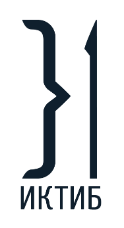
МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное учреждения высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт компьютерных технологий и информационной безопасности

Кафедра математического обеспечения и применения ЭВМ



**ЛАБОРАТОРНАЯ РАБОТА № 5**

**ОТЧЁТ**

по дисциплине

**«Разработка прикладных решений на языке Python»**

Выполнил:

Студент группы

КТмо2-16

|  |  |  |
| --- | --- | --- |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Руденко К. Д. |
|  | *подпись* |  |

Проверил:

Доцент кафедры САиТ, к.т.н.

|  |  |  |
| --- | --- | --- |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Лапшин В. С. |
|  | *подпись* |  |

Оценка

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

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2025 г.

**5 лабораторная работа.**

**Руденко Константин Дмитриевич – 3 вариант**

**Теоретические вопросы :**

1.Какие основные модули PyQt нужно импортировать для создания простого окна (QApplication и QWidget)

**Ответ:**

**QApplication - управляет основным циклом приложения и настройками**

**QWidget - базовый класс для всех объектов интерфейса, используется как основное окно**

2. Что такое цикл событий (event loop) в PyQt и как он запускается с помощью app.exec\_()? Почему это важн о?

**Ответ: Цикл событий - это бесконечный цикл, который постоянно ожидает и обрабатывает пользовательские действия (клики мыши, нажатия клавиш, системные события). Запускается методом app.exec\_(). Обеспевает отклик интерфейса на действия пользователяБез него приложение сразу завершит работу после вызова window.show()/ Координирует обработку всех событий в правильном порядке**

3. Как использовать метод move() для позиционирования кнопки относительно родительского виджета?

**Ответ: Метод move(x, y) позволяет точно позиционировать виджеты относительно родительского контейнера:**

**x - координата по горизонтали (в пикселях от левого края)**

**y - координата по вертикали (в пикселях от верхнего края)**

**Координатная система начинается в левом верхнем углу родительского виджета**

4.Как добавить поле ввода текста (QLineEdit) и метку (QLabel) в интерфейс? Как получить текст из поля?

**Ответ:Для работы с текстовыми элементами используются:**

**QLineEdit - однострочное поле ввода текста**

**QLabel - текстовая метка для отображения информации**

**Получение текста из поля ввода осуществляется методом text():**

Часть 1

**Изображение выглядит как текст, снимок экрана, диаграмма, программное обеспечение

Автоматически созданное описание**

Решение:

import pandas as pd  
import matplotlib.pyplot as plt  
from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg  
import tkinter as tk  
from tkinter import ttk  
  
df = pd.read\_csv("birth\_and\_death\_2017.csv")  
  
df["Родилось"] = pd.to\_numeric(df["Родилось"], errors="coerce")  
df["Умерло"] = pd.to\_numeric(df["Умерло"], errors="coerce")  
  
years = sorted(df["Год"].unique())  
  
root = tk.Tk()  
root.title("Рождение и смертность (1950–2016)")  
root.geometry("1100x700")  
  
top\_frame = tk.Frame(root)  
top\_frame.pack(pady=10)  
  
check\_frame = tk.LabelFrame(root, text="Выберите года", padx=10, pady=10)  
check\_frame.pack(side=tk.LEFT, fill=tk.Y, padx=10, pady=10)  
  
plot\_frame = tk.Frame(root)  
plot\_frame.pack(side=tk.RIGHT, fill=tk.BOTH, expand=True)  
  
tk.Label(top\_frame, text="Тип данных:", font=("Arial", 12)).pack(side=tk.LEFT, padx=5)  
type\_combo = ttk.Combobox(top\_frame, values=["Все", "Городское", "Сельское"], state="readonly", width=15)  
type\_combo.set("Все")  
type\_combo.pack(side=tk.LEFT, padx=5)  
  
selected\_years = {year: tk.BooleanVar(value=True) for year in years}  
canvas = None # для хранения текущего графика  
  
  
def plot\_graph():  
 global canvas  
 if canvas:  
 canvas.get\_tk\_widget().destroy()  
  
 fig, ax = plt.subplots(figsize=(8, 5))  
  
 selection = type\_combo.get()  
 active\_years = [year for year, var in selected\_years.items() if var.get()]  
  
 if not active\_years:  
 ax.text(0.5, 0.5, "Нет выбранных годов", ha="center", va="center", fontsize=14)  
 ax.axis("off")  
 else:  
 data = df[df["Год"].isin(active\_years)]  
  
 if selection == "Все":  
 city = data[data["Тип"] == "Городское"]  
 rural = data[data["Тип"] == "Сельское"]  
  
 ax.plot(city["Год"], city["Родилось"], color="blue", label="Родилось (Городское)")  
 ax.plot(city["Год"], city["Умерло"], color="orange", label="Умерло (Городское)")  
 ax.plot(rural["Год"], rural["Родилось"], color="blue", linestyle="--", label="Родилось (Сельское)")  
 ax.plot(rural["Год"], rural["Умерло"], color="orange", linestyle="--", label="Умерло (Сельское)")  
 else:  
 data = data[data["Тип"] == selection]  
 ax.plot(data["Год"], data["Родилось"], color="blue", label="Родилось")  
 ax.plot(data["Год"], data["Умерло"], color="orange", label="Умерло")  
  
