Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Лабораторна роботи №9

з дисципліни «Спеціалізовані мови програмування»

на тему

«Створення та рефакторінг програмно-інформаційного продукту засобами Python»

Виконав:

Костюк В. О.

Перевірив:

Щербак С. С.

Львів 2024

**Мета:** розробка програмно-інформаційного продукту засобами.

**План роботи**

Завдання 1. Створити скрипт запуску лабораторних робіт 1-8 (Runner) з єдиним меню для управління додатками використовуючи патерн FACADE https://refactoring.guru/uk/design-patterns/facade

Завдання 2. Зробити рефакторінг додатків, які були зроблені в лб 1-8, для підтримки можливості запуску через Runner

Завдання 3. Зробити рефакторинг додатків, які були зроблені в лб 1-8, використовуючи багаторівневу архітектуру додатків (див. приклад нижче) та всі принципи об’єктно-орієнтованого підходу

Завдання 4. Створити бібліотеку класів, які повторно використовуються у всіх лабораторних роботах та зробити рефакторінг додатків для підтримки цієї бібліотеки. Таких класів в бібліотеці має буде як найменш 5

Завдання 5. Додати логування функцій в класи бібліотеки програмного продукту використовуючи https://docs.python.org/uk/3/howto/logging.html

Завдання 6. Додати коментарі до програмного коду та сформувати документацію програмного продукту засобами pydoc. Документація має бути представлена у вигляді сторінок тексту на консолі, подана у веб-браузері та збережена у файлах HTML

Завдання 7. Документація та код програмного продукту має бути розміщено в GIT repo

Завдання 8. Проведіть статичний аналіз коду продукту засобами PYLINT https://pylint.readthedocs.io/en/stable/ та виправте помилки, які були ідентифіковані. Первинний репорт з помилками додайте до звіту лабораторної роботи

Завдання 9. Підготуйте звіт до лабораторной роботи

**Реалізація:**

**ЛР 1 – Введення в Python:**

**Мета:** створення консольної програми-калькулятора за допомогою основних синтаксичних конструкцій Python, з іншим завданням на заміну тестуванню та валідації.

**План роботи**

Завдання 1: Введення користувача

Створіть Python-програму, яка приймає введення користувача для двох чисел і оператора (наприклад, +, -, \*, /).

Завдання 2: Перевірка оператора

Перевірте чи введений оператор є дійсним (тобто одним із +, -, \*, /). Якщо ні, відобразіть повідомлення про помилку і попросіть користувача ввести дійсний оператор.

Завдання 3: Обчислення

Виконайте обчислення на основі введення користувача (наприклад, додавання, віднімання, множення, ділення) і відобразіть результат.

Завдання 4: Повторення обчислень

Запитайте користувача, чи він хоче виконати ще одне обчислення. Якщо так, дозвольте йому ввести нові числа і оператор. Якщо ні, вийдіть з програми.

Завдання 5: Обробка помилок

Реалізуйте обробку помилок для обробки ділення на нуль або інших потенційних помилок. Відобразіть відповідне повідомлення про помилку, якщо виникає помилка.

Завдання 6: Десяткові числа

Змініть калькулятор так, щоб він обробляв десяткові числа (плаваючу кому) для більш точних обчислень.

Завдання 7: Додаткові операції

Додайте підтримку додаткових операцій, таких як піднесення до степеня (^), квадратний корінь (√) і залишок від ділення (%).

Завдання 8: Функція пам'яті

Реалізуйте функцію пам'яті, яка дозволяє користувачам зберігати і відновлювати результати. Додайте можливості для зберігання та отримання значень з пам'яті.

Завдання 9: Історія обчислень

Створіть журнал, який зберігає історію попередніх обчислень, включаючи вираз і результат. Дозвольте користувачам переглядати історію своїх обчислень.

Завдання 10: Налаштування користувача

Надайте користувачам можливість налаштувати поведінку калькулятора, таку як зміну кількості десяткових розрядів, які відображаються, або налаштування функцій пам'яті.  
  
