Т. 12. №2 2026
https://doi.org/10.33619/2414-2948/123

УДК 004.738 https://doi.org/10.33619/2414-2948/123/17

ОСОБЕННОСТИ ПРИМЕНЕНИЯ БИБЛИОТЕК ГРАФИЧЕСКОГО ИНТЕРФЕЙСА В ЯЗЫКЕ ПРОГРАММИРОВАНИЯ PYTHON

©Адиева Г. М. *, ORCID: 0000-0002-3722-4564, SPIN-код: 5030-9170,*
Ошский технологический университет им. М. Адышева,
г.Ош , Кыргызстан, gulzinaadieva@gmail.com
©Биймамат уулу Ш.*, Ошский технологический университет им. М. Адышева,*
г. Ош, Кыргызстан

FEATURES OF USING GUI LIBRARIES IN THE PYTHON PROGRAMMING LANGUAGE

©Adieva G.*, ORCID: 0000-0002-3722-4564, SPIN-code: 5030-9170,*
Osh Technological University named after M. Adyshev,
Osh, Kyrgyzstan, gulzinaadieva@gmail.com
©Beimamat uulu Sh.*, Osh Technological University named after M. Adyshev,*
Osh, Kyrgyzstan

Аннотация. Python является одним из наиболее востребованных языков программирования благодаря простоте синтаксиса, высокой читаемости кода. Для разработки приложений с графическим интерфейсом пользователя в Python доступны специализированные инструменты, обеспечивающие создание визуальных компонентов, обработку событий и управление отрисовкой элементов. В данной статье рассматриваются особенности наиболее популярных GUI-библиотек Tkinter, PyQt6, Kivy, а также инструментов Qt Designer и Qt Creator в контексте проектирования кроссплатформенных настольных приложений. Приведен пример реализации приложения для учета спортсменов с использованием связки Qt Designer и PyQt6.

Abstract. Python is currently one of the most popular programming languages due to its simple syntax and highly readable code. Python offers specialized tools for developing GUI applications, enabling the creation of visual components, event handling, and control over the rendering of elements. This article examines the features of the most popular GUI libraries — Tkinter, PyQt6, and Kivy — as well as Qt Designer and Qt Creator, in the context of developing cross-platform desktop applications. An example of implementing an application for tracking athletes using Qt Designer and PyQt6 is also provided.

Ключевые слова: Python, графические библиотеки, Tkinter, PyQt6, Kivy, PyCharm.

Keywords: Python, graphical libraries, Tkinter, PyQt6, QtDesigner, Qt Creator.

Язык программирования Python — это высокоуровневый, мультипарадигменный и интерпретируемый язык общего назначения, ориентированный на простоту, читаемость кода и высокую производительность разработчика [1]

Главным преимуществом Python является простота и мощная система с большим количеством готовых библиотек и активным сообществом разработчиков, что облегчает решение сложных задач и обучение основам данного языка.

Т. 12. №2 2026
https://doi.org/10.33619/2414-2948/123

В работах И. Г. Гниденко, Ф. Ф. Павлова, Д. Ю. Федорова рассматриваются возможности и краткие характеристики языка программирования Python и его библиотек [2].

Авторы С. Б. Колганова и Н. В. Бужинская в своей работе демонстрируют применение библиотеки Tkinter для реализации простой игры «Крестики-нолики», подчеркивая ее доступность для начинающих разработчиков [3].

Авторы Р. Р. Нурутдинова, Р. Я. Шайхутдинова, Д. С. Зарипова, Г. А. Гареева описывают разработку программного кода для контроля качества деталей. Для решения этой задачи применен библиотека компьютерного зрения OpenCV [4].

Актуальность исследования обусловлена растущим спросом на программное обеспечение в различных областях — образовании, здравоохранении, спорте, производстве. При этом Python остается одним из ведущих языков для прототипирования и разработки GUI-приложений, особенно в условиях ограниченных ресурсов. Одним из ключевых преимуществ языка являются встроенные библиотеки, которые дают возможность создавать различные системы управления, веб приложения, настольные приложения. Библиотека — это набор модулей, функций облегчающих, специфические операции с использованием программирования. Целью исследования является анализ и сопоставление особенностей основных GUI–библиотек Python, а также практическая реализация приложения для учета спортсменов с использованием инструментария Qt Designer и библиотеки PyQt6.

Методы исследования

На начальном этапе использовался метод теоретического анализа, включающий изучение научной и технической литературы. Рассматривались архитектурные особенности изучаемых библиотек. Для систематизации накопленных знаний и выявления ключевых различий между библиотеками был применен метод сравнительного анализа (Таблица). На следующем этапе был реализован метод проектирования. Для проектирования был применен редактор Qt Designer.

