МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Кафедра інформаційних систем та мереж

   
 

Лабораторна робота №1

з курсу

Спеціалізовані мови програмування

   
 

Виконав студент

групи ІТ-21сп

**Гула В. М.**

Львів – 2023

Лабораторна робота № 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, +, -, \*, /).

**Код:**  
def main():

first\_num = float(input("Enter first number:\n"))

second\_number = float(input("Enter second number:\n"))

operator = input("Enter operator(+,-,\*,/):\n")

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

main()

**Результат:**

Структура:  
Зображення, що містить текст, знімок екрана, Шрифт, мультимедіа

Автоматично згенерований опис  
Виконання:  
Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис

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.

**Код:**  
def main():

first\_num = enter\_number("Enter first number:\n")

second\_number = enter\_number("Enter second number:\n")

operator = enter\_operator(['+', '-', '\*', '/'])

def enter\_operator(valid\_operators: list[str], input\_message: str = "Enter operator\n") -> str:

while True:

operator = input(input\_message)

if operator in valid\_operators:

return operator

else:

print("Invalid operator entered.")

print(f"Valid operators: {valid\_operators}")

def enter\_number(input\_msg: str = "Enter number:\n") -> float:

while True:

user\_input = input(input\_msg)

try:

number = float(user\_input)

return number

except ValueError:

print("Invalid number entered.")

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

main()

**Результат:**  
Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис

Завдання 3: Обчислення

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

**Код:**  
def main():

first\_num = enter\_number("Enter first number:\n")

second\_number = enter\_number("Enter second number:\n")

operator = enter\_operator(['+', '-', '\*', '/'])

result = do\_operation(first\_num, second\_number, operator)

print(f"Result: {result}")

def do\_operation(num1: float, num2: float, operator: str) -> float:

if operator == '+':

return num1 + num2

elif operator == '-':

return num1 - num2

elif operator == '\*':

return num1 \* num2

elif operator == '/':

return num1 / num2

else:

print("Invalid operator passed")

def enter\_operator(valid\_operators: list[str], input\_message: str = "Enter operator\n") -> str:

while True:

operator = input(input\_message)

if operator in valid\_operators:

return operator

else:

print("Invalid operator entered.")

print(f"Valid operators: {valid\_operators}")

def enter\_number(input\_msg: str = "Enter number:\n") -> float:

while True:

user\_input = input(input\_msg)

try:

number = float(user\_input)

return number

except ValueError:

print("Invalid number entered.")

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

main()

**Результат:**  
Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис

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.

**Код:**

main.py:  
from calculator import Calculator

def main():

flag = True

while flag:

print("Program options:\n1 - make calculation\n0 - exit")

option = int(input("Choose option:\n"))

if option == 1:

calculator = Calculator()

print(f"Result: {calculator.do\_operation()}")

elif option == 0:

flag = False

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

main()

calculator.py:  
class Calculator:

def \_\_init\_\_(self):

self.\_\_first\_number = Calculator.\_\_enter\_number("Enter first number\n")

self.\_\_second\_number = Calculator.\_\_enter\_number("Enter second number\n")

self.\_\_operator = Calculator.\_\_enter\_operator(['+', '-', '\*', '/'])

def do\_operation(self) -> float:

if self.\_\_operator == '+':

return self.\_\_first\_number + self.\_\_second\_number

elif self.\_\_operator == '-':

return self.\_\_first\_number - self.\_\_second\_number

elif self.\_\_operator == '\*':

return self.\_\_first\_number \* self.\_\_second\_number

elif self.\_\_operator == '/':

return self.\_\_first\_number / self.\_\_second\_number

else:

print("Invalid operator passed")

@staticmethod

def \_\_enter\_operator(valid\_operators: list[str], input\_message: str = "Enter operator\n") -> str:

while True:

operator = input(input\_message)

if operator in valid\_operators:

return operator

else:

print("Invalid operator entered.")

print(f"Valid operators: {valid\_operators}")

@staticmethod

def \_\_enter\_number(input\_msg: str = "Enter number:\n") -> float:

while True:

user\_input = input(input\_msg)

try:

number = float(user\_input)

return number

except ValueError:

print("Invalid number entered.")

