МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА» (БГТУ им. В.Г. Шухова)
Кафедра программного обеспечения
вычислительной техники и автоматизированных
систем

Лабораторная работа №13
по дисциплине: ООП
тема: «**Знакомство с библиотеками языка Python. PyQT.**»

Выполнил: студент группы ВТ-231
Масленников Д. А.
Проверили:
Буханов Д. Г.

Белгород 2025

Цель работы: приобретение практических навыков создания приложений на языке Python, QT приложения.

Задание:

Для выполнения лабораторной работы требуется установить интерпретатор Python версии 3.6+. Выполнить написание программы-сценария в соответствии с вариантом задания (табл. 1). Провести тестирование. Оформить отчет.

Вариант 14

4	QT-Библиотека (с поискам по тексту)

Код программы:

```
import sys
from PyQt5.QtWidgets import (QApplication, QMainWindow, QWidget, QVBoxLayout,
                             QHBoxLayout, QLabel, QLineEdit, QPushButton,
                             QListWidget, QTextEdit, QMessageBox, QDialog,
                             QDialogButtonBox)


class Book:
    """Класс для представления книги в библиотеке"""
    def __init__(self, title, author, year, content):
        self.title = title
        self.author = author
        self.year = year
        self.content = content


    def __str__(self):
        return f"{self.title} ({self.author}, {self.year})"


    def contains_text(self, text):
        """Проверяет, содержится ли текст в любом поле книги (без учета
регистра)"""
        text_lower = text.lower()
        return (text_lower in self.title.lower() or
                text_lower in self.author.lower() or
                text_lower in str(self.year).lower() or
                text_lower in self.content.lower())


class Library:
    """Класс библиотеки с полной бизнес-логикой"""
    def __init__(self):
        self.books = []
        self.init_sample_books()


    def init_sample_books(self):
        """Инициализация тестовыми данными"""
        sample_books = [
            ("Clean Code", "Роберт Мартин", 2002, "пишем чистый код"),
            ("Па́ттерны проектирования", "Банда четырех", 1994, "стоит знать"),
            ("ООП", "Гради Буч", 1991, "БАЗА")
        ]
        for title, author, year, content in sample_books:
            self.add_book(title, author, year, content)


    def add_book(self, title, author, year, content):
        """Добавление новой книги с валидацией"""
        if not all([title, author, year, content]):
            raise ValueError("Все поля должны быть заполнены")


        try:
            year = int(year)
        except ValueError:
```

```
            raise ValueError("Год должен быть числом")

        self.books.append(Book(title, author, year, content))
        return True

    def search_books(self, text):
        """Поиск книг по тексту (без учета регистра)"""
        if not text:
            return self.books
        return [book for book in self.books if book.contains_text(text)]

    def get_all_books(self):
        """Получение всех книг"""
        return self.books

    def get_book_by_title(self, title):
        """Получение книги по названию"""
        for book in self.books:
            if book.title == title:
                return book
        return None

class LibraryApp(QMainWindow):
    """Класс приложения, отвечающий только за интерфейс"""
    def __init__(self):
        super().__init__()
        self.setWindowTitle("QT-Библиотека")
        self.setGeometry(100, 100, 800, 600)
        self.library = Library()
        self.init_ui()

    def init_ui(self):
        """Инициализация пользовательского интерфейса"""
        main_widget = QWidget()
        self.setCentralWidget(main_widget)

        # Создание элементов интерфейса
        self.search_input = QLineEdit()
        self.search_button = QPushButton("Поиск")
        self.add_book_btn = QPushButton("Добавить книгу")
        self.books_list = QListWidget()
        self.book_details = QTextEdit()

        # Настройка элементов
        self.search_input.setPlaceholderText("Введите текст для поиска...")
        self.book_details.setReadOnly(True)

        # Подключение сигналов
        self.search_button.clicked.connect(self.handle_search)
        self.add_book_btn.clicked.connect(self.show_add_dialog)
        self.books_list.itemClicked.connect(self.show_book_details)
```

```
        # Компоновка интерфейса
        search_layout = QHBoxLayout()
        search_layout.addWidget(self.search_input)
        search_layout.addWidget(self.search_button)
        search_layout.addWidget(self.add_book_btn)

        main_layout = QVBoxLayout()
        main_layout.addLayout(search_layout)
        main_layout.addWidget(QLabel("Список книг:"))
        main_layout.addWidget(self.books_list)
        main_layout.addWidget(QLabel("Содержание:"))
        main_layout.addWidget(self.book_details)

        main_widget.setLayout(main_layout)
        self.update_books_list()

    # Методы работы с интерфейсом
    def update_books_list(self, books=None):
        """Обновление списка книг в интерфейсе"""
        self.books_list.clear()
        books_to_show = books if books is not None else
self.library.get_all_books()
        for book in books_to_show:
            self.books_list.addItem(str(book))

    def show_book_details(self, item):
        """Отображение деталей книги в интерфейсе"""
        title = item.text().split(" (")[0]
        book = self.library.get_book_by_title(title)
        if book:
            details = f"Название: {book.title}\nАвтор: {book.author}\nГод:
{book.year}\n\n{book.content}"
            self.book_details.setPlainText(details)

    def handle_search(self):
        """Обработка поиска (только интерфейс)"""
        search_text = self.search_input.text()
        found_books = self.library.search_books(search_text)

        if search_text and not found_books:
            QMessageBox.information(self, "Поиск", "Книги не найдены.")
        self.update_books_list(found_books)

    def show_add_dialog(self):
        """Показ диалога добавления книги (только интерфейс)"""
        dialog = QDialog(self)
        dialog.setWindowTitle("Добавить книгу")

        # Элементы формы
        title_input = QLineEdit()
        author_input = QLineEdit()
```

```
        year_input = QLineEdit()
        content_input = QTextEdit()

        buttons = QDialogButtonBox(QDialogButtonBox.Ok |
QDialogButtonBox.Cancel)
        buttons.accepted.connect(lambda: self.try_add_book(
            title_input.text(),
            author_input.text(),
            year_input.text(),
            content_input.toPlainText(),
            dialog
        ))
        buttons.rejected.connect(dialog.reject)

        # Компоновка
        layout = QVBoxLayout()
        layout.addWidget(QLabel("Название:"))
        layout.addWidget(title_input)
        layout.addWidget(QLabel("Автор:"))
        layout.addWidget(author_input)
        layout.addWidget(QLabel("Год:"))
        layout.addWidget(year_input)
        layout.addWidget(QLabel("Содержание:"))
        layout.addWidget(content_input)
        layout.addWidget(buttons)

        dialog.setLayout(layout)
        dialog.exec_()

    def try_add_book(self, title, author, year, content, dialog):
        """Попытка добавления книги (только интерфейс)"""
        try:
            self.library.add_book(title, author, year, content)
            self.update_books_list()
            dialog.close()
        except ValueError as e:
            QMessageBox.warning(self, "Ошибка", str(e))

if __name__ == "__main__":
    app = QApplication(sys.argv)
    window = LibraryApp()
    window.show()
    sys.exit(app.exec_())
```

Вывод программы: