Міністерство освіти і науки України Національний університет “Львівська політехніка” Кафедра інформаційних систем та мереж

ЗВІТ

про виконання лабораторної роботи № 2

***“ Основи побудови об’єктно-орієнтованих додатків на Python”***

з дисципліни “***Спеціальні мови програмування*** ”

Виконав студент групи IT-32

Пазюк С.А

Прийняв: Щербак С.С

# ЛЬВІВ – 2023

**Тема роботи:** Основи побудови об’єктно-орієнтованих додатків на Python

**Мета роботи:** Розробка консольного калькулятора в об’єктно орієнтованому стилі з використанням класів

**План роботи**

Завдання 1: Створення класу Calculator

Створіть клас Calculator, який буде служити основою для додатка калькулятора.

Завдання 2: Ініціалізація калькулятора

Реалізуйте метод \_\_init\_\_ у класі Calculator для ініціалізації необхідних атрибутів або змінних.

Завдання 3: Введення користувача

Перемістіть функціональність введення користувача в метод у межах класу Calculator. Метод повинен приймати введення для двох чисел і оператора.

Завдання 4: Перевірка оператора

Реалізуйте метод у класі Calculator, щоб перевірити, чи введений оператор є дійсним (тобто одним із +, -, \*, /). Відобразіть повідомлення про помилку, якщо він не є дійсним.

Завдання 5: Обчислення

Створіть метод у класі Calculator, який виконує обчислення на основі введення користувача (наприклад, додавання, віднімання, множення, ділення).

Завдання 6: Обробка помилок

Реалізуйте обробку помилок у межах класу Calculator для обробки ділення на нуль або інших потенційних помилок. Відобразіть відповідні повідомлення про помилку.

Завдання 7: Повторення обчислень

Додайте метод до класу Calculator, щоб запитати користувача, чи він хоче виконати ще одне обчислення. Якщо так, дозвольте йому ввести нові числа і оператор. Якщо ні, вийдіть з програми.

Завдання 8: Десяткові числа

Модифікуйте клас Calculator для обробки десяткових чисел (плаваюча кома) для більш точних обчислень.

Завдання 9: Додаткові операції

Розширте клас Calculator, щоб підтримувати додаткові операції, такі як піднесення до степеня (^), квадратний корінь (√) та залишок від ділення (%).

Завдання 10: Інтерфейс, зрозумілий для користувача

Покращте інтерфейс користувача у межах класу Calculator, надавши чіткі запити, повідомлення та форматування виводу для зручності читання.

**Хід роботи**

**Реалізований код:**

import math

# концепції

# 1-Класи і об'єкти: Я визначив клас Calculator, який містить властивості та методи для створення об'єктів калькулятора.

# Наприклад, calculator = Calculator(selected\_language) створює об'єкт calculator на основі класу Calculator.

# 2-Інкапсуляція: Я використав приватні властивості та методи класу Calculator, такі як self.result, self.history,

# інші методи та властивість self.language, для приховування деталей реалізації від користувача класу.

# Користувач має доступ лише до граничного інтерфейсу, який я визначив.

# 3-Наслідування: відсутні

# 4-Поліморфізм: У коді використовую метод calculate, який може виконувати різні операції з числами в залежності від оператора,

# введеного користувачем. Це можна розглядати як приклад поліморфізму, оскільки метод calculate працює з різними типами операцій

# Завдання 1: Створення класу Calculator

class Calculator:

# Завдання 2: Ініціалізація калькулятора

def \_\_init\_\_(self, language='ukrainian'):

self.result = None

self.history = []

self.language = language # Змінна language для вибору мови (за замовчуванням - українська).

# Метод для встановлення обраної мови.

def set\_language(self, language):

self.language = language

# Метод для отримання обраної мови.

def get\_language(self):

return self.language

# Завдання 10: Інтерфейс, зрозумілий для користувача

# Залежно від обраної мови повертаємо відповідний запит для введення виразу користувачем.

def get\_operator\_prompt(self):

if self.language == 'english':

return "Enter an expression (e.g., '2 + 2' or 'exit'): "

elif self.language == 'ukrainian':

return "Введіть вираз (наприклад, '2 + 2' або 'вихід'): "

else:

return "Enter an expression (e.g., '2 + 2' or 'exit'): "

# Завдання 7: Повторення обчислень

def get\_continue\_prompt(self):

if self.language == 'english':

return "Do you want to continue (yes/no)? "

elif self.language == 'ukrainian':

return "Бажаєте продовжити (так/ні)? "

else:

return "Do you want to continue (yes/no)? "

def calculate(self, num1, operator, num2):

try:

# Завдання 8: Десяткові числа

num1 = float(num1)

num2 = float(num2)