**Результат:**

Структура:  
Зображення, що містить текст, знімок екрана, Шрифт, дизайн

Автоматично згенерований опис

Виконання:  
Зображення, що містить текст, знімок екрана, програмне забезпечення

Автоматично згенерований опис

Завдання 5: Обробка помилок

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

**Код:**

main.py:  
from calculator import Calculator

def main():

flag = True

while flag:

print("Program options:\n1 - make calculation\n0 - exit")

option = int(input("Choose option:\n"))

if option == 1:

calculator = Calculator()

print(f"Result: {calculator.do\_operation()}")

elif option == 0:

flag = False

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

main()

calculator.py:  
class Calculator:

def \_\_init\_\_(self):

self.\_\_first\_number = Calculator.\_\_enter\_number("Enter first number\n")

self.\_\_second\_number = Calculator.\_\_enter\_number("Enter second number\n")

self.\_\_operator = Calculator.\_\_enter\_operator(['+', '-', '\*', '/'])

def do\_operation(self) -> float:

if self.\_\_operator == '+':

return self.\_\_first\_number + self.\_\_second\_number

elif self.\_\_operator == '-':

return self.\_\_first\_number - self.\_\_second\_number

elif self.\_\_operator == '\*':

return self.\_\_first\_number \* self.\_\_second\_number

elif self.\_\_operator == '/':

if self.\_\_second\_number == 0:

print("Cannot divide by zero")

else:

return self.\_\_first\_number / self.\_\_second\_number

else:

print("Invalid operator passed")

@staticmethod

def \_\_enter\_operator(valid\_operators: list[str], input\_message: str = "Enter operator\n") -> str:

while True:

operator = input(input\_message)

if operator in valid\_operators:

return operator

else:

print("Invalid operator entered.")

print(f"Valid operators: {valid\_operators}")

@staticmethod

def \_\_enter\_number(input\_msg: str = "Enter number:\n") -> float:

while True:

user\_input = input(input\_msg)

try:

number = float(user\_input)

return number

except ValueError:

print("Invalid number entered.")

**Результат:**  
Зображення, що містить текст, знімок екрана, Шрифт, дизайн

Автоматично згенерований опис

Завдання 6: Десяткові числа

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

**Виконання:** код ніяк не змінюється, ця функціональність вже реалізована

Завдання 7: Додаткові операції

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

**Код:**

main.py – залишився без змін

calculator.py:  
import math

class Calculator:

def \_\_init\_\_(self):

self.\_\_first\_number = Calculator.\_\_enter\_number("Enter first number\n")

self.\_\_second\_number = Calculator.\_\_enter\_number("Enter second number\n")

valid\_operators = ['+', '-', '\*', '/', '^', '%', 'sqrt']

operator\_input\_message = f"Enter operator:\nValid operators:{valid\_operators}\n"

self.\_\_operator = Calculator.\_\_enter\_operator(valid\_operators, operator\_input\_message)

def do\_operation(self) -> float:

if self.\_\_operator == '+':

return self.\_\_first\_number + self.\_\_second\_number

elif self.\_\_operator == '-':

return self.\_\_first\_number - self.\_\_second\_number

elif self.\_\_operator == '\*':

return self.\_\_first\_number \* self.\_\_second\_number

elif self.\_\_operator == '/':

if self.\_\_second\_number == 0:

print("Cannot divide by zero")

else:

return self.\_\_first\_number / self.\_\_second\_number

elif self.\_\_operator == '^':

return pow(self.\_\_first\_number, self.\_\_second\_number)

elif self.\_\_operator == '%':

return self.\_\_first\_number % self.\_\_second\_number

elif self.\_\_operator == 'sqrt':

return self.get\_sqrt()

else:

print("Invalid operator passed")

def get\_sqrt(self) -> float:

print(f"Your numbers:\n1 - {self.\_\_first\_number}\n2 - {self.\_\_second\_number}")

option = int(input("Choose the number that should be converted?\n"))

if option == 1:

if self.\_\_first\_number < 0:

print(f"Cannot get square of {self.\_\_first\_number} cause it is < 0")

else:

return math.sqrt(self.\_\_first\_number)

if option == 2:

if self.\_\_second\_number < 0:

print(f"Cannot get square of {self.\_\_second\_number} cause it is < 0")

else:

return math.sqrt(self.\_\_second\_number)

