ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ РЕСПУБЛИКИ ХАКАСИЯ  
«ХАКАССКИЙ ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ»

Специальность 09.02.07

«Информационные системы и программирование»

ТЕМА: Введение в PyQt

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Студент гр. | ИС(ТП)-31 |  |  | | | |  |  | Соломатин А.Д. | | |
|  |  |  | *подпись* | | | |  |  | *Фамилия, И.О.* | | |
| Руководитель |  |  |  |  |  |  | | | |  | Брюханова И.Н. |
|  | *оценка* |  | *дата* |  |  | *подпись* | | | |  | *Фамилия, И.О.* |

Абакан, 2024

**Цель работы: https://github.com/Aleksey4ik6/Alekseyka**

**Задание №1**

Для дальнейшего изучения создания десктопных приложений

на PyQt вам понадобится знание системы сигналов и слотов. Это

задание поможет вам понять эту систему.

Создайте небольшое приложение с одной кнопкой и текстовой

меткой. При каждом нажатии на кнопку, число в метке должно

увеличиваться на 1. Используйте сигналы и слоты для связывания

кнопки с изменением текста в метке.

Ход выполнения:

− Создайте главное окно приложения.

o Используйте QMainWindow или простой QWidget.

− Добавьте элементы интерфейса:

o Кнопку (QPushButton) с текстом (например,

«Нажми меня»).

o Метку (QLabel), которая будет показывать число

нажатий.

− Реализуйте логику через сигналы и слоты (используйте

вспомогательный материал):

o Подключите сигнал clicked от кнопки к

пользовательскому слоту, который увеличивает

счётчик и обновляет текст в метке.

Ожидаемый результат:

При запуске приложения метка показывает «0». При каждом

нажатии кнопки метка увеличивает своё значение на 1.

2.Листинг программы:

import sys

from PyQt6.QtWidgets import QApplication, QWidget, QPushButton, QLabel, QVBoxLayout

from PyQt6.QtCore import Qt

class CounterApp(QWidget):

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

super().\_\_init\_\_()

self.counter = 0

# Элементы интерфейса

self.button = QPushButton("Нажми меня")

self.label = QLabel("0", alignment=Qt.AlignmentFlag.AlignCenter)

# Связываем сигнал и слот

self.button.clicked.connect(self.increment\_counter)

# Компоновка

layout = QVBoxLayout()

layout.addWidget(self.label)

layout.addWidget(self.button)

self.setLayout(layout)

# Настройки окна

self.setWindowTitle("Счётчик")

self.resize(200, 150)

def increment\_counter(self):

self.counter += 1

self.label.setText(str(self.counter))

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

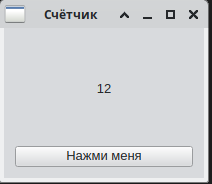
app = QApplication(sys.argv)

window = CounterApp()

window.show()

sys.exit(app.exec())

3.Результаты

****

**Задание №2**

При запуске приложения метка показывает «0». При каждом

нажатии кнопки метка увеличивает своё значение на 1.

Описание задачи 2 «Настройка отображения текста»:

Создайте приложение, в котором:

− выбирается цвет текста из выпадающего списка

(QComboBox);

− устанавливается стиль текста (жирный или курсив) с

помощью чекбоксов (QCheckBox);

− выбирается размер текста с помощью радиокнопок

(QRadioButton).

Изменения применяются к тексту в метке (QLabel) в режиме

реального времени через сигналы и слоты.

2.Листинг программы:

import sys

from PyQt6.QtWidgets import (

QApplication, QWidget, QLabel, QComboBox, QCheckBox, QRadioButton, QVBoxLayout, QHBoxLayout

)

from PyQt6.QtCore import Qt

from PyQt6.QtGui import QFont

class TextStylerApp(QWidget):

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

super().\_\_init\_\_()

self.init\_ui()

def init\_ui(self):

# Метка с текстом

self.label = QLabel("Пример текста")

self.label.setAlignment(Qt.AlignmentFlag.AlignCenter)

# Выпадающий список для выбора цвета текста

self.color\_combo = QComboBox()

self.color\_combo.addItems(["Чёрный", "Красный", "Зелёный", "Синий"])

self.color\_combo.currentTextChanged.connect(self.update\_style)

# Чекбоксы для стилей текста

self.bold\_check = QCheckBox("Жирный")

self.italic\_check = QCheckBox("Курсив")

self.bold\_check.stateChanged.connect(self.update\_style)

self.italic\_check.stateChanged.connect(self.update\_style)

# Радиокнопки для размера текста

self.small\_radio = QRadioButton("Маленький")

self.small\_radio.setChecked(True)

self.medium\_radio = QRadioButton("Средний")

self.large\_radio = QRadioButton("Большой")

self.small\_radio.toggled.connect(self.update\_style)

self.medium\_radio.toggled.connect(self.update\_style)

self.large\_radio.toggled.connect(self.update\_style)

# Компоновка

layout = QVBoxLayout()

layout.addWidget(self.label)

layout.addWidget(self.color\_combo)

layout.addWidget(self.bold\_check)

layout.addWidget(self.italic\_check)

layout.addWidget(self.small\_radio)

layout.addWidget(self.medium\_radio)

layout.addWidget(self.large\_radio)

self.setLayout(layout)

# Настройки окна

self.setWindowTitle("Стилизация текста")

self.resize(300, 300)

def update\_style(self):

color\_map = {

"Чёрный": "black",

"Красный": "red",

"Зелёный": "green",

"Синий": "blue"

