**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**

Кафедра інформаційних систем та мереж

**Звіт**

**з дисципліни «Спеціалізовані мови програмування»**

**Лабораторна робота № 1**

**Виконав:**

студент групи ІТ-21сп

Масняк Б. В.

**Прийняв**:

Щербак С.С.

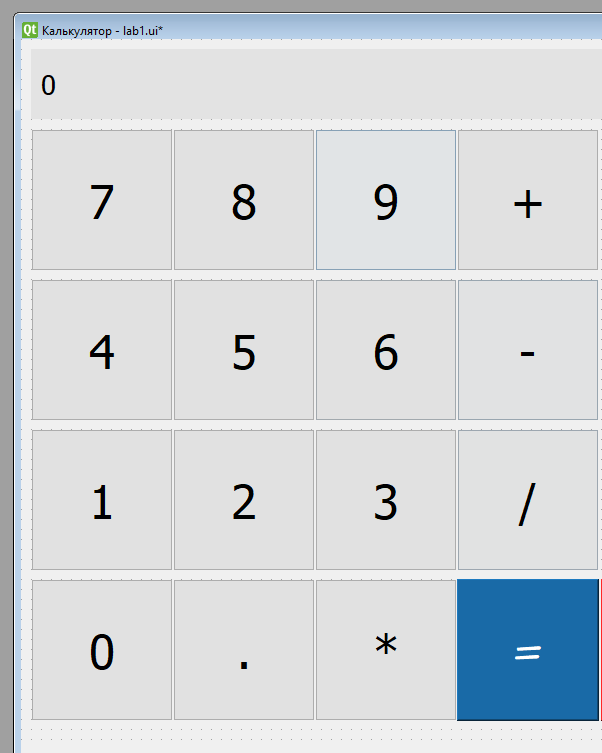
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Львів – 2023

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

**Хід роботи**

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



Код:

|  |
| --- |
| def retranslateUi(self, MainWindow):  \_translate = QtCore.QCoreApplication.translate  MainWindow.setWindowTitle(\_translate("MainWindow", "Калькулятор"))  self.label\_result.setText(\_translate("MainWindow", "0"))  self.btn\_7.setText(\_translate("MainWindow", "7"))  self.btn\_8.setText(\_translate("MainWindow", "8"))  self.btn\_9.setText(\_translate("MainWindow", "9"))  self.btn\_add.setText(\_translate("MainWindow", "+"))  self.btn\_6.setText(\_translate("MainWindow", "6"))  self.btn\_subtract.setText(\_translate("MainWindow", "-"))  self.btn\_5.setText(\_translate("MainWindow", "5"))  self.btn\_4.setText(\_translate("MainWindow", "4"))  self.btn\_3.setText(\_translate("MainWindow", "3"))  self.btn\_divide.setText(\_translate("MainWindow", "/"))  self.btn\_2.setText(\_translate("MainWindow", "2"))  self.btn\_1.setText(\_translate("MainWindow", "1"))  self.btn\_0.setText(\_translate("MainWindow", "0"))  self.btn\_equal.setText(\_translate("MainWindow", "="))  self.btn\_dot.setText(\_translate("MainWindow", "."))  self.btn\_multiply.setText(\_translate("MainWindow", "\*"))  def add\_functions(self):  self.btn\_0.clicked.connect(lambda: self.write\_number(self.btn\_0.text()))  self.btn\_1.clicked.connect(lambda: self.write\_number(self.btn\_1.text()))  self.btn\_2.clicked.connect(lambda: self.write\_number(self.btn\_2.text()))  self.btn\_3.clicked.connect(lambda: self.write\_number(self.btn\_3.text()))  self.btn\_4.clicked.connect(lambda: self.write\_number(self.btn\_4.text()))  self.btn\_5.clicked.connect(lambda: self.write\_number(self.btn\_5.text()))  self.btn\_6.clicked.connect(lambda: self.write\_number(self.btn\_6.text()))  self.btn\_7.clicked.connect(lambda: self.write\_number(self.btn\_7.text()))  self.btn\_8.clicked.connect(lambda: self.write\_number(self.btn\_8.text()))  self.btn\_9.clicked.connect(lambda: self.write\_number(self.btn\_9.text()))  self.btn\_multiply.clicked.connect(lambda: self.write\_number(self.btn\_multiply.text()))  self.btn\_add.clicked.connect(lambda: self.write\_number(self.btn\_add.text()))  self.btn\_interest.clicked.connect(lambda: self.write\_number(self.btn\_interest.text()))  self.btn\_divide.clicked.connect(lambda: self.write\_number(self.btn\_divide.text()))  self.btn\_subtract.clicked.connect(lambda: self.write\_number(self.btn\_subtract.text()))  self.btn\_equal.clicked.connect(self.results) |