@staticmethod

def \_\_enter\_operator(valid\_operators: list[str], input\_message: str = "Enter operator\n") -> str:

while True:

operator = input(input\_message)

if operator in valid\_operators:

return operator

else:

print("Invalid operator entered.")

print(f"Valid operators: {valid\_operators}")

@staticmethod

def \_\_enter\_number(input\_msg: str = "Enter number:\n") -> float:

while True:

user\_input = input(input\_msg)

try:

number = float(user\_input)

return number

except ValueError:

print("Invalid number entered.")

Результат:  
Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис

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.

**Код:**

main.py – залишився без змін

calculator.py:  
import math

from calculator.calculator\_memory import CalculationMemory

class Calculator:

def \_\_init\_\_(self, valid\_operators: list[str]):

self.\_\_memory = CalculationMemory("data.json")

self.\_\_pick\_init\_type(valid\_operators)

def do\_operation(self) -> float:

result = None

if self.\_\_operator == '+':

result = self.\_\_first\_number + self.\_\_second\_number

elif self.\_\_operator == '-':

result = self.\_\_first\_number - self.\_\_second\_number

elif self.\_\_operator == '\*':

result = self.\_\_first\_number \* self.\_\_second\_number

elif self.\_\_operator == '/':

if self.\_\_second\_number == 0:

print("Cannot divide by zero")

else:

result = self.\_\_first\_number / self.\_\_second\_number

elif self.\_\_operator == '^':

result = pow(self.\_\_first\_number, self.\_\_second\_number)

elif self.\_\_operator == '%':

result = self.\_\_first\_number % self.\_\_second\_number

elif self.\_\_operator == 'sqrt':

result = self.\_\_get\_sqrt()

else:

print("Invalid operator passed")

self.\_\_save\_option(result)

return result

def \_\_pick\_init\_type(self, valid\_operators: list[str]):

flag = True

option = int(input("Choose option:\n1 - Create new operation\n2 - use numbers from memory\n"))

while flag:

if option == 1:

self.\_\_new\_init(valid\_operators)

flag = False

elif option == 2:

self.\_\_init\_from\_memory(valid\_operators)

flag = False

else:

print("Invalid option entered")

def \_\_save\_option(self, result):

option = input("Do you wanna save data into file?(y/n)")

if option == 'y':

self.\_\_memory.write\_record(self.\_\_first\_number, self.\_\_second\_number, self.\_\_operator, result)

else:

print("Ok, let's move on")

def \_\_new\_init(self, valid\_operators):

self.\_\_first\_number = Calculator.\_\_enter\_number("Enter first number\n")

self.\_\_second\_number = Calculator.\_\_enter\_number("Enter second number\n")

self.\_\_operator = Calculator.\_\_enter\_operator(valid\_operators,

f"Enter operator:\nValid operators:{valid\_operators}\n")

def \_\_init\_from\_memory(self, valid\_operators: list[str]):

memory\_message = ''

data = self.\_\_memory.get\_all\_records()

for record in data:

memory\_message += ("{0} {1} {2} {3} = {4}"

