Тема: Журнал личных учебных достижений
Используется язык программирования Python версии 3.8 или выше.
Графический интерфейс создаётся с помощью стандартной библиотеки tkinter.
База данных — SQLite (встроена в Python, отдельная установка не требуется).
Экспорт отчёта выполняется с помощью библиотеки python-docx.
Допускается использование модуля json для загрузки справочника типов.
Запрещено использовать любые другие сторонние библиотеки.
Запрещено использовать регулярные выражения.
Запрещено применять сложные объектно-ориентированные конструкции — допускается только один основной класс или вообще без классов.
Все надписи в интерфейсе и в коде должны быть на русском языке.
В исходном коде не должно быть комментариев.
Файлы проекта
В одной папке должны находиться два файла:
main.py — основной файл программы.
types.json — файл со списком типов достижений.
Содержимое файла types.json:
```
[
  "Олимпиада",
  "Сертификат",
  "Проект",
  "Экзамен",
  "Конференция"
]
```
Файл должен быть сохранён в кодировке UTF-8.
Шаг 1. Создание главного окна
Откройте файл main.py и введите следующий код:
```
import tkinter as tk
from tkinter import ttk
root = tk.Tk()
root.title("Журнал достижений")
root.geometry("600x400")
notebook = ttk.Notebook(root)
notebook.pack(fill="both", expand=True)
tab_add = ttk.Frame(notebook)
tab_list = ttk.Frame(notebook)
notebook.add(tab_add, text="Добавить")
notebook.add(tab_list, text="Мои достижения")
root.mainloop()
```
Проверка: запустите программу. Должно появиться окно с двумя вкладками — «Добавить» и «Мои достижения».
Шаг 2. Добавление формы ввода
Добавьте в начало файла (после импортов) следующую функцию:
```
def create_add_form(parent):
    tk.Label(parent, text="Название:").pack(anchor="w", padx=10, pady=5)
    name_entry = tk.Entry(parent, width=50)
    name_entry.pack(padx=10, pady=5)
    tk.Label(parent, text="Дата (ГГГГ-ММ-ДД):").pack(anchor="w", padx=10, pady=5)
    date_entry = tk.Entry(parent, width=20)
    date_entry.pack(padx=10, pady=5)
    tk.Label(parent, text="Тип:").pack(anchor="w", padx=10, pady=5)
    type_combo = ttk.Combobox(parent, values=["Олимпиада", "Сертификат", "Проект", "Экзамен", "Конференция"], state="readonly")
    type_combo.pack(padx=10, pady=5)
    type_combo.set("Олимпиада")
    tk.Label(parent, text="Уровень:").pack(anchor="w", padx=10, pady=5)
    level_combo = ttk.Combobox(parent, values=["локальный", "региональный", "национальный"], state="readonly")
    level_combo.pack(padx=10, pady=5)
    level_combo.set("локальный")
    tk.Label(parent, text="Описание:").pack(anchor="w", padx=10, pady=5)
    desc_text = tk.Text(parent, height=4, width=50)
    desc_text.pack(padx=10, pady=5)
    save_btn = tk.Button(parent, text="Сохранить")
    save_btn.pack(pady=10)
    return name_entry, date_entry, type_combo, level_combo, desc_text, save_btn
```
Теперь замените строку:
```
tab_add = ttk.Frame(notebook)
```
```
tab_add = ttk.Frame(notebook)
name_entry, date_entry, type_combo, level_combo, desc_text, save_btn = create_add_form(tab_add)
```
Проверка: на вкладке «Добавить» должны отображаться поля ввода и кнопка «Сохранить».
Шаг 3. Создание базы данных
Добавьте в начало файла (после импортов):
```
import sqlite3
import os
```
Добавьте функцию:
```
def init_db():
    if not os.path.exists("достижения.db"):
        conn = sqlite3.connect("достижения.db")
        cur = conn.cursor()
        cur.execute("""
            CREATE TABLE достижения (
                id INTEGER PRIMARY KEY,
                название TEXT NOT NULL,
                дата TEXT NOT NULL,
                тип TEXT NOT NULL,
                уровень TEXT NOT NULL,
                описание TEXT
            )
        """)
        conn.commit()
        conn.close()
```
Вызовите эту функцию перед строкой root.mainloop():
```
init_db()
```
Проверка: после запуска программы в той же папке должен появиться файл достижения.db.
Шаг 4. Сохранение данных в базу
Добавьте функцию:
```
def save_to_db(name, date, typ, level, desc):
    conn = sqlite3.connect("достижения.db")
    cur = conn.cursor()
    cur.execute("INSERT INTO достижения (название, дата, тип, уровень, описание) VALUES (?, ?, ?, ?, ?)",
                (name, date, typ, level, desc))
    conn.commit()
    conn.close()
```
Подключите кнопку «Сохранить» к обработчику:
```
def on_save():
```

```
    name = name_entry.get().strip()
    date = date_entry.get().strip()
    typ = type_combo.get()
    level = level_combo.get()
    desc = desc_text.get("1.0", "end-1c").strip()
    if name and date:
        save_to_db(name, date, typ, level, desc)
        print("Сохранено:", name)
    else:
        print("Заполните название и дату")
save_btn.config(command=on_save)
```