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

*Користувач може ввести, лише, дійсні оператори, тому перевірка не потрібна*

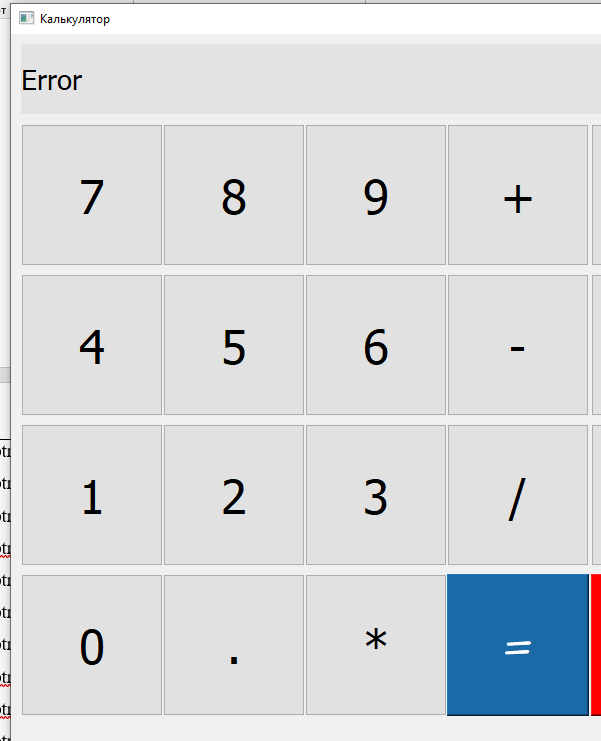
1. Обчислення. Виконайте обчислення на основі введення користувача (наприклад, додавання, віднімання, множення, ділення) і відобразіть результат.

|  |
| --- |
| def add\_functions(self):  self.btn\_equal.clicked.connect(self.results)  def results(self):  res = eval(self.label\_result.text())  self.label\_result.setText(str(res)) |

1. 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.

*Програма написана з використанням графічного інтерфейсу. Користувач може вийти з програми в будь-який час, а функція обчислень дозоляє повторне виконання.*

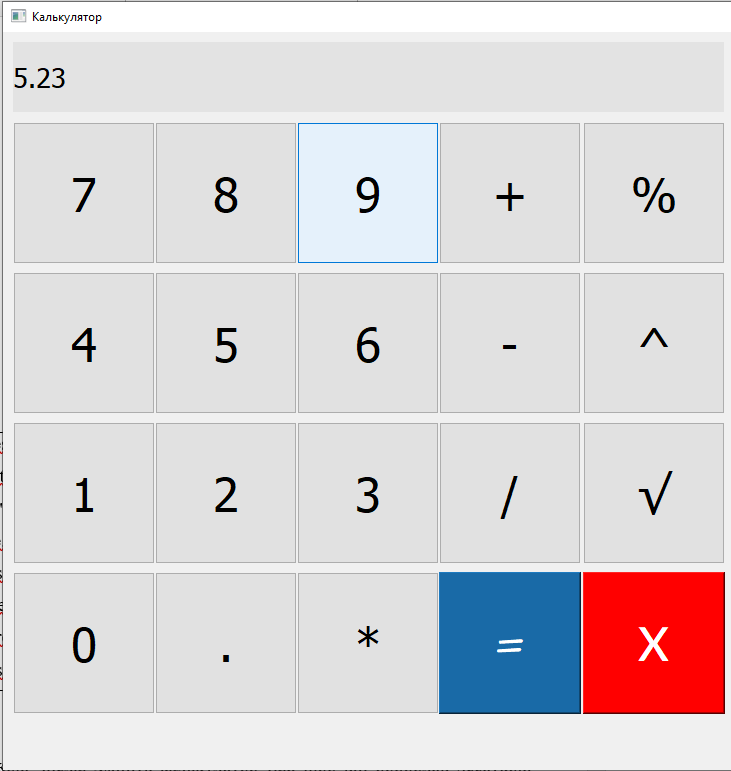
1. Обробка помилок. Реалізуйте обробку помилок для обробки ділення на нуль або інших потенційних помилок. Відобразіть відповідне повідомлення про помилку, якщо виникає помилка.