.format(record['index'], record['firstNumber'], record['operator'],

record['secondNumber'], record['result']))

print("Choose memory data from list by its index:")

print(memory\_message)

index = int(input('Enter index:\n'))

record = self.\_\_memory.get\_record\_by\_index(index)

if record is None:

print('Invalid index')

return

self.\_\_first\_number = record["firstNumber"]

self.\_\_second\_number = record["secondNumber"]

self.\_\_operator = self.\_\_enter\_operator(valid\_operators, "Enter new operator:\n")

def \_\_get\_sqrt(self) -> float:

print(f"Your numbers:\n1 - {self.\_\_first\_number}\n2 - {self.\_\_second\_number}")

option = int(input("Choose the number that should be converted?\n"))

if option == 1:

if self.\_\_first\_number < 0:

print(f"Cannot get square of {self.\_\_first\_number} cause it is < 0")

else:

return math.sqrt(self.\_\_first\_number)

if option == 2:

if self.\_\_second\_number < 0:

print(f"Cannot get square of {self.\_\_second\_number} cause it is < 0")

else:

return math.sqrt(self.\_\_second\_number)

@staticmethod

def \_\_enter\_operator(valid\_operators: list[str], input\_message: str = "Enter operator\n") -> str:

while True:

operator = input(input\_message)

if operator in valid\_operators:

return operator

else:

print("Invalid operator entered.")

print(f"Valid operators: {valid\_operators}")

@staticmethod

def \_\_enter\_number(input\_msg: str = "Enter number:\n") -> float:

while True:

user\_input = input(input\_msg)

try:

number = float(user\_input)

return number

except ValueError:

print("Invalid number entered.")

calculator\_memory.py  
import json

class CalculationMemory:

def \_\_init\_\_(self, file\_name):

self.\_\_file\_name = file\_name

self.\_\_data = self.\_\_load\_data()

def write\_record(self, first\_number, second\_number, operator, result):

index = len(self.\_\_data)

record = {

"index": index,

"firstNumber": first\_number,

"secondNumber": second\_number,

"operator": operator,

"result": result

}

self.\_\_data.append(record)

self.\_\_save\_data()

def get\_all\_records(self):

return self.\_\_data

def clear\_memory(self):

self.\_\_data = []

self.\_\_save\_data()

def get\_record\_by\_index(self, index):

for record in self.\_\_data:

if record["index"] == index:

return record

return None

def \_\_load\_data(self):

try:

with open(self.\_\_file\_name, 'r') as file:

return json.load(file)

except FileNotFoundError:

return []

def \_\_save\_data(self):

with open(self.\_\_file\_name, 'w') as file:

json.dump(self.\_\_data, file)

**Результат:**

Структура:  
Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис  
Виконання:  
Зображення, що містить текст, знімок екрана

Автоматично згенерований опис

Файл збереження:  
Зображення, що містить текст, знімок екрана, програмне забезпечення, Мультимедійне програмне забезпечення

Автоматично згенерований опис

Завдання 9: Історія обчислень

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

**Код:**

main.py  
from calculator import Calculator

def main():

valid\_operators = ['+', '-', '\*', '/', '^', '%', 'sqrt']

flag = True

calculator = Calculator("data.json")

while flag:

print("Program options:\n1 - make calculation\n2 - see calculation history\n0 - exit")

option = int(input("Choose option:\n"))

if option == 1:

calculator.setup\_calculator(valid\_operators)

print(f"Result: {calculator.do\_operation()}")

elif option == 2:

calculator.print\_history()

elif option == 0:

flag = False

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

main()

calculator.py  
import math

from calculator.calculator\_memory import CalculationMemory

class Calculator:

def \_\_init\_\_(self, history\_file\_name):

self.\_\_memory = CalculationMemory(history\_file\_name)

def setup\_calculator(self, valid\_operators: list[str]):

self.\_\_pick\_init\_type(valid\_operators)

def do\_operation(self) -> float:

result = None

if self.\_\_operator == '+':

result = self.\_\_first\_number + self.\_\_second\_number

elif self.\_\_operator == '-':

result = self.\_\_first\_number - self.\_\_second\_number

elif self.\_\_operator == '\*':

result = self.\_\_first\_number \* self.\_\_second\_number

elif self.\_\_operator == '/':

if self.\_\_second\_number == 0:

print("Cannot divide by zero")

else:

result = self.\_\_first\_number / self.\_\_second\_number

elif self.\_\_operator == '^':

result = pow(self.\_\_first\_number, self.\_\_second\_number)

elif self.\_\_operator == '%':

