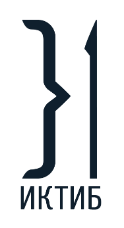
МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное учреждения высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт компьютерных технологий и информационной безопасности

Кафедра математического обеспечения и применения ЭВМ



**ЛАБОРАТОРНАЯ РАБОТА № 6**

**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.

**6 лабораторная работа.**

**Руденко Константин Дмитриевич – 2 вариант**

**Часть 1.**

**Вариант 2:**

**Задание:** Приложение для управления задачами

Описание: Создать приложение для управления задачами.

Компоненты:

- Основные компоненты: Главное окно, панель инструментов.

- Таблицы: Список задач с полями (название, приоритет, статус).

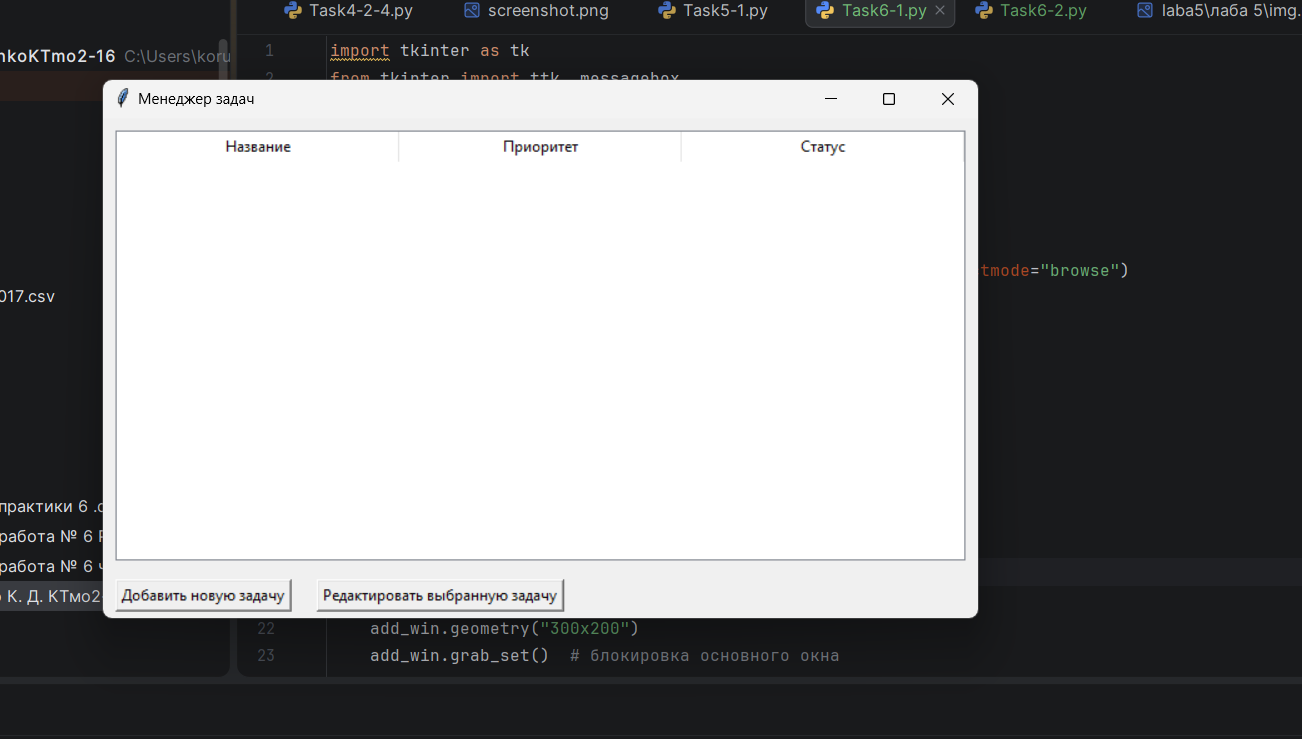
- Списки: Выбор статуса задачи (выполнена, в процессе, отменена).

- Сигналы: Кнопка "Добавить задачу" добавляет новую задачу, а кнопка "Изменить" редактирует выбранную.