**Реалізація:**

**Файл runner.py:**

import sys

import unittest

from lab1.main import main as lab1\_main

from lab2.main import main as lab2\_main

from lab3.main import main as lab3\_main

from lab4.main import main as lab4\_main

from lab5.main import main as lab5\_main

from lab7.main import main as lab7\_main

from lab8.main import main as lab8\_main

class RunnerFacade:

    """

    Facade class to manage the execution of all laboratory works.

    """

    def \_\_init\_\_(self):

        # Словник з номером лабораторної роботи та відповідною функцією для запуску

        self.labs = {

            1: lab1\_main,

            2: lab2\_main,

            3: lab3\_main,

            4: lab4\_main,

            5: lab5\_main,

            6: self.run\_lab6\_tests,

            7: lab7\_main,

            8: lab8\_main,

        }

    def run\_lab(self, lab\_number):

        """

        Запускає лабораторну роботу за її номером.

        """

        if lab\_number in self.labs:

            print(f"\nЗапуск лабораторної роботи {lab\_number}...\n")

            try:

                self.labs[lab\_number]()  # Виклик функції main() для відповідного модуля

            except Exception as e:

                print(f"Помилка під час виконання лабораторної роботи {lab\_number}: {e}")

        else:

            print(f"Лабораторна робота {lab\_number} не знайдена.")

    @staticmethod

    def run\_lab6\_tests():

        """

        Запуск юніт-тестів для Лабораторної роботи 6.

        """

        print("\nЗапуск юніт-тестів для Лабораторної роботи 6...\n")

        try:

            # Вказуємо шлях до тестового файлу для Lab6

            tests = unittest.defaultTestLoader.discover('lab6', pattern='test\_\*.py')

            runner = unittest.TextTestRunner()

            runner.run(tests)

        except Exception as e:

            print(f"Помилка під час виконання юніт-тестів: {e}")

def display\_menu():

    """

    Display a menu for selecting the laboratory work to run.

    """

    print("\n=== Laboratory Work Runner ===")

    print("1. Run Lab 1")

    print("2. Run Lab 2")

    print("3. Run Lab 3")

    print("4. Run Lab 4")

    print("5. Run Lab 5")

    print("6. Run Lab 6 (Unit Tests)")

    print("7. Run Lab 7")

    print("8. Run Lab 8")

    print("0. Exit")

def main():

    """

    Main function to handle user input and invoke the corresponding lab work.

    """

    runner = RunnerFacade()

    while True:

        display\_menu()

        choice = input("Enter your choice: ").strip()

        if choice.isdigit():

            choice = int(choice)

            if choice == 0:

                print("Exiting the program.")

                sys.exit(0)

            else:

                runner.run\_lab(choice)

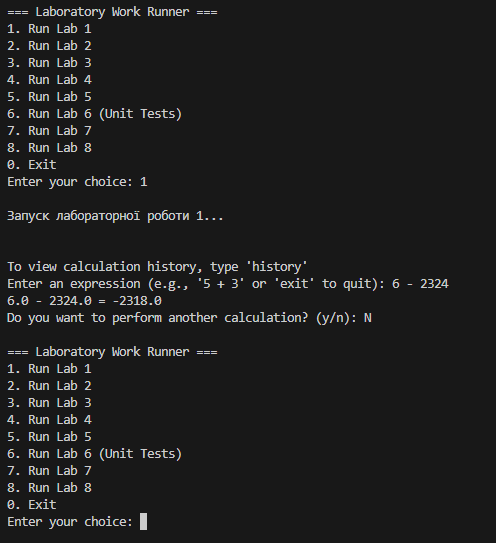
        else:

            print("Invalid input! Please enter a number from the menu.")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

**Результат виконання:**



**Висновок:** на цій лабораторній роботі я об’єднав і доопрацював всі лабораторні роботи в один великий проект, який працює через runner.py