Таблица

СРАВНИТЕЛЬНЫЙ АНАЛИЗ GUI – БИБЛИОТЕК PYTHON

Библиотека	*Преимущества*	*Ограничения*
Tkinter	Встроен в стандартную библиотеку Python, минимальные зависимости; простота освоения.	Устаревший внешний вид; ограниченные возможности кастомизации; слабая поддержка современных UI-паттернов.
PyQt6	Современный, профессиональный интерфейс; поддержка кроссплатформенности; широкие возможности визуализации; интеграция с Qt Designer.	Крупный размер установки; лицензионные ограничения.
Kivy	Ориентирован на сенсорные интерфейсы; поддержка мультитач и анимаций; кроссплатформенность; открытая лицензия.	Отсутствие «нативного» внешнего вида на десктопе; менее зрелая документация; ограниченная поддержка стандартных виджетов.

Реализация приложения «Учет спортсменов» с использованием PyQt6 и Qt Designer. PyQt6 представляет собой официальные Python – привязки к фреймворку Qt. Qt является одним из наиболее развитых и широко применяемых инструментариев для создания кроссплатформенных приложений с графическим интерфейсом пользователя, первоначально

Т. 12. №2 2026
https://doi.org/10.33619/2414-2948/123

реализованный на C++. Qt обеспечивает поддержку всех основных платформ включая настольные и мобильные.

Под графическим интерфейсом пользователя (GUI) в современной вычислительной практике понимается программно реализованная среда, обеспечивающая визуальное взаимодействие между пользователем и программным приложением посредством отображаемых на экране интерактивных элементов — таких как окна, кнопки, панели инструментов, значки и др. Управление такими элементами осуществляется с использованием стандартных устройств ввода, что значительно повышает доступность и интуитивность программного обеспечения. В Python PyQt является одной из наиболее зрелых и функционально насыщенных библиотек для проектирования и реализации GUI. Отличительной особенностью является тесная интеграция с инструментом Qt Designer. Визуальный редактор интерфейсов Qt Designer позволяет разработчику конструировать интерфейсы в режиме WYSIWYG (что видишь, то и получаешь), используя метод перетаскивания стандартных и кастомизированных виджетов, а также настраивать их свойства через графическую панель. Для практического применения PyQt в проекте требуется предварительная установка соответствующего пакета командой pip install PyQt6, а также – при необходимости визуального проектирования – установка Qt Designer. После установки библиотек, запускаем дизайнер и создаем приложение.

Для создания окна формы из панели инструментов Display Widgets, расположенной слева, в рабочую область перетаскиваются необходимые элементы, такие как Label, Table Widget, Line Edit, Push Button (Рисунок 1). На панели инструментов справа находится Property Editor, который позволяет изменять свойства приложения.