**Решение:**

import tkinter as tk  
from tkinter import ttk, messagebox  
  
root = tk.Tk()  
root.title("Менеджер задач")  
root.geometry("700x400")  
  
columns = ("Название", "Приоритет", "Статус")  
tree = ttk.Treeview(root, columns=columns, show="headings", selectmode="browse")  
for col in columns:  
 tree.heading(col, text=col)  
 tree.column(col, width=200, anchor="center")  
tree.pack(fill=tk.BOTH, expand=True, padx=10, pady=10)  
  
priorities = ["Низкий", "Средний", "Высокий"]  
statuses = ["Выполнена", "В процессе", "Отменена"]  
  
  
def add\_task\_window():  
 add\_win = tk.Toplevel(root)  
 add\_win.title("Добавить задачу")  
 add\_win.geometry("300x200")  
 add\_win.grab\_set() # блокировка основного окна  
  
 tk.Label(add\_win, text="Название:").pack(pady=5)  
 entry\_name = tk.Entry(add\_win, width=25)  
 entry\_name.pack()  
 entry\_name.focus()  
  
 tk.Label(add\_win, text="Приоритет:").pack(pady=5)  
 combo\_priority = ttk.Combobox(add\_win, values=priorities, state="readonly")  
 combo\_priority.current(0)  
 combo\_priority.pack()  
  
 tk.Label(add\_win, text="Статус:").pack(pady=5)  
 combo\_status = ttk.Combobox(add\_win, values=statuses, state="readonly")  
 combo\_status.current(1)  
 combo\_status.pack()  
  
 def save\_task():  
 name = entry\_name.get().strip()  
 priority = combo\_priority.get()  
 status = combo\_status.get()  
 if not name:  
 messagebox.showwarning("Ошибка", "Название задачи не может быть пустым!")  
 return  
 tree.insert("", tk.END, values=(name, priority, status))  
 add\_win.destroy()  
  
 tk.Button(add\_win, text="Добавить задачу", command=save\_task).pack(pady=10)  
  
  
def edit\_task\_window():  
 selected = tree.selection()  
 if not selected:  
 messagebox.showinfo("Информация", "Выберите задачу для редактирования")  
 return  
  
 item = tree.item(selected)  
 values = item["values"]  
  
 edit\_win = tk.Toplevel(root)  
 edit\_win.title("Редактировать задачу")  
 edit\_win.geometry("300x200")  
 edit\_win.grab\_set()  
  
 tk.Label(edit\_win, text="Название:").pack(pady=5)  
 entry\_name = tk.Entry(edit\_win, width=25)  
 entry\_name.insert(0, values[0])  
 entry\_name.pack()  
 entry\_name.focus()  
  
 tk.Label(edit\_win, text="Приоритет:").pack(pady=5)  
 combo\_priority = ttk.Combobox(edit\_win, values=priorities, state="readonly")  
 combo\_priority.set(values[1])  
 combo\_priority.pack()  
  
 tk.Label(edit\_win, text="Статус:").pack(pady=5)  
 combo\_status = ttk.Combobox(edit\_win, values=statuses, state="readonly")  
 combo\_status.set(values[2])  
 combo\_status.pack()  
  
 def save\_changes():  
 name = entry\_name.get().strip()  
 priority = combo\_priority.get()  
 status = combo\_status.get()  
 if not name:  
 messagebox.showwarning("Ошибка", "Название задачи не может быть пустым!")  
 return  
 tree.item(selected, values=(name, priority, status))  
 edit\_win.destroy()  
  
 tk.Button(edit\_win, text="Сохранить изменения", command=save\_changes).pack(pady=10)  
  
  
btn\_add\_main = tk.Button(root, text="Добавить новую задачу", command=add\_task\_window)  
btn\_add\_main.pack(side=tk.LEFT, padx=10, pady=5)  
  
btn\_edit\_main = tk.Button(root, text="Редактировать выбранную задачу", command=edit\_task\_window)  
btn\_edit\_main.pack(side=tk.LEFT, padx=10, pady=5)  
  
root.mainloop()

**Примеры как все выглядит:**

****

**Изображение выглядит как программное обеспечение, текст, Мультимедийное программное обеспечение, Значок на компьютере

Содержимое, созданное искусственным интеллектом, может быть неверным.**

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

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, программное обеспечение, Мультимедийное программное обеспечение, Значок на компьютере

Содержимое, созданное искусственным интеллектом, может быть неверным.**

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

Содержимое, созданное искусственным интеллектом, может быть неверным.**

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

Содержимое, созданное искусственным интеллектом, может быть неверным.**

**Часть 2**

Вариант 2 Редактор векторной графики (Работа с графикой)

