

Разработка скрипта на Python для автоматической маркировки изображений

Акентьев Данила Денисович

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

Данная работа описывает разработку приложения для наложения логотипа на изображения с использованием библиотеки Python Pillow и графического интерфейса Tkinter. Приложение позволяет пользователю выбрать исходное изображение и логотип, автоматически масштабирует логотип до 10% от размера изображения и размещает его в нижнем правом углу. Разработанное приложение предоставляет простой и удобный интерфейс для автоматизированного добавления логотипов, что может быть полезно для различных задач, связанных с обработкой изображений.

Ключевые слова: Python, Pillow, Tkinter, обработка изображений, наложение логотипа, графический интерфейс, автоматизация, масштабирование, GUI.

Development of a Python script for automatic labeling of images

Akentev Danila Denisovich

Sholom-Aleichem Priamursky State University

Student

Abstract

This paper describes the development of an application for applying a logo to images using the Python Pillow library and the Tkinter GUI. The application allows the user to select the original image and logo, automatically scales the logo to 10% of the image size and places it in the lower right corner. The developed application provides a simple and user-friendly interface for the automated addition of logos, which can be useful for various tasks related to image processing.

Keywords: Python, Pillow, Tkinter, image processing, logo overlay, graphical interface, automation, scaling, GUI.

1 Введение

1.1 Актуальность

Автоматизация задач по обработке изображений, таких как наложение логотипов, является актуальной задачей во многих областях, включая веб-дизайн, рекламу, и создание контента. Ручное добавление логотипов на множество изображений трудоемко и занимает много времени. Разработка автоматизированного приложения значительно повышает эффективность и скорость работы.

1.2 Обзор исследований

Для ознакомления с языком программирования Python и его применением в разработке графических интерфейсов, рекомендуем обратиться к следующим источникам. В.А. Машков и В.И. Литвиненко рассматривают применение Python для задач самодиагностики на системном уровне [1]. Г.Д. Бухарова и П.С. Комельских подчеркивают важность и необходимость включения Python в учебный процесс [2]. Г.С. Сейдаметов демонстрирует особенности использования Python в подготовке инженеров-программистов [3], а Э.А. Усеинов иллюстрирует применение объектно-ориентированного программирования в рамках изучения Python [4]. Разработка приложений с графическим пользовательским интерфейсом (GUI) на Python с использованием Tkinter также активно исследовалась. Например, А.С. Матвеева в своей статье рассматривает применение паттерна «Команда» для создания модульных и структурированных приложений на Tkinter, демонстрируя повышение гибкости, расширяемости, надежности и удобства использования [5].

1.3 Цель исследования

Цель данной работы – разработка и реализация простого в использовании приложения на языке Python с графическим интерфейсом, позволяющего автоматически накладывать логотип на изображения, масштабируя его до 10% от размера изображения и размещая в нижнем правом углу.

2 Материалы и методы

Для реализации приложения были использованы следующие материалы и методы:

- Язык программирования: Python.
- Библиотека обработки изображений: Pillow (PIL).
- Библиотека для создания графического интерфейса: Tkinter.
- Методы: Чтение изображений с использованием Pillow, автоматическое масштабирование логотипа, наложение логотипа на изображение с учетом прозрачности (если доступно), сохранение результирующего изображения, создание простого интуитивно понятного графического интерфейса с помощью Tkinter.

3 Результаты и обсуждения

1. Импорт необходимых библиотек

В начале кода импортируются необходимые библиотеки(рис.1).

```
import tkinter as tk
from tkinter import filedialog
from PIL import Image, ImageTk
import os
```

Рисунок 1. Импорт библиотек

tkinter и filedialog используются для создания GUI и диалогового окна выбора файлов.

PIL (Pillow) — библиотека для работы с изображениями. Image используется для загрузки и обработки изображений, ImageTk для отображения изображений в GUI.

os необходим для работы с файловой системой (создание директории и сохранение файла).

2. Класс LogoApp — основной класс приложения (рис.2).

Создается класс LogoApp, который инкапсулирует всю логику приложения. В методе `__init__` инициализируются переменные:

```
class LogoApp:
    def __init__(self, master):
        self.master = master
        master.title("Наложение логотипа")

        self.image_path = None
        self.logo_path = None
        self.image_preview = None
        self.preview_image = None

        self.image_width = 0
        self.image_height = 0
        self.logo_width = 0
        self.logo_height = 0

        tk.Button(master, text="Выбрать фотографию", command=self.choose_image).pack(pady=10)
        tk.Button(master, text="Выбрать логотип", command=self.choose_logo).pack(pady=10)
        tk.Button(master, text="Наложить логотип", command=self.add_logo).pack(pady=10)

        self.preview_label = tk.Label(master)
        self.preview_label.pack()
        self.status_label = tk.Label(master, text="")
        self.status_label.pack(pady=5)
```

Рисунок 2. Основной класс приложения

Инициализируются пути к файлам (`image_path`, `logo_path`), переменные для хранения превью изображений, переменные для координат логотипа, а также переменные для хранения размеров изображений. Далее создаются элементы GUI: кнопки для выбора файлов, поле для отображения превью и метка для отображения статуса.