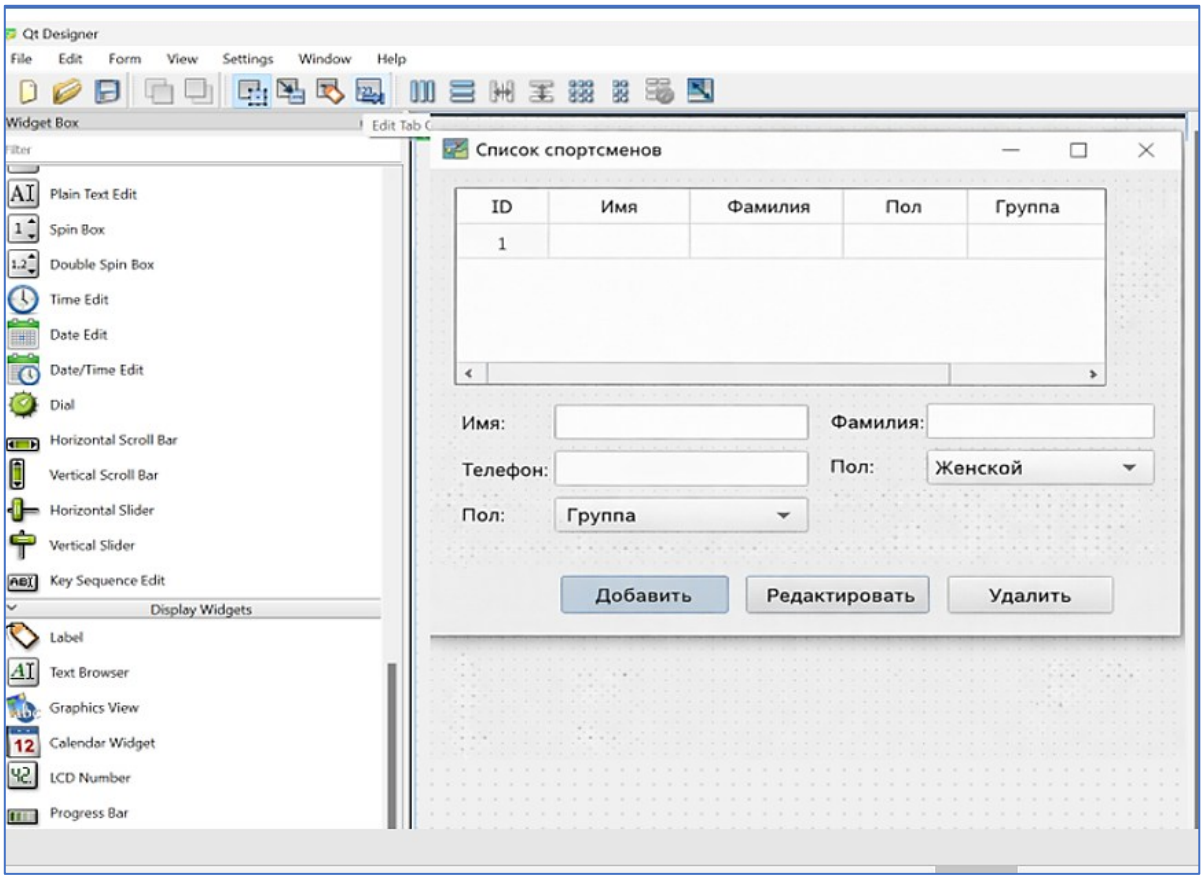

Рисунок 1. Создание формы в Qt Designer

Т. 12. №2 2026
https://doi.org/10.33619/2414-2948/123

Созданный интерфейс сохраняется как athletes.ui. Далее в PyCharm реализуется логика взаимодействия с базой данных SQLite. Скопируем в проект файл формы созданный в Qt Designer athletes.ui. Создаем новый файл пишем следующий код для загрузки *.ui напрямую.

```
import sys
from PyQt6 import uic
from PyQt6.QtWidgets import QApplication, QMainWindow, QTableWidgetItem
import sqlite3

class AthleteApp(QMainWindow):
    def __init__(self):
        super().__init__()
        # Загрузка формы напрямую из .ui
        uic.loadUi("athletes.ui", self)

        # Подключение к базе данных
        self.conn = sqlite3.connect("sportclub.db")
        self.cursor = self.conn.cursor()
        self.create_table()

        # Привязка кнопок к функциям
        self.addButton.clicked.connect(self.add_athlete)
        self.deleteButton.clicked.connect(self.delete_athlete)

        # Загрузка спортсменов в таблицу
        self.load_athletes()

    def create_table(self):
        self.cursor.execute("""
            CREATE TABLE IF NOT EXISTS athletes (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                first_name TEXT,
                last_name TEXT,
                birth_date TEXT,
                gender TEXT,
                group_name TEXT
            )
        """)
        self.conn.commit()

    def add_athlete(self):
        first_name = self.firstNameEdit.text()
        last_name = self.lastNameEdit.text()
        birth_date = self.birthDateEdit.text()
        gender = self.genderCombo.currentText()
        group_name = self.groupCombo.currentText()

        self.cursor.execute(
            "INSERT INTO athletes (first_name, last_name, birth_date, gender, group_name) VALUES (?, ?, ?, ?, ?)",
            (first_name, last_name, birth_date, gender, group_name)
        )
```

```
            self.conn.commit()
            self.load_athletes()

        def delete_athlete(self):
            selected_row = self.tableWidget.currentRow()
            if selected_row >= 0:
                athlete_id = int(self.tableWidget.item(selected_row, 0).text())
                self.cursor.execute("DELETE FROM athletes WHERE id = ?", (athlete_id,))
                self.conn.commit()
                self.load_athletes()

        def load_athletes(self):
            self.cursor.execute("SELECT * FROM athletes")
            rows = self.cursor.fetchall()
            self.tableWidget.setRowCount(len(rows))
            for i, row in enumerate(rows):
                for j, value in enumerate(row):
                    self.tableWidget.setItem(i, j, QTableWidgetItem(str(value)))
    if __name__ == "__main__":
        app = QApplication(sys.argv)
        window = AthleteApp()
        window.show()
        sys.exit(app.exec_())
```

И запускаем файл на выполнение. Откроется окно с формой (Рисунок 2). При настройке в виджетов в коде имена должны совпадать с именами в Qt Designer.