}

color = color\_map[self.color\_combo.currentText()]

bold = QFont.Weight.Bold if self.bold\_check.isChecked() else QFont.Weight.Normal

italic = self.italic\_check.isChecked()

size = 12 if self.small\_radio.isChecked() else 16 if self.medium\_radio.isChecked() else 20

font = QFont()

font.setPointSize(size)

font.setWeight(bold)

font.setItalic(italic)

self.label.setFont(font)

self.label.setStyleSheet(f"color: {color};")

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

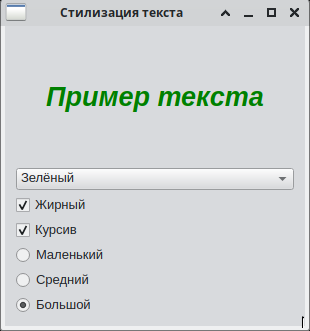
app = QApplication(sys.argv)

window = TextStylerApp()

window.show()

sys.exit(app.exec())

3.Результаты



**Задание №3**

**В предыдущем практическом занятии вы реализовывали**

**калькулятор по предложенному макету. Теперь заказчик просит вас**

**реализовать функционал калькулятора, а именно выполнение**

**базовых арифметических операций (сложение, вычитание,**

**умножение, деление) между двумя числами. Используйте сигналы и**

**слоты для связи виджетов и реализации функционала.**

**Ожидаемый результат:**

**При корректном вводе чисел и нажатии кнопки результат**

**отображается. При ошибке отображается соответствующее**

**сообщение (например, «Некорреткный ввод» или «Деление на ноль»).**

**2.Листинг программы:**

**import sys**

from PyQt6.QtWidgets import (

QApplication, QMainWindow, QVBoxLayout, QFrame, QGridLayout, QWidget,

QPushButton, QLineEdit

)

from PyQt6.QtCore import QSize, Qt

from PyQt6.QtGui import QFont

class Calculator(QMainWindow):

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

super().\_\_init\_\_()

self.setWindowTitle("Калькулятор")

self.setFixedSize(QSize(300, 450))

# Центральный виджет и основная компоновка

central\_widget = QWidget()

self.setCentralWidget(central\_widget)

main\_layout = QVBoxLayout(central\_widget)

# Экран калькулятора

self.display = QLineEdit()

self.display.setFixedHeight(50)

self.display.setReadOnly(True)

self.display.setFont(QFont('Arial', 20))

self.display.setAlignment(Qt.AlignmentFlag.AlignRight)

main\_layout.addWidget(self.display)

# Кнопки калькулятора

self.buttons = [

'7', '8', '9', '/',

'4', '5', '6', '\*',

'1', '2', '3', '-',

'0', '.', 'C', '+'

]

# Сетка для кнопок

grid\_layout = QGridLayout()

main\_layout.addLayout(grid\_layout)

for i, text in enumerate(self.buttons):

button = QPushButton(text)

button.setFixedSize(QSize(50, 50))

grid\_layout.addWidget(button, i // 4, i % 4)

button.clicked.connect(self.on\_button\_click)

# Кнопка "равно"

self.equal\_button = QPushButton('=')

self.equal\_button.setFixedSize(QSize(110, 50))

grid\_layout.addWidget(self.equal\_button, 4, 2, 1, 2)

self.equal\_button.clicked.connect(self.on\_equal)

# Переменные для операций

self.current\_input = ""

self.first\_operand = None

self.operator = None

def on\_button\_click(self):

button = self.sender()

text = button.text()

if text == 'C':

self.current\_input = ""

self.first\_operand = None

self.operator = None

self.display.clear()

elif text in '0123456789.':

self.current\_input += text

self.display.setText(self.current\_input)

elif text in '+-\*/':

if self.current\_input:

try:

self.first\_operand = float(self.current\_input)

self.operator = text

self.current\_input = ""

except ValueError:

self.display.setText("Ошибка")

def on\_equal(self):

if self.operator and self.current\_input:

try:

second\_operand = float(self.current\_input)

if self.operator == '+':

result = self.first\_operand + second\_operand

elif self.operator == '-':

result = self.first\_operand - second\_operand

elif self.operator == '\*':

result = self.first\_operand \* second\_operand

elif self.operator == '/':

if second\_operand == 0:

self.display.setText("Деление на ноль!")

return

result = self.first\_operand / second\_operand

self.display.setText(str(result))

self.current\_input = str(result)

self.first\_operand = result

self.operator = None

except ValueError:

self.display.setText("Ошибка")

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

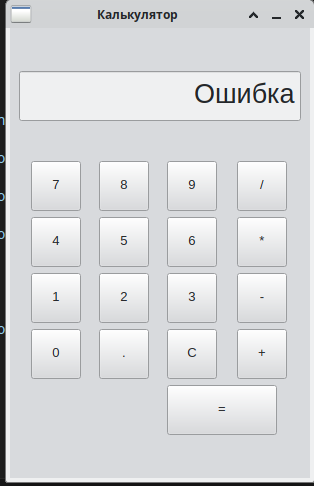
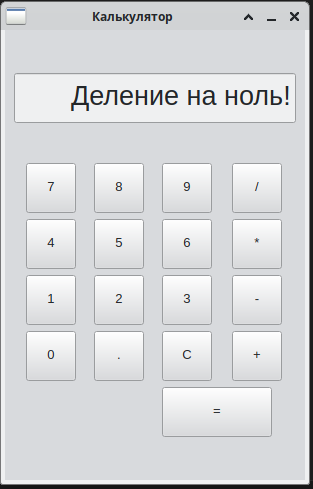
app = QApplication(sys.argv)

calculator = Calculator()

calculator.show()

sys.exit(app.exec())

**3.Результаты**

****