 ax.set\_xlabel("Год")  
 ax.set\_ylabel("Человек")  
 ax.set\_title(f"Демографические данные: {selection.lower()}")  
 ax.legend()  
 ax.grid(True)  
  
 canvas = FigureCanvasTkAgg(fig, master=plot\_frame)  
 canvas.draw()  
 canvas.get\_tk\_widget().pack(fill=tk.BOTH, expand=True)  
  
  
def toggle\_year(\_=None):  
 plot\_graph()  
  
  
for i, year in enumerate(years):  
 cb = tk.Checkbutton(check\_frame, text=str(year), variable=selected\_years[year],  
 command=toggle\_year)  
 cb.grid(row=i // 3, column=i % 3, sticky="w")  
  
btn = tk.Button(top\_frame, text="Обновить график", command=plot\_graph)  
btn.pack(side=tk.LEFT, padx=10)  
  
plot\_graph()  
  
root.mainloop()

Пример работы программы:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Часть 2

Практическая работа № 5 часть 2 «Компоненты списков и выбора в PyQt5»

**Цель задания**: Создать приложение с графическим интерфейсом на PyQt5, которое позволяет пользователю:

1. Просматривать список задач в QListWidget.
2. Выбирать категорию задач с помощью QComboBox.
3. При выборе категории в выпадающем списке отображать только задачи, относящиеся к выбранной категории.
4. При двойном клике на задачу в списке показывать сообщение с описанием задачи.

Вариант 6.

import sys  
from PyQt5.QtWidgets import (QApplication, QMainWindow, QVBoxLayout, QHBoxLayout,  
 QWidget, QComboBox, QListWidget, QTextEdit,  
 QPushButton, QLabel, QMessageBox, QInputDialog,  
 QListWidgetItem)  
from PyQt5.QtCore import Qt  
  
  
class NoteApp(QMainWindow):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.notes = {  
 "Работа": [  
 {"title": "План на квартал",  
 "content": "1. Завершить проект А\n2. Провести встречи с командой\n3. Подготовить отчеты"},  
 {"title": "Идеи для проекта",  
 "content": "• Реализовать новый модуль\n• Оптимизировать производительность\n• Добавить аналитику"}  
 ],  
 "Идеи": [  
 {"title": "Придумать дизайн приложения",  
 "content": "Создать современный интерфейс с темной темой и анимациями"}  
 ],  
 "Личное": [  
 {"title": "Покупки", "content": "• Молоко\n• Хлеб\n• Фрукты"},  
 {"title": "Фильмы к просмотру", "content": "Интерстеллар\nНачало\nОстров проклятых"}  
 ]  
 }  
 self.initUI()  
  
 def initUI(self):  
 self.setWindowTitle('Приложение для заметок')  
 self.setGeometry(100, 100, 800, 600)  
  
 # Центральный виджет  
 central\_widget = QWidget()  
 self.setCentralWidget(central\_widget)  
  
 # Основной layout  
 main\_layout = QHBoxLayout()  
 central\_widget.setLayout(main\_layout)  
  
 # Левая панель (список заметок)  
 left\_panel = QVBoxLayout()  
  
 # Выбор категории  
 category\_label = QLabel('Категория:')  
 self.category\_combo = QComboBox()  
 self.category\_combo.addItems(["Работа", "Идеи", "Личное"])  
 self.category\_combo.currentTextChanged.connect(self.filter\_notes)  
  
 # Список заметок  
 self.notes\_list = QListWidget()  
 self.notes\_list.itemDoubleClicked.connect(self.show\_note\_content)  
  
 # Кнопка новой заметки  
 self.new\_note\_btn = QPushButton('Новая заметка')  
 self.new\_note\_btn.clicked.connect(self.create\_new\_note)  
  
 # Кнопка удаления заметки  
 self.delete\_note\_btn = QPushButton('Удалить заметку')  
 self.delete\_note\_btn.clicked.connect(self.delete\_note)  
  
 left\_panel.addWidget(category\_label)  
 left\_panel.addWidget(self.category\_combo)  
 left\_panel.addWidget(QLabel('Заметки:'))  
 left\_panel.addWidget(self.notes\_list)  
 left\_panel.addWidget(self.new\_note\_btn)  
 left\_panel.addWidget(self.delete\_note\_btn)  
  
 # Правая панель (редактор)  
 right\_panel = QVBoxLayout()  
  
 # Поля для редактирования  
 self.title\_edit = QTextEdit()  
 self.title\_edit.setMaximumHeight(40)  
 self.title\_edit.setPlaceholderText("Заголовок заметки...")  
  
 self.content\_edit = QTextEdit()  
 self.content\_edit.setPlaceholderText("Содержание заметки...")  
  