3. Функции для выбора файлов и загрузки размеров (рис.3).

Функции `choose_image` и `choose_logo` открывают диалоговые окна для выбора файлов изображения и логотипа соответственно, используя `filedialog.askopenfilename`. Функции `load_image_size` и `load_logo_size` загружают изображения с помощью `Image.open` и сохраняют их размеры в соответствующие переменные. Обработка исключений обеспечивает устойчивость к ошибкам при открытии файлов.

```

def choose_image(self):
    self.image_path = self.ask_for_file("Выбрать фотографию", "*.png *.jpg *.jpeg *.gif *.bmp")
    if self.image_path:
        self.load_image_size()
        self.show_preview()

def load_image_size(self):
    try:
        img = Image.open(self.image_path)
        self.image_width, self.image_height = img.size
    except Exception as e:
        self.show_status(f"Ошибка загрузки изображения: {e}")

def choose_logo(self):
    self.logo_path = self.ask_for_file("Выберите логотип", "*.png *.jpg *.jpeg *.gif *.bmp")
    if self.logo_path:
        self.load_logo_size()
        self.update_preview()

def load_logo_size(self):
    try:
        logo = Image.open(self.logo_path)
        self.logo_width, self.logo_height = logo.size
    except Exception as e:
        self.show_status(f"Ошибка загрузки логотипа: {e}")

```

Рисунок 3. Функции выбора файлов и загрузки размеров

4. Функция для обновления превью (рис.4).

Функция `update_preview` является ключевой. Она загружает изображения, масштабирует логотип до 10% от размера основного изображения, вычисляет координаты для размещения логотипа в нижнем правом углу, накладывает логотип на основное изображение с помощью `base_img.paste(logo, (x, y), logo)`, создает превью и отображает его в GUI.

```

def update_preview(self, *args):
    if not self.image_path or not self.logo_path:
        return

    try:
        base_img = Image.open(self.image_path)
        logo = Image.open(self.logo_path).convert("RGBA")

        logo_width = int(base_img.width * 0.1)
        logo_height = int(base_img.height * 0.1)
        logo = logo.resize((logo_width, logo_height), Image.LANCZOS)

        x = base_img.width - logo_width - 10
        y = base_img.height - logo_height - 10

        base_img.paste(logo, (x, y), logo)
        base_img.thumbnail((500, 500))
        self.preview_image = ImageTk.PhotoImage(base_img)
        self.preview_label.config(image=self.preview_image)
        self.preview_label.image = self.preview_image

    except Exception as e:
        self.show_status(f"Ошибка обновления превью: {e}")

```

Рисунок 4. Функция обновления превью

Обработка исключений предотвращает сбои в работе приложения в случае ошибок.

5. Функция для сохранения результата (рис.5).

Функция `add_logo` выполняет те же действия, что и `update_preview`, но вместо отображения превью сохраняет результирующее изображение в файл.

Она создает директорию `with_logo`, если она не существует, и сохраняет изображение с помощью `base_img.save`.

```
def add_logo(self):
    self.show_status("")
    try:
        base_img = Image.open(self.image_path)
        logo = Image.open(self.logo_path).convert("RGBA")

        logo_width = int(base_img.width * 0.1)
        logo_height = int(base_img.height * 0.1)
        logo = logo.resize((logo_width, logo_height), Image.LANCZOS)

        x = base_img.width - logo_width - 10
        y = base_img.height - logo_height - 10

        base_img.paste(logo, (x, y), logo)

        output_path = os.path.join("with_logo", os.path.basename(self.image_path).split(".")[0] + "_with_logo.png")
        os.makedirs("with_logo", exist_ok=True)
        base_img.save(output_path)
        self.show_status(f"Логотип добавлен: {output_path}")

    except FileNotFoundError:
        self.show_status("Ошибка: Файл не найден.")
    except Exception as e:
        self.show_status(f"Ошибка: {e}")
```

Рисунок 5. Функция сохранения результата

6. Запуск приложения (рис.6).

В конце кода создается главный объект окна `tk.Tk()` и экземпляр класса `LogoApp`, запускающий приложение:

```
root = tk.Tk()
app = LogoApp(root)
root.mainloop()
```

Рисунок 6. Запуск приложения

7. Внешний вид приложения и результат работы (рис.7-).

