинна запитати користувача ввести математичну операцію, яку він хоче виконати (наприклад, "+", "-", "\*", "/"). Якщо введена команда "exit", цикл завершується. В залежності від вибору операції застосовується певний кейс, але якщо операції не знайдено з наданих кейсів то вона буде помилкою.

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



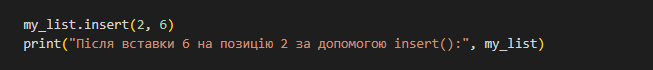
Усі тестувальні функції будуть в функції def functions. Я задала початковий список для того щоб з ним працювати надалі та показувати роботу функцій.



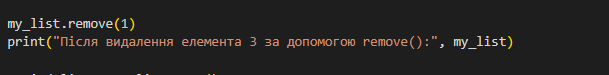
1. Функція append додає одиничний елемент в кінець списку.



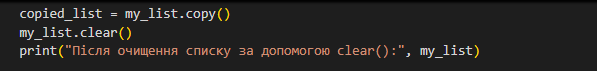
1. Функція extend додає елементи зі списку [4, 7] в кінець поточного списку.



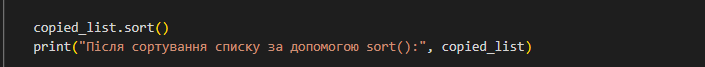
1. Функція insert вставляє елемент на задану позицію в списку.



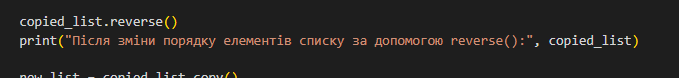
1. Функція видаляє заданий за номером елемент зі списку.



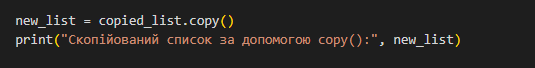
1. Функція copy створює копію списку my\_list та за допомогою clear очищає оригінальний список.



1. Функція sort сортує елементи списку в порядку зростання.

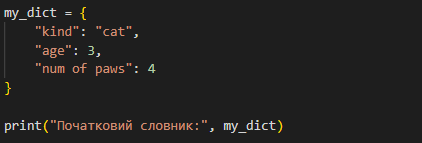


1. Функція reverse змінює порядок елементів списку на зворотний.



1. Функція new\_list = copied\_list.copy копіює список copied\_list в новий список new\_list.

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



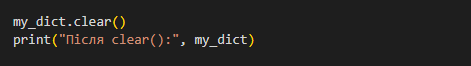
1. Створюється словник my\_dict та виводиться на консоль.



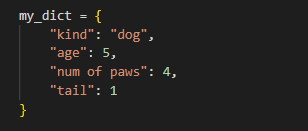
1. Після виконання update в словник додається новий елемент з ключем "tail" і значенням 1



1. Функція del видаляєт age. Після виконання del словник більше не міститиме пари "age": 3



1. Функція clear очищує словник.



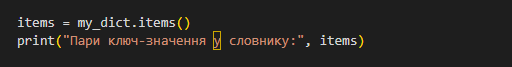
1. Наповнюю словник новими значеннями.



1. Функція keys повертає видимий список усіх ключів словника.

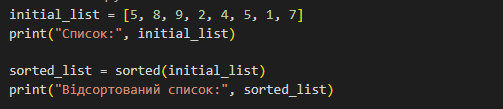


1. Функція values повертає значення словника.

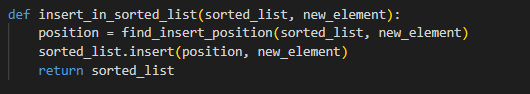


1. Функція items повертає пари ключ-значення.

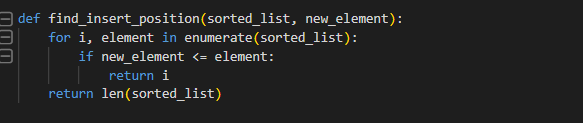
* Маючи відсортований список, написати функцію пошуку позиції для вставки нового елементу в список.