Проверка: введите данные, нажмите «Сохранить». В консоли должно появиться сообщение. Проверьте базу с помощью любого SQLite-просмотрщика.

Шаг 5. Отображение списка достижений

Добавьте функцию:

```
def load_records():
    conn = sqlite3.connect("достижения.db")
    cur = conn.cursor()
    cur.execute("SELECT дата, название, тип, уровень FROM достижения ORDER BY дата DESC")
    rows = cur.fetchall()
    conn.close()
    return rows
```

На вкладке «Мои достижения» создайте список:

```
listbox = tk.Listbox(tab_list, width=80, height=15)
listbox.pack(padx=10, pady=10)
def refresh_list():
    listbox.delete(0, tk.END)
    records = load_records()
    for date, name, typ, level in records:
        listbox.insert(tk.END, f"[{date}] {name} ({typ}, {level})")
refresh_list()
```

Проверка: переключитесь на вкладку «Мои достижения» — там должен отображаться список записей.

Шаг 6. Экспорт отчёта в Word

Добавьте в начало файла:

```
from docx import Document
```

Добавьте функцию:

```
def load_records_with_desc():
    conn = sqlite3.connect("достижения.db")
    cur = conn.cursor()
    cur.execute("SELECT дата, название, тип, уровень, описание FROM достижения ORDER BY дата DESC")
    rows = cur.fetchall()
    conn.close()
    return rows
```

Добавьте функцию экспорта:

```
def export_to_word():
    doc = Document()
    doc.add_heading("Личные учебные достижения", 0)
    records = load_records_with_desc()
    for date, name, typ, level, desc in records:
        p = doc.add_paragraph()
        p.add_run(name).bold = True
        p.add_run(f" — {date}").italic = True
        p.add_run(f" ({typ}, {level})")
        if desc:
            doc.add_paragraph(desc)
    doc.save("достижения.docx")
    print("Отчёт сохранён: достижения.docx")
```

Добавьте кнопку на вкладку «Мои достижения»:

```
export_btn = tk.Button(tab_list, text="Экспорт в Word", command=export_to_word)
export_btn.pack(pady=10)
```

Проверка: нажмите кнопку — в папке появится файл достижения.docx. Откройте его — форматирование должно быть корректным.

Шаг 7. Загрузка типов из JSON-файла

Добавьте в начало файла:

```
import json
```

Добавьте функцию:

```
def load_types():
    try:
        with open("types.json", "r", encoding="utf-8") as f:
            return json.load(f)
    except:
        return ["Олимпиада", "Сертификат", "Проект", "Экзамен", "Конференция"]
```

В функции create_add_form замените строку:

```
type_combo = ttk.Combobox(parent, values=["Олимпиада", "Сертификат", "Проект", "Экзамен", "Конференция"], state="readonly")
types = load_types()
type_combo = ttk.Combobox(parent, values=types, state="readonly")
```

Проверка: измените содержимое файла types.json — список типов в программе должен обновиться.