# Завдання 5: Обчислення

# Виконуємо обчислення на основі введених чисел та оператора.

if operator == '+':

self.result = num1 + num2

elif operator == '-':

self.result = num1 - num2

elif operator == '\*':

self.result = num1 \* num2

elif operator == '/':

# Завдання 6: Обробка помилок

# Обробка помилки ділення на нуль.

if num2 == 0:

return "Помилка: Не можна ділити на нуль!" if self.language == 'ukrainian' else "Error: Cannot divide by zero!"

self.result = num1 / num2

# Завдання 9: Додаткові операції

elif operator == '^':

self.result = num1 \*\* num2

elif operator == '√':

if num1 < 0:

return "Помилка: Від'ємне число під коренем!" if self.language == 'ukrainian' else "Error: Negative number under the square root!"

self.result = math.sqrt(num1)

elif operator == '%':

if num2 == 0:

return "Помилка: Ділення на нуль!" if self.language == 'ukrainian' else "Error: Division by zero!"

self.result = num1 % num2

else:

return "Помилка: Невірний оператор!" if self.language == 'ukrainian' else "Error: Invalid operator!"

return self.result

except ValueError:

# Завдання 6: Обробка помилок

# Обробка помилки невірного формату числа.

return "Помилка: Невірний формат числа!" if self.language == 'ukrainian' else "Error: Invalid number format!"

# Завдання 4: Перевірка оператора

def check\_operator(self, operator):

if operator in ('+', '-', '\*', '/'):

return True

else:

return False

# Зберігаємо вираз та результат в історію обчислень.

def add\_to\_history(self, expression, result):

self.history.append((expression, result))

def show\_history(self):

if not self.history:

# Завдання 10: Інтерфейс, зрозумілий для користувача

# Виводимо повідомлення, якщо історія порожня.

print("Історія обчислень порожня." if self.language == 'ukrainian' else "Calculation history is empty.")

else:

# Виводимо історію обчислень, якщо вона не порожня.

print("Історія обчислень:" if self.language == 'ukrainian' else "Calculation history:")

for idx, (expr, res) in enumerate(self.history, start=1):

print(f"{idx}. {expr} = {res}")

# Завдання 3: Введення користувача

# Запитуємо вираз користувача та повертаємо його.

def user\_input(self):

expression = input(self.get\_operator\_prompt())

return expression

def perform\_calculation(self, expression):

try:

num1, operator, num2 = map(str.strip, expression.split())

# Виконуємо обчислення та додаємо їх в історію.

if self.check\_operator(operator):

result = self.calculate(num1, operator, num2)

if isinstance(result, str):

print(result)

else:

print(f"Результат: {result}")

self.add\_to\_history(expression, result)

else:

# Завдання 6: Обробка помилок

# Повідомлення про помилку, якщо оператор невірний.

print("Помилка: Невірний оператор!" if self.language == 'ukrainian' else "Error: Invalid operator!")

except ValueError:

print("Помилка: Невірний формат виразу!" if self.language == 'ukrainian' else "Error: Invalid expression format!")

def main\_loop(self):

while True:

expression = self.user\_input()

if expression.lower() == 'вихід' or expression.lower() == 'exit':

break

self.perform\_calculation(expression)

self.show\_history()

next\_operation = input(self.get\_continue\_prompt())

if next\_operation.lower() != 'так' and next\_operation.lower() != 'yes':

break

print("Програма завершена." if self.language == 'ukrainian' else "Program terminated.")

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

print("Виберіть мову (українська або англійська):")

selected\_language = input("Choose a language (ukrainian or english): ").lower()

# Завдання 10: Інтерфейс, зрозумілий для користувача

while selected\_language not in ['ukrainian', 'english']:

print("Неправильно вибрана мова. Будь ласка, виберіть українську або англійську.")

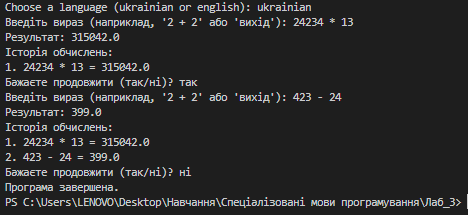
print("Invalid language selection. Please choose ukrainian or english.")

selected\_language = input("Choose a language (ukrainian or english): ").lower()

calculator = Calculator(selected\_language)

calculator.main\_loop()

**Результат виконання программи**:



**Висновок:** Виконавши ці завдання, я навчився перетворювати консольний калькулятор у об'єктно-орієнтований калькулятор, використовуючи класи в Python. Цей проект допоможе вам вивчити концепції об'єктно-орієнтованого програмування та організацію, зберігаючи функціональність і інтерфейс користувача калькулятора.