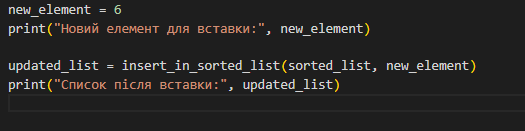
1. Спочатку створюєю список initial\_list з випадковими елементами. Використовую функція sorted, щоб створити новий список sorted\_list, в якому елементи відсортовані в порядку зростання.



1. Функція insert\_in\_sorted\_list вставляє новий елемент у відсортований список. Викликається функція find\_insert\_position, яка визначає правильну позицію для вставки нового елемента так, щоб список залишався відсортованим. Потім за допомогою методу insert елемент вставляється в знайдену позицію, і список повертається.

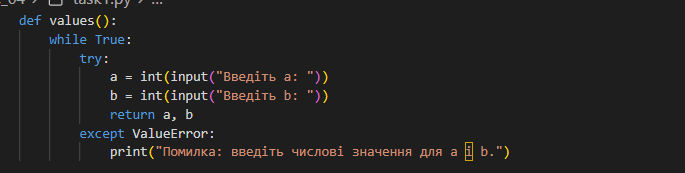


1. Функція find\_insert\_position шукає позицію, на яку потрібно вставити новий елемент, щоб зберегти відсортованість списку. За допомогою циклу for та методу enumerate перебираються всі елементи в списку sorted\_list. Якщо новий елемент менший або рівний поточному елементу, то новий елемент повинен бути вставлений на поточну позицію. Якщо жоден елемент не підходить, це означає, що новий елемент має бути вставлений в кінець списку.

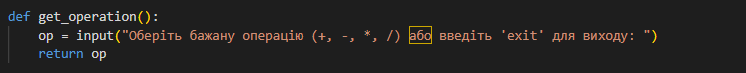


1. Новий елемент додається у відсортований список з пошуком потрібного місця щоб список залишався відсортованим.
2. **Виняткові ситуації**

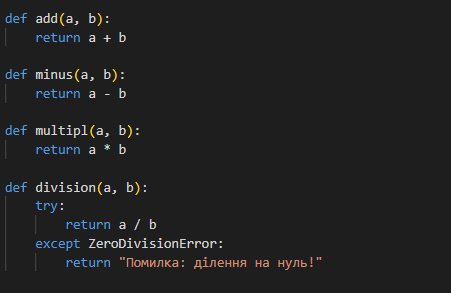
* Розширити програму калькулятор функцією запитів даних для виконання операцій від користувача, що обробляє виняткові ситуації.
* Розширити функцію ділення обробкою виняткової ситуації ділення но нуль



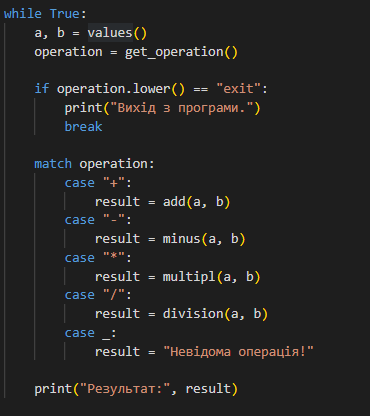
1. Програма запитує два числові значення в користувача, якщо вони будуть некоректрими то програма видасть помилку.



1. Функція запитує операцію яку ви хочете зробити з числами або пропонує вийти з циклу.



1. Функції виконання операцій, буде залежати від вибору користувача.

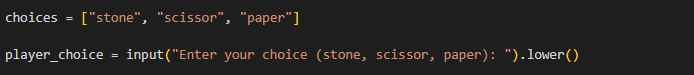


1. Основний цикл програми, який виконується до тих пір, поки користувач не введе exit.
2. **Бібліотеки**

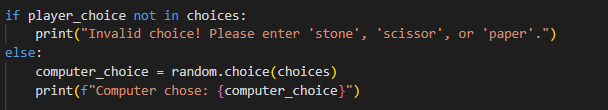
* Гра з комп’ютером: камінь, ножиці, папір. Програма виконує запит від користувача на введення одного із значень ["stone", "scissor", "paper"]. Наступним кроком, використовуючи модуль random, програма у випадковому порядку вибирає одне із значень ["stone", "scissor", "paper"]. В залежності від умови,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