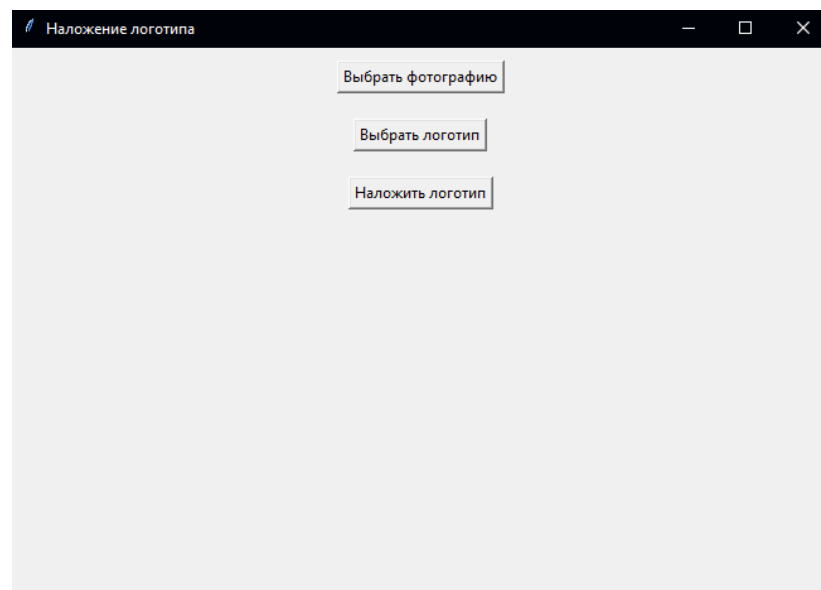


Рисунок 7. Начальный экран

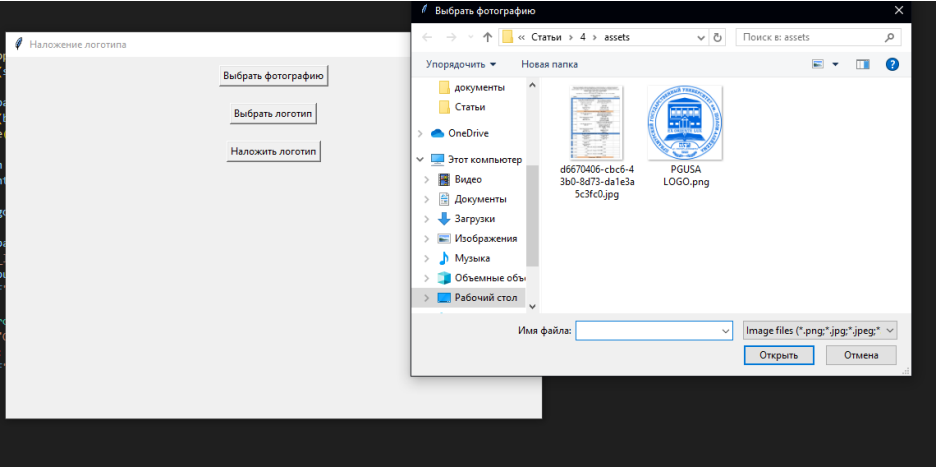


Рисунок 8. Выбор файлов

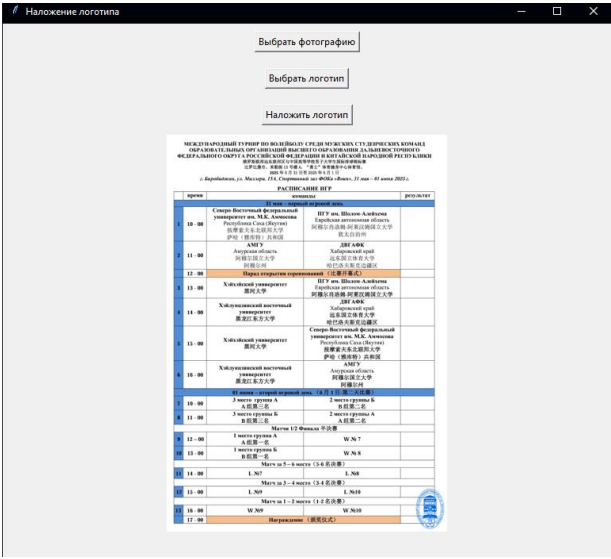


Рисунок 9. Предпросмотр изображения

МЕЖДУНАРОДНЫЙ ТУРНИР ПО ВОЛЕЙБОЛУ СРЕДИ МУЖСКИХ СТУДЕНЧЕСКИХ КОМАНД ОБРАЗОВАТЕЛЬНЫХ ОРГАНИЗАЦИЙ ВЫСШЕГО ОБРАЗОВАНИЯ ДАЛЬНЕВОСТОЧНОГО ФЕДЕРАЛЬНОГО ОКРУГА РОССИЙСКОЙ ФЕДЕРАЦИИ И КИТАЙСКОЙ НАРОДНОЙ РЕСПУБЛИКИ
俄罗斯联邦远东联邦区与中国高等院校男子大学生国际排球锦标赛
比罗比詹市，米勤街15号楼A，“勇士”体育健身中心体育馆。
2025年5月31日至2025年6月1日
г. Биробиджан, ул. Миллера, 15А, Спортивный зал ФОКа «Воин», 31 мая – 01 июня 2025 г.

РАСПИСАНИЕ ИГР

время	команды	результат
31 мая – первый игровой день		
1 10 - 00	Северо-Восточный федеральный университет им. М.К. Аммосова Республика Саха (Якутия) 按察素夫东北联邦大学 萨哈（雅库特）共和国	ПГУ им. Шолом-Алейхема Еврейская автономная область 阿穆尔肖洛姆-阿莱汉姆国立大学 犹太自治州
2 11 - 00	АМГУ Амурская область 阿穆尔国立大学 阿穆尔州	ДВГАФК Хабаровский край 远东国立体育大学 哈巴洛夫斯克边疆区
12 - 00	Парад открытия соревнований. (比赛开幕式)	
3 13 - 00	Хэйхэйский университет 黑河大学	ПГУ им. Шолом-Алейхема Еврейская автономная область 阿穆尔肖洛姆-阿莱汉姆国立大学
4 14 - 00	Хэйлунианьский восточный университет 黑龙江东方大学	ДВГАФК Хабаровский край 远东国立体育大学 哈巴洛夫斯克边疆区
5 15 - 00	Хэйхэйский университет 黑河大学	Северо-Восточный федеральный университет им. М.К. Аммосова Республика Саха (Якутия) 按察素夫东北联邦大学 萨哈（雅库特）共和国
6 16 - 00	Хэйлунианьский восточный университет 黑龙江东方大学	АМГУ Амурская область 阿穆尔国立大学 阿穆尔州
01 июня – второй игровой день. (6月1日-第二天比赛)		
7 10 - 00	3 место группы А A组第三名	2 место группы Б B组第二名
8 11 - 00	3 место группы Б B组第三名	2 место группы А A组第二名
Матчи 1/2 Финала 半决赛		
9 12 - 00	1 место группы А A组第一名	W № 7
10 13 - 00	1 место группы Б B组第一名	W № 8
Матч за 5 – 6 место (5-6名决赛)		
11 14 - 00	L №7	L №8
Матч за 3 – 4 место (3-4名决赛)		
12 15 - 00	L №9	L №10
Матч за 1 – 2 место (1-2名决赛)		
13 16 - 00	W №9	W №10