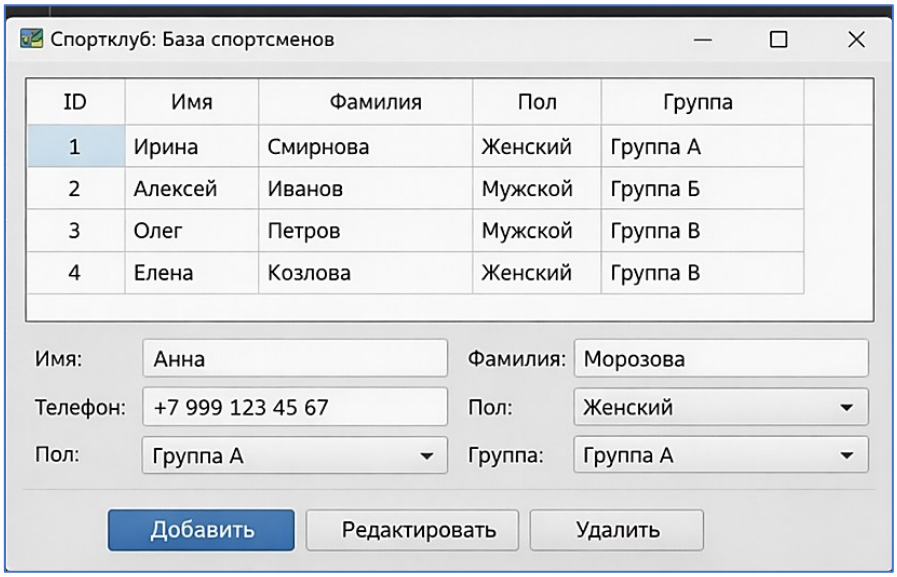

Рисунок 2. Приложение «Учет спортсменов»

Вывод

Проведенный анализ показал, что выбор GUI- библиотеки в Python должен основываться на требованиях к проекту: для простых прототипов и обучения целесообразно использовать

Т. 12. №2 2026
https://doi.org/10.33619/2414-2948/123

Tkinter; для профессиональных, кроссплатформенных настольных приложений – PyQt6 в связке Qt Designer; для мобильных и мультитач-ориентированных приложений – Kivy.

Таким образом, сочетание высокой производительности фреймворка Qt, гибкости Python как языка реализации логики приложения и возможностей декларативного проектирования интерфейсов делает PyQt перспективным инструментом для научного и инженерного программирования, где важны как кроссплатформенность, так и удобство разработки.

Список литературы:

1. Воронов М. В., Пименов В. И., Небаев И. А. Системы искусственного интеллекта. М.: Юрайт, 2025. 268 с.

2. Бухаров Т. А., Нафикова А. Р., Мигранова Е. А. Обзор языка программирования Python и его библиотек // Colloquium-journal. 2019. №3-1 (27). С. 45-52.

3. Колганова С. Б., Бужинская Н. В. Применение языка программирования Python для разработки игр // Наука и перспективы. 2025. №2. С. 112-118.

4. Нурутдинов Р. Р., Шайхутдинов Р. Я., Зарипов Д. С., Гареева Г. А. Разработка приложения для контроля качества деталей на брак // IJAS. 2023. №1. С. 34-41.

References:

1. Voronov, M. V., Pimenov, V. I., & Nebaev, I. A. (2025). Sistemy iskusstvennogo intellekta. Moscow. (in Russian).

2. Bukharov, T. A., Nafikova, A. R., & Migranova, E. A. (2019). Obzor yazyka programmirovaniya Python i ego bibliotek. *Colloquium-journal,* (3-1 (27)), 45-52. (in Russian).

3. Kolganova, S. B., Buzhinskaya, N. V. (2025). Primenenie yazyka programmirovaniya Python dlya razrabotki igr. *Nauka i perspektivy,* (2),112-118. (in Russian).

4. Nurutdinov, R. R., Shaikhutdinov, R. Ya., Zaripov, D. S., & Gareeva, G. A. (2023). Razrabotka prilozheniya dlya kontrolya kachestva detalei na brak. *IJAS,* (1), 34-41. (in Russian).

Поступила в редакцию
12.12.2025 г.

Принята к публикации
17.12.2025 г.

Ссылка для цитирования:

Адиева Г. М. , Биймамат уулу Ш. Особенности применения библиотек графического интерфейса в языке программирования Python // Бюллетень науки и практики. 2026. Т. 12. №2. С. 158-163. https://doi.org/10.33619/2414-2948/123/17

Cite as (APA):

Adieva, G., & Beimamat uulu, Sh. (2026). Features of Using GUI Libraries in the Python Programming Language. *Bulletin of Science and Practice, 12*(2), 158-163. (in Russian). https://doi.org/10.33619/2414-2948/123/17