result = self.\_\_first\_number % self.\_\_second\_number

elif self.\_\_operator == 'sqrt':

result = self.\_\_get\_sqrt()

else:

print("Invalid operator passed")

self.\_\_save\_option(result)

return result

def print\_history(self):

print(self.\_\_memory.get\_formatted\_string\_data())

def \_\_pick\_init\_type(self, valid\_operators: list[str]):

flag = True

option = int(input("Choose option:\n1 - Create new operation\n2 - use numbers from memory\n"))

while flag:

if option == 1:

self.\_\_new\_init(valid\_operators)

flag = False

elif option == 2:

self.\_\_init\_from\_memory(valid\_operators)

flag = False

else:

print("Invalid option entered")

def \_\_save\_option(self, result):

option = input("Do you wanna save data into file?(y/n)")

if option == 'y':

self.\_\_memory.write\_record(self.\_\_first\_number, self.\_\_second\_number, self.\_\_operator, result)

else:

print("Ok, let's move on")

def \_\_new\_init(self, valid\_operators):

self.\_\_first\_number = Calculator.\_\_enter\_number("Enter first number\n")

self.\_\_second\_number = Calculator.\_\_enter\_number("Enter second number\n")

self.\_\_operator = Calculator.\_\_enter\_operator(valid\_operators,

f"Enter operator:\nValid operators:{valid\_operators}\n")

def \_\_init\_from\_memory(self, valid\_operators: list[str]):

print("Choose memory data from list by its index:")

self.print\_history()

index = int(input('Enter index:\n'))

record = self.\_\_memory.get\_record\_by\_index(index)

if record is None:

print('Invalid index')

return

self.\_\_first\_number = record["firstNumber"]

self.\_\_second\_number = record["secondNumber"]

self.\_\_operator = self.\_\_enter\_operator(valid\_operators, "Enter new operator:\n")

def \_\_get\_sqrt(self) -> float:

print(f"Your numbers:\n1 - {self.\_\_first\_number}\n2 - {self.\_\_second\_number}")

option = int(input("Choose the number that should be converted?\n"))

if option == 1:

if self.\_\_first\_number < 0:

print(f"Cannot get square of {self.\_\_first\_number} cause it is < 0")

else:

return math.sqrt(self.\_\_first\_number)

if option == 2:

if self.\_\_second\_number < 0:

print(f"Cannot get square of {self.\_\_second\_number} cause it is < 0")

else:

return math.sqrt(self.\_\_second\_number)

@staticmethod

def \_\_enter\_operator(valid\_operators: list[str], input\_message: str = "Enter operator\n") -> str:

while True:

operator = input(input\_message)

if operator in valid\_operators:

return operator

else:

print("Invalid operator entered.")

print(f"Valid operators: {valid\_operators}")

@staticmethod

def \_\_enter\_number(input\_msg: str = "Enter number:\n") -> float:

while True:

user\_input = input(input\_msg)

try:

number = float(user\_input)

return number

except ValueError:

print("Invalid number entered.")

calculator\_memory.py  
import json

class CalculationMemory:

def \_\_init\_\_(self, file\_name):

self.\_\_file\_name = file\_name

self.\_\_data = self.\_\_load\_data()

def write\_record(self, first\_number, second\_number, operator, result):

index = len(self.\_\_data)

record = {

"index": index,

"firstNumber": first\_number,

"secondNumber": second\_number,

"operator": operator,

"result": result

}

self.\_\_data.append(record)

self.\_\_save\_data()

def get\_all\_records(self):

return self.\_\_data

def clear\_memory(self):

self.\_\_data = []

self.\_\_save\_data()

def get\_record\_by\_index(self, index):

for record in self.\_\_data:

if record["index"] == index:

return record

return None

def get\_formatted\_string\_data(self) -> str:

memory\_message = ''

for record in self.\_\_data:

memory\_message += ("{0}) {1} {2} {3} = {4}\n"