17 - 00	Награждение (颁奖仪式)	

Рисунок 10. Готовый результат

4 Выводы

В результате работы было разработано приложение на языке Python, которое успешно выполняет задачу автоматического наложения логотипа на изображения. Приложение обладает простым и интуитивно понятным графическим интерфейсом, что упрощает его использование. Автоматизация процесса наложения логотипов позволяет значительно повысить эффективность работы и сэкономить время, что делает разработанное приложение полезным инструментом для различных задач, связанных с обработкой изображений. Дальнейшие исследования могут быть направлены на расширение функциональности приложения, например, добавление возможности изменения размера и положения логотипа, поддержки различных форматов файлов, и интеграции с другими инструментами обработки изображений.

Библиографический список

1. Машков В.А., Литвиненко В.И. Использование языка программирования python 3 и системы компьютерной алгебры sympy на факультативных занятиях по теории чисел // Электротехнические и компьютерные системы. 2017. С. 48-54
2. Бухарова Г.Д., Комельских П.С. Важность и необходимость внедрения

- языка программирования python в процесс обучения студентов // Новые информационные технологии в образовании Материалы VII международной научно-практической конференции. 2014. С. 40-42.
3. Сейдаметов Г.С. Особенности использования языка программирования python в подготовке будущих инженеров-программистов // International scientific review. 2019. №. 9(10). С. 50-51
 4. Усеинов Э.А. Объектно-ориентированное программирование в рамках дисциплины «язык программирования python» // Ученые записки крымского инженерно-педагогического университета. 2012. №. 34. С. 157-160.
 5. Матвеева А. С. Реализация паттерна поведения «Команда» в графическом интерфейсе пользователя на Python // Постулат. 2025. №1.