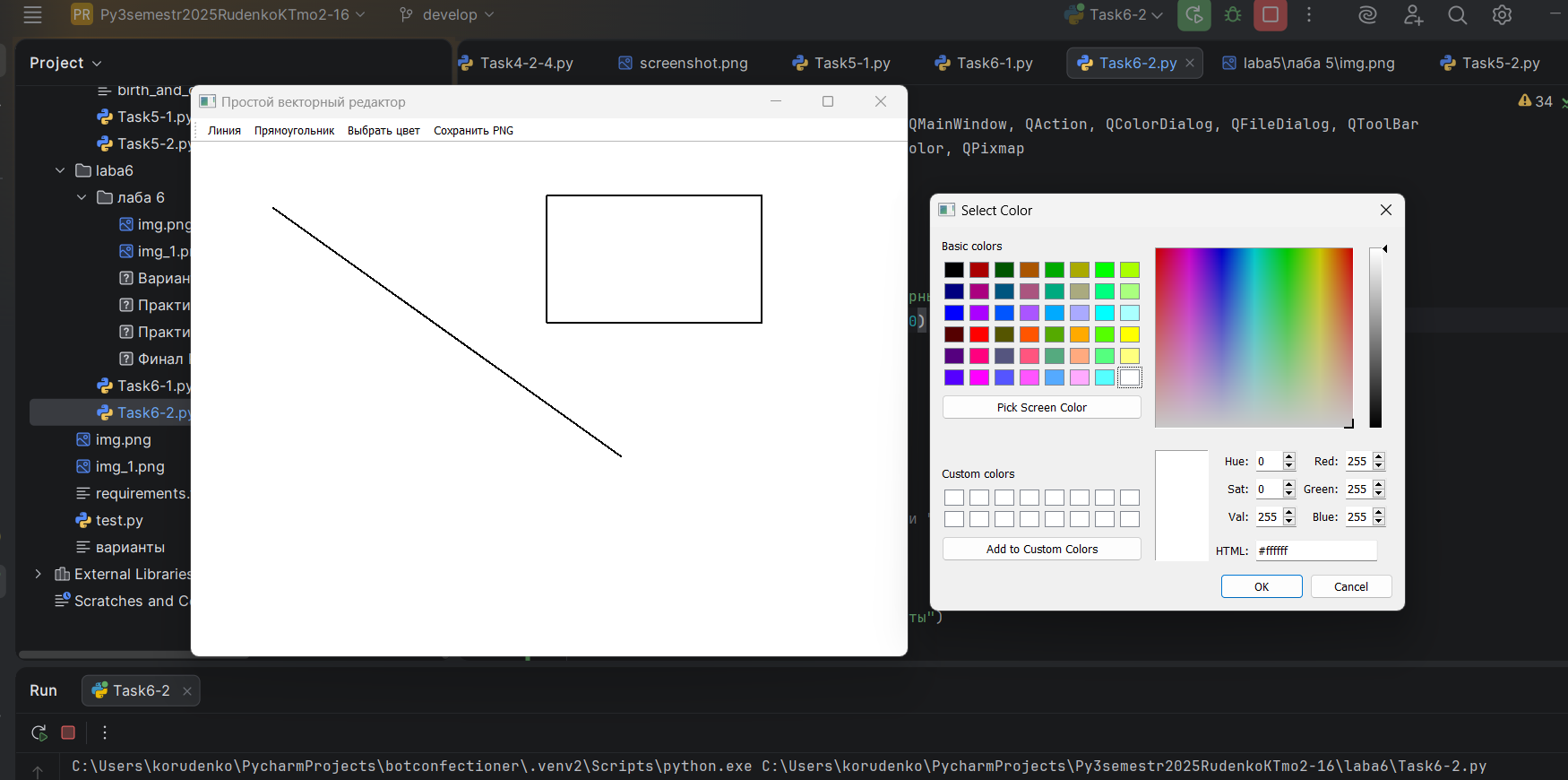
**Цели**: Освоить создание простого редактора векторной графики в PyQt5, включая рисование линий и прямоугольников с выбором цвета.

**Задание**: Создайте приложение, где пользователь может рисовать линии и прямоугольники на холсте, выбирать цвет и сохранять результат в PNG. Используйте мышь для рисования и QToolBar для инструментов.