.format(record['index'], record['firstNumber'], record['operator'],

record['secondNumber'], record['result']))

return memory\_message

def \_\_load\_data(self):

try:

with open(self.\_\_file\_name, 'r') as file:

return json.load(file)

except FileNotFoundError:

return []

def \_\_save\_data(self):

with open(self.\_\_file\_name, 'w') as file:

json.dump(self.\_\_data, file)

**Результат:**Зображення, що містить текст, знімок екрана, програмне забезпечення, Шрифт

Автоматично згенерований опис

Завдання 10: Налаштування користувача

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

**Код:**

main.py  
from calculator import Calculator, CalculationSettings

def main():

valid\_operators = ['+', '-', '\*', '/', '^', '%', 'sqrt']

flag = True

calculator = Calculator("data.json")

calc\_settings = CalculationSettings()

while flag:

print("Program options:\n1 - make calculation\n2 - see calculation history\n"

"3 - clear memory\n4 - change calculation settings\n0 - exit")

option = int(input("Choose option:\n"))

if option == 1:

calculator.setup\_calculator(valid\_operators)

print("Result:")

result = calculator.do\_operation()

print(f"{result:.{calc\_settings.digits\_after\_point}f}")

elif option == 2:

calculator.print\_history()

elif option == 3:

calculator.clear\_memory()

elif option == 4:

calc\_settings.settings\_interface()

elif option == 0:

flag = False

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

main()

calculator.py  
import math

from calculator.calculation\_settings import CalculationSettings

from calculator.calculator\_memory import CalculationMemory

class Calculator:

def \_\_init\_\_(self, file\_name: str):

self.\_\_memory = CalculationMemory(file\_name)

def setup\_calculator(self, valid\_operators: list[str]):

self.\_\_pick\_init\_type(valid\_operators)

def do\_operation(self) -> float:

result = None

if self.\_\_operator == '+':

result = self.\_\_first\_number + self.\_\_second\_number

elif self.\_\_operator == '-':

result = self.\_\_first\_number - self.\_\_second\_number

elif self.\_\_operator == '\*':

result = self.\_\_first\_number \* self.\_\_second\_number

elif self.\_\_operator == '/':

if self.\_\_second\_number == 0:

print("Cannot divide by zero")

else:

result = self.\_\_first\_number / self.\_\_second\_number

elif self.\_\_operator == '^':

result = pow(self.\_\_first\_number, self.\_\_second\_number)

elif self.\_\_operator == '%':

result = self.\_\_first\_number % self.\_\_second\_number

elif self.\_\_operator == 'sqrt':

result = self.\_\_get\_sqrt()

else:

print("Invalid operator passed")

self.\_\_save\_option(result)

return result

def print\_history(self):

print(self.\_\_memory.get\_formatted\_string\_data())

def clear\_memory(self):

print(self.\_\_memory.clear\_memory())

def \_\_pick\_init\_type(self, valid\_operators: list[str]):

flag = True

option = int(input("Choose option:\n1 - Create new operation\n2 - use numbers from memory\n"))

while flag:

if option == 1:

self.\_\_new\_init(valid\_operators)

flag = False

elif option == 2:

self.\_\_init\_from\_memory(valid\_operators)

flag = False

else:

print("Invalid option entered")

def \_\_save\_option(self, result):

option = input("Do you wanna save data into file?(y/n)")

if option == 'y':

self.\_\_memory.add\_record(self.\_\_first\_number, self.\_\_second\_number, self.\_\_operator, result)

else:

print("Ok, let's move on")

def \_\_new\_init(self, valid\_operators):

self.\_\_first\_number = Calculator.\_\_enter\_number("Enter first number\n")

self.\_\_second\_number = Calculator.\_\_enter\_number("Enter second number\n")

self.\_\_operator = Calculator.\_\_enter\_operator(valid\_operators,

f"Enter operator:\nValid operators:{valid\_operators}\n")

def \_\_init\_from\_memory(self, valid\_operators: list[str]):

print("Choose memory data from list by its index:")

self.print\_history()

index = int(input('Enter index:\n'))

record = self.\_\_memory.get\_record\_by\_index(index)

if record is None:

print('Invalid index')