 # Кнопки редактора  
 self.save\_btn = QPushButton('Сохранить')  
 self.save\_btn.clicked.connect(self.save\_note)  
  
 self.cancel\_btn = QPushButton('Отмена')  
 self.cancel\_btn.clicked.connect(self.cancel\_edit)  
  
 right\_panel.addWidget(QLabel('Заголовок:'))  
 right\_panel.addWidget(self.title\_edit)  
 right\_panel.addWidget(QLabel('Содержание:'))  
 right\_panel.addWidget(self.content\_edit)  
 right\_panel.addWidget(self.save\_btn)  
 right\_panel.addWidget(self.cancel\_btn)  
  
 # Добавляем панели в основной layout  
 main\_layout.addLayout(left\_panel, 1)  
 main\_layout.addLayout(right\_panel, 2)  
  
 # Загружаем начальные данные  
 self.filter\_notes()  
 self.current\_note\_index = -1  
 self.current\_category = ""  
  
 def filter\_notes(self):  
 *"""Фильтрует заметки по выбранной категории"""* category = self.category\_combo.currentText()  
 self.notes\_list.clear()  
  
 if category in self.notes:  
 for note in self.notes[category]:  
 item = QListWidgetItem(note["title"])  
 self.notes\_list.addItem(item)  
  
 def show\_note\_content(self, item):  
 *"""Показывает содержание заметки при двойном клике"""* category = self.category\_combo.currentText()  
 note\_title = item.text()  
  
 # Находим заметку  
 for i, note in enumerate(self.notes[category]):  
 if note["title"] == note\_title:  
 # Показываем в редакторе  
 self.title\_edit.setPlainText(note["title"])  
 self.content\_edit.setPlainText(note["content"])  
 self.current\_note\_index = i  
 self.current\_category = category  
 break  
  
 def create\_new\_note(self):  
 *"""Создает новую заметку"""* category = self.category\_combo.currentText()  
  
 # Очищаем редактор  
 self.title\_edit.clear()  
 self.content\_edit.clear()  
 self.current\_note\_index = -1  
 self.current\_category = category  
  
 # Устанавливаем фокус на заголовок  
 self.title\_edit.setFocus()  
  
 def save\_note(self):  
 *"""Сохраняет заметку"""* title = self.title\_edit.toPlainText().strip()  
 content = self.content\_edit.toPlainText().strip()  
 category = self.current\_category or self.category\_combo.currentText()  
  
 if not title:  
 QMessageBox.warning(self, 'Ошибка', 'Введите заголовок заметки!')  
 return  
  
 if category not in self.notes:  
 self.notes[category] = []  
  
 if self.current\_note\_index == -1:  
 # Новая заметка  
 new\_note = {"title": title, "content": content}  
 self.notes[category].append(new\_note)  
 QMessageBox.information(self, 'Успех', 'Заметка создана!')  
 else:  
 # Редактирование существующей заметки  
 self.notes[category][self.current\_note\_index] = {"title": title, "content": content}  
 QMessageBox.information(self, 'Успех', 'Заметка обновлена!')  
  
 # Обновляем список  
 self.filter\_notes()  
 self.current\_note\_index = -1  
  
 def delete\_note(self):  
 *"""Удаляет выбранную заметку"""* current\_item = self.notes\_list.currentItem()  
 if not current\_item:  
 QMessageBox.warning(self, 'Ошибка', 'Выберите заметку для удаления!')  
 return  
  
 category = self.category\_combo.currentText()  
 note\_title = current\_item.text()  
  
 reply = QMessageBox.question(self, 'Подтверждение',  
 f'Удалить заметку "{note\_title}"?',  
 QMessageBox.Yes | QMessageBox.No)  
  
 if reply == QMessageBox.Yes:  
 # Удаляем заметку  
 self.notes[category] = [note for note in self.notes[category]  
 if note["title"] != note\_title]  
  
 # Очищаем редактор  
 self.title\_edit.clear()  
 self.content\_edit.clear()  
 self.current\_note\_index = -1  
  
 # Обновляем список  
 self.filter\_notes()  
 QMessageBox.information(self, 'Успех', 'Заметка удалена!')  
  
 def cancel\_edit(self):  
 *"""Отменяет редактирование"""* self.title\_edit.clear()  
 self.content\_edit.clear()  
 self.current\_note\_index = -1  
 self.notes\_list.clearSelection()  
  
  
def main():  
 app = QApplication(sys.argv)  
 window = NoteApp()  
 window.show()  
 sys.exit(app.exec\_())  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

Изображение выглядит как текст, снимок экрана, программное обеспечение, компьютер

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Содержимое, созданное искусственным интеллектом, может быть неверным.  
Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Ссылка на гитхаб: [all/Py3semestr2025RudenkoKTmo2-16/laba5/лаба 5 at master · Kosten-73/all · GitHub](https://github.com/Kosten-73/all/tree/master/Py3semestr2025RudenkoKTmo2-16/laba5/%D0%BB%D0%B0%D0%B1%D0%B0%205)