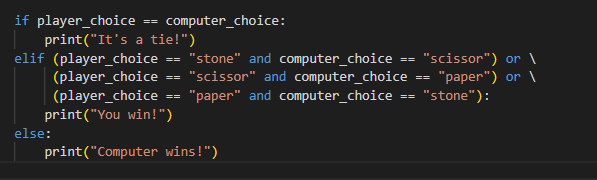
1. Імпортується модуль random, який використовується для генерації випадкового вибору комп'ютера.



1. Створюю список допустимих виборів камінь ножиці папір та запитую користувача про можливий вибір.



1. Якщо вибору гравця немає в списку то програма просить обрати щось зі списку, якщо все добре то комп’ютер генерує рандомний вибір зі списку.

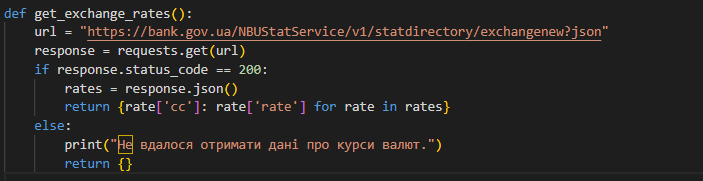


1. Якщо вибір комп’ютера і вибір гравця співпав, то буде нічия, камінь перемагає ножиці, ножиці перемагають папір, папір перемагає камінь. Якщо виконуються умови перемоги гравця, то він переміг, якщо ні – переміг комп’ютер.

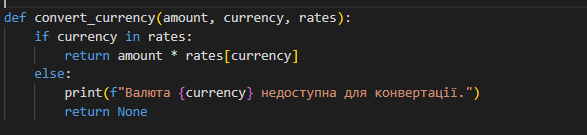
* Програма конвертування іноземної валюти в українську гривню. Для отримання актуальних курсів валют необхідно використовувати API НБУ та модуль, що надає можливість виконувати запити до сторонніх сервісів requests. Достатня умова роботи – можливість конвертації для трьох іноземних валют EUR, USD, PLN.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

****

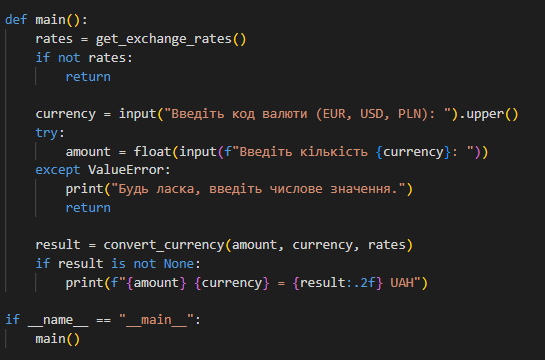
1. Імпорт модуля requests для виконання http запитів.



1. URL: Адреса API НБУ, яка повертає список курсів валют у форматі JSON. Функція requests.get(url): Відправляє GET-запит до API, якщо він успішний обробляється JSON-відповідь.



1. Amount – сума, яку потрібно конвертувати , currency – код валюти (наприклад, USD, EUR), rates – словник курсів валют. Якщо валюта доступна в rates, розраховується результат конвертації (amount \* rate). Якщо валюти немає в списку, виводиться повідомлення про помилку.



1. Викликається get\_exchange\_rates(). Якщо курсів немає (порожній словник), програма завершує роботу. Запитує код валюти у форматі ISO (наприклад, USD, EUR). Пробує зчитати суму для конвертації. У разі некоректного введення виводить помилку. Викликається convert\_currency. Якщо результат не None, виводиться конвертована сума у гривнях з округленням до 2 знаків після коми
2. Блок if \_\_name\_\_ == "\_\_main\_\_" забезпечує виконання лише якщо скрипт запущено напряму, а не імпортовано як модуль.

* 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.



Файл calc.py. Імпортує в себе функції з файлів functions.py та operations.py.