return

self.\_\_first\_number = record["firstNumber"]

self.\_\_second\_number = record["secondNumber"]

self.\_\_operator = self.\_\_enter\_operator(valid\_operators, "Enter new operator:\n")

def \_\_get\_sqrt(self) -> float:

print(f"Your numbers:\n1 - {self.\_\_first\_number}\n2 - {self.\_\_second\_number}")

option = int(input("Choose the number that should be converted?\n"))

if option == 1:

if self.\_\_first\_number < 0:

print(f"Cannot get square of {self.\_\_first\_number} cause it is < 0")

else:

return math.sqrt(self.\_\_first\_number)

if option == 2:

if self.\_\_second\_number < 0:

print(f"Cannot get square of {self.\_\_second\_number} cause it is < 0")

else:

return math.sqrt(self.\_\_second\_number)

@staticmethod

def \_\_enter\_operator(valid\_operators: list[str], input\_message: str = "Enter operator\n") -> str:

while True:

operator = input(input\_message)

if operator in valid\_operators:

return operator

else:

print("Invalid operator entered.")

print(f"Valid operators: {valid\_operators}")

@staticmethod

def \_\_enter\_number(input\_msg: str = "Enter number:\n") -> float:

while True:

user\_input = input(input\_msg)

try:

number = float(user\_input)

return number

except ValueError:

print("Invalid number entered.")

calculator\_memory.py  
import json

class CalculationMemory:

def \_\_init\_\_(self, file\_name):

self.\_\_file\_name = file\_name

self.\_\_data = self.\_\_load\_data()

def add\_record(self, first\_number, second\_number, operator, result):

index = len(self.\_\_data)

record = {

"index": index,

"firstNumber": first\_number,

"secondNumber": second\_number,

"operator": operator,

"result": result

}

self.\_\_data.append(record)

self.\_\_save\_data()

def get\_all\_records(self):

return self.\_\_data

def clear\_memory(self):

self.\_\_data = []

self.\_\_save\_data()

def get\_record\_by\_index(self, index):

for record in self.\_\_data:

if record["index"] == index:

return record

return None

def get\_formatted\_string\_data(self) -> str:

memory\_message = ''

for record in self.\_\_data:

memory\_message += ("{0}) {1} {2} {3} = {4}\n"

.format(record['index'], record['firstNumber'], record['operator'],

record['secondNumber'], record['result']))

return memory\_message

def \_\_load\_data(self):

try:

with open(self.\_\_file\_name, 'r') as file:

return json.load(file)

except FileNotFoundError:

return []

def \_\_save\_data(self):

with open(self.\_\_file\_name, 'w') as file:

json.dump(self.\_\_data, file)

calculator\_settings.py

class CalculationSettings:

def \_\_init\_\_(self, digits\_after\_point=2):

self.\_\_digits\_after\_point = digits\_after\_point

def settings\_interface(self):

flag = True

while flag:

option = int(input("Enter setting option:\n1 - change number of digits after point\n"

"0 - exit\n"))

if option == 1:

digits = int(input("Enter number of digits\n"))

self.digits\_after\_point = digits

elif option == 0:

flag = False

else:

print("Invalid option entered")

flag = False

@property

def digits\_after\_point(self):

return self.\_\_digits\_after\_point

@digits\_after\_point.setter

def digits\_after\_point(self, value):

if type(value) != int:

print("Value should be integer")

elif value < 0:

print("You cannot set value smaller then 0")

else:

self.\_\_digits\_after\_point = value

**Результат:**

Структура:  
Зображення, що містить текст, знімок екрана, програмне забезпечення, Шрифт

Автоматично згенерований опис

Виконання  
Зображення, що містить текст, знімок екрана

Автоматично згенерований опис  
Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис

Висновки. Виконавши ці завдання, я створив простий консольний калькулятор на Python, який може виконувати арифметичні операції, обробляти помилки та надавати користувачу зручний інтерфейс. Цей проект допоміг мені вивчити основний синтаксис Python і концепції, такі як введення користувача, умовні оператори, цикли та обробка помилок.