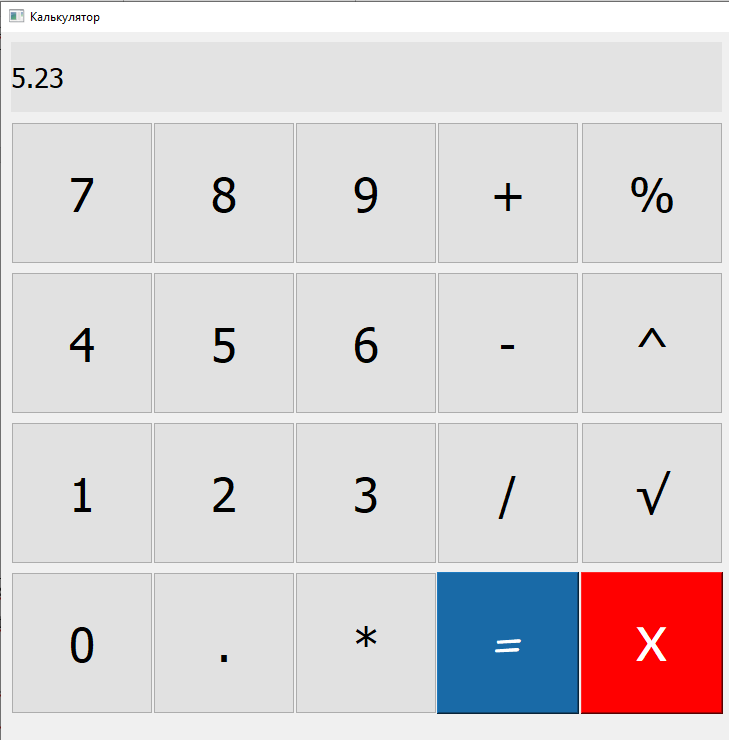
Код:

|  |
| --- |
| def results(self):  pattern = r"/0\.\d+"  if ("/0" in self.label\_result.text() or "Error" in self.label\_result.text()) and not re.search(pattern, self.label\_result.text()):  self.label\_result.setText("Error")  else:  res = eval(self.label\_result.text())  self.label\_result.setText(str(res)) |

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



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



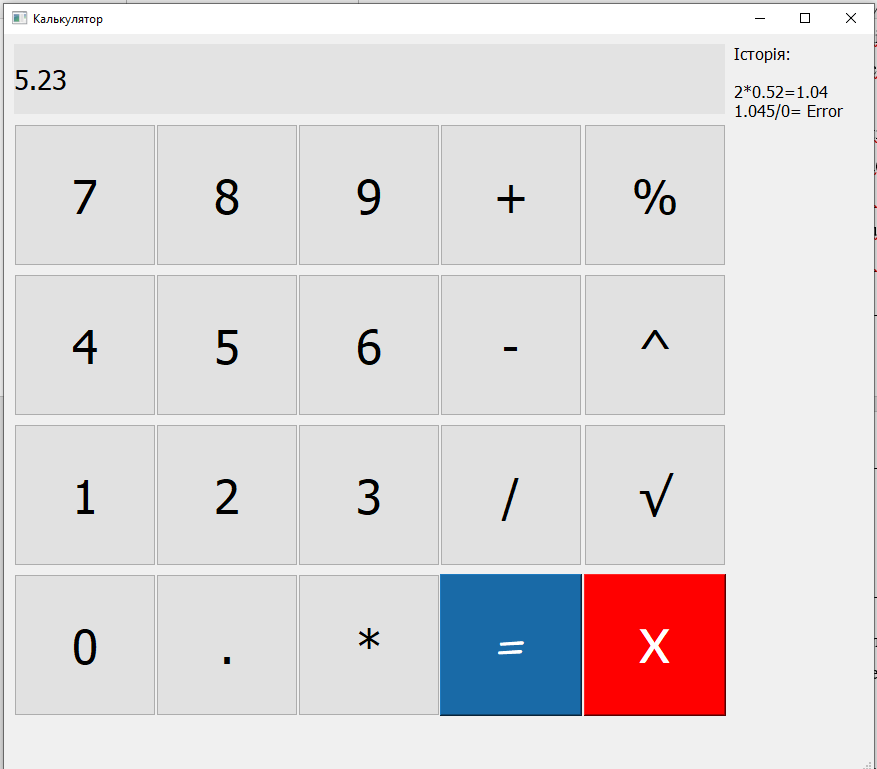
Код:

|  |
| --- |
| self.btn\_interest.setText(\_translate("MainWindow", "%")) self.btn\_pow.setText(\_translate("MainWindow", "^")) self.btn\_root.setText(\_translate("MainWindow", "√")) self.btn\_del.setText(\_translate("MainWindow", "Х"))  self.btn\_pow.clicked.connect(lambda: self.write\_number("\*\*")) self.btn\_interest.clicked.connect(lambda: self.write\_number(self.btn\_interest.text())) self.btn\_del.clicked.connect(self.del\_function) self.btn\_root.clicked.connect(self.root\_function)  def root\_function(self):  if float(eval(self.label\_result.text())) < 0 :  self.label\_history.setText(self.label\_history.text() + "\n√(" + self.label\_result.text() + ")= Error")  self.label\_result.setText("Error")  else:  res = math.sqrt(eval(self.label\_result.text()))  self.label\_history.setText(self.label\_history.text() + "\n√(" + self.label\_result.text() + ")=" + str(res))  self.label\_result.setText(str(res)) |