**Решение:**

import sys  
from PyQt5.QtWidgets import QApplication, QMainWindow, QAction, QColorDialog, QFileDialog, QToolBar  
from PyQt5.QtGui import QPainter, QPen, QColor, QPixmap  
from PyQt5.QtCore import Qt, QPoint, QRect  
  
  
class Canvas(QMainWindow):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.setWindowTitle("Простой векторный редактор")  
 self.setGeometry(100, 100, 800, 600)  
  
 self.pixmap = QPixmap(self.size())  
 self.pixmap.fill(Qt.white)  
 self.drawing = False  
 self.last\_point = QPoint()  
 self.start\_point = QPoint()  
 self.current\_shape = "Линия"  
 self.color = QColor(Qt.black)  
  
 self.toolbar = QToolBar("Инструменты")  
 self.addToolBar(self.toolbar)  
  
 self.line\_action = QAction("Линия", self)  
 self.line\_action.triggered.connect(lambda: self.set\_shape("Линия"))  
 self.toolbar.addAction(self.line\_action)  
  
 self.rect\_action = QAction("Прямоугольник", self)  
 self.rect\_action.triggered.connect(lambda: self.set\_shape("Прямоугольник"))  
 self.toolbar.addAction(self.rect\_action)  
  
 self.color\_action = QAction("Выбрать цвет", self)  
 self.color\_action.triggered.connect(self.choose\_color)  
 self.toolbar.addAction(self.color\_action)  
  
 self.save\_action = QAction("Сохранить PNG", self)  
 self.save\_action.triggered.connect(self.save\_image)  
 self.toolbar.addAction(self.save\_action)  
  
 def set\_shape(self, shape):  
 self.current\_shape = shape  
  
 def choose\_color(self):  
 color = QColorDialog.getColor()  
 if color.isValid():  
 self.color = color  
  
 def save\_image(self):  
 filename, \_ = QFileDialog.getSaveFileName(self, "Сохранить изображение", "", "PNG Files (\*.png)")  
 if filename:  
 self.pixmap.save(filename, "PNG")  
  
 def mousePressEvent(self, event):  
 if event.button() == Qt.LeftButton:  
 self.start\_point = event.pos()  
 self.drawing = True  
  
 def mouseMoveEvent(self, event):  
 if self.drawing:  
 self.last\_point = event.pos()  
 self.update() # вызываем paintEvent для динамического отображения  
  
 def mouseReleaseEvent(self, event):  
 if event.button() == Qt.LeftButton and self.drawing:  
 painter = QPainter(self.pixmap)  
 pen = QPen(self.color, 2)  
 painter.setPen(pen)  
  
 if self.current\_shape == "Линия":  
 painter.drawLine(self.start\_point, event.pos())  
 elif self.current\_shape == "Прямоугольник":  
 rect = QRect(self.start\_point, event.pos())  
 painter.drawRect(rect.normalized())  
  
 self.drawing = False  
 self.update()  
  
 def paintEvent(self, event):  
 canvas\_painter = QPainter(self)  
 canvas\_painter.drawPixmap(self.rect(), self.pixmap)  
  
 if self.drawing:  
 pen = QPen(self.color, 2, Qt.SolidLine)  
 canvas\_painter.setPen(pen)  
 if self.current\_shape == "Линия":  
 canvas\_painter.drawLine(self.start\_point, self.last\_point)  
 elif self.current\_shape == "Прямоугольник":  
 rect = QRect(self.start\_point, self.last\_point)  
 canvas\_painter.drawRect(rect.normalized())  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 app = QApplication(sys.argv)  
 window = Canvas()  
 window.show()  
 sys.exit(app.exec\_())

**Примеры скриншотов:**

****

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

Содержимое, созданное искусственным интеллектом, может быть неверным.**

**Изображение выглядит как диаграмма, снимок экрана, текст, линия

Содержимое, созданное искусственным интеллектом, может быть неверным.**

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

Содержимое, созданное искусственным интеллектом, может быть неверным.**

Ссылка на гитхаб: [all/Py3semestr2025RudenkoKTmo2-16/laba3 at master · Kosten-73/all](https://github.com/Kosten-73/all/tree/master/Py3semestr2025RudenkoKTmo2-16/laba3)