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

|  |
| --- |
| self.label\_history.setText(\_translate("MainWindow", "Історія:\n")) |

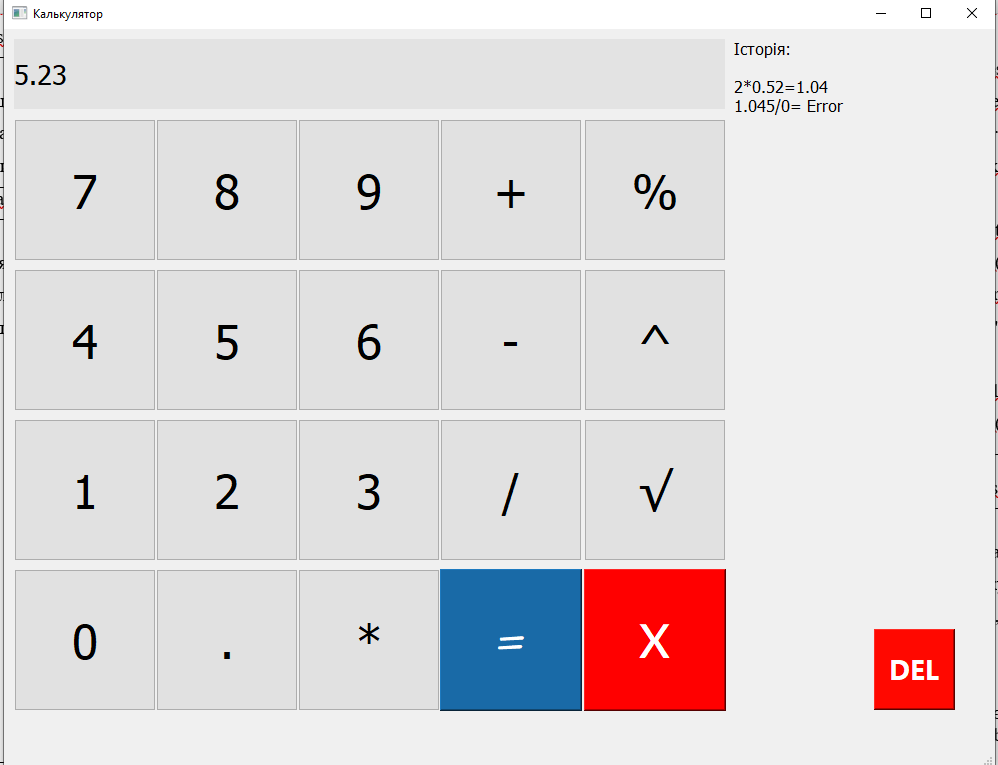
1. Історія обчислень. Створіть журнал, який зберігає історію попередніх обчислень, включаючи вираз і результат. Дозвольте користувачам переглядати історію своїх обчислень.



Код:

|  |
| --- |
| def results(self):  pattern = r"/0\.\d+"  if ("/0" in self.label\_result.text() or "Error" in self.label\_result.text()) and not re.search(pattern, self.label\_result.text()):  self.label\_history.setText(self.label\_history.text() + "\n" + self.label\_result.text() + "= Error")  self.label\_result.setText("Error")  else:  res = eval(self.label\_result.text())  self.label\_history.setText(self.label\_history.text() + "\n" + self.label\_result.text() + "=" + str(res))  self.label\_result.setText(str(res))  def root\_function(self):  if float(eval(self.label\_result.text())) < 0 :  self.label\_history.setText(self.label\_history.text() + "\n√(" + self.label\_result.text() + ")= Error")  self.label\_result.setText("Error")  else:  res = math.sqrt(eval(self.label\_result.text()))  self.label\_history.setText(self.label\_history.text() + "\n√(" + self.label\_result.text() + ")=" + str(res))  self.label\_result.setText(str(res)) |

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



Код:

|  |
| --- |
| self.btn\_clear.setText(\_translate("MainWindow", "DEL"))  def add\_functions(self):  self.btn\_clear.clicked.connect(self.clear\_function)  def clear\_function(self):  self.label\_history.setText("Історія:\n") |

**Висновок:** під час виконання цієї лабораторної роботи я опанував основи роботи в мові програмування Python, ознайомився з бібліотекою PyQt5 і навчився створювати прості застосунки з користувацьким інтерфейсом.