استخراج لینک‌های قابل دانلود و اضافه کردن به IDM

یک برنامه پایتون با رابط گرافیکی برای استخراج لینک‌های قابل دانلود از یک سایت و اضافه کردن آنها به صف دانلود IDM

معرفی برنامه

این برنامه به شما امکان می‌دهد تا با وارد کردن آدرس یک وبسایت، تمام لینک‌های قابل دانلود را استخراج کرده و مستقیماً به نرم‌افزار Internet Download Manager (IDM) اضافه کنید. برنامه دارای رابط گرافیکی کاربرپسند بوده و استفاده از آن بسیار ساده است.

ویژگی‌های برنامه

- رابط گرافیکی کاربرپسند
- استخراج خودکار لینک‌های قابل دانلود
- امکان فیلتر کردن فایل‌ها بر اساس پسوند
- اضافه کردن مستقیم به صف دانلود IDM
- نمایش پیشرفت فرآیند استخراج
- امکان تنظیم پروکسی (اختیاری)

پیش‌نیازها

- پایتون نسخه 3.6 یا بالاتر
- نرم‌افزار Internet Download Manager
- کتابخانه‌های: requests, beautifulsoup4, pywin32
- کتابخانه Tkinter (معمولاً همراه پایتون نصب می‌شود)

نمای برنامه

استخراج لینک‌های قابل دانلود و اضافه به IDM

آدرس سایت:

https://example.com [استخراج]

پسوندهای مورد نظر (با کاما جدا کنید): zip,rar,pdf,mp3,mp4,exe

☐ استفاده از پروکسی

آدرس:پورت

[INFO] در حال استخراج لینک‌ها از https://example.com

[+] لینک یافت شد: https://example.com/files/document.pdf

[+] لینک یافت شد: https://example.com/download/app.zip

[+] لینک یافت شد: https://example.com/media/sample.mp3

[INFO] اضافه کردن لینک‌ها به IDM...

۶۶٪ تکمیل شده - ۳ لینک از ۳ سایت

[توقف] [اضافه به IDM]

کد برنامه

```
import tkinter as tk
import tkinter as tk
from tkinter import ttk, scrolledtext, messagebox
import threading
import requests
from bs4 import BeautifulSoup
import win32com.client
import re
import os
import time
from urllib.parse import urljoin, urlparse


class IDMDownloaderApp:
    def __init__(self, root):
        self.root = root
        self.root.title("استخراج لینک‌های قابل دانلود و اضافه به IDM")
        self.root.geometry("700x600")
        self.root.configure(bg="#f0f2f5")

        # تنظیم فونت برای بهبود نمایش متون فارسی
        default_font = ('Tahoma', 10)

        # تنظیم متغیرهای مورد نیاز
        self.url_var = tk.StringVar()
        self.extensions_var = tk.StringVar(value="zip,rar,pdf,mp3,mp4,exe")
        self.proxy_var = tk.StringVar()
        self.use_proxy_var = tk.BooleanVar(value=False)
        self.stop_extraction = threading.Event()
        self.extracted_links = []

        # ایجاد فریم اصلی
        main_frame = tk.Frame(root, bg="#f0f2f5", padx=20, pady=20)
        main_frame.pack(fill=tk.BOTH, expand=True)

        # بخش ورود آدرس سایت
        url_frame = tk.Frame(main_frame, bg="#f0f2f5")
        url_frame.pack(fill=tk.X, pady=5)

        tk.Label(url_frame, text="آدرس سایت:", bg="#f0f2f5", font=default_font).pack(anchor=tk.W)

        url_input_frame = tk.Frame(url_frame, bg="#f0f2f5")
        url_input_frame.pack(fill=tk.X, pady=2)

        self.url_entry = tk.Entry(url_input_frame, textvariable=self.url_var, font=default_font)
        self.url_entry.pack(side=tk.LEFT, fill=tk.X, expand=True, padx=(0, 5))

        self.extract_btn = tk.Button(url_input_frame, text="استخراج",
                                     command=self.start_extraction, bg="#4a7aff", fg="white",
                                     font=default_font, padx=10)
        self.extract_btn.pack(side=tk.RIGHT)

        # بخش پسوندهای مورد نظر
        ext_frame = tk.Frame(main_frame, bg="#f0f2f5")
        ext_frame.pack(fill=tk.X, pady=5)

        tk.Label(ext_frame, text="پسوندهای مورد نظر (با کاما جدا کنید):",
                bg="#f0f2f5", font=default_font).pack(anchor=tk.W)

        self.ext_entry = tk.Entry(ext_frame, textvariable=self.extensions_var, font=default_font)
        self.ext_entry.pack(fill=tk.X, pady=2)

        # بخش پروکسی
        proxy_frame = tk.Frame(main_frame, bg="#f0f2f5")
        proxy_frame.pack(fill=tk.X, pady=5)

        proxy_check = tk.Checkbutton(proxy_frame, text="استفاده از پروکسی",
                                    variable=self.use_proxy_var, bg="#f0f2f5",
                                    font=default_font, command=self.toggle_proxy)
        proxy_check.pack(anchor=tk.W)

        self.proxy_entry = tk.Entry(proxy_frame, textvariable=self.proxy_var,
                                  font=default_font, state=tk.DISABLED)
        self.proxy_entry.pack(fill=tk.X, pady=2)

        # بخش نمایش لاگ
        log_frame = tk.Frame(main_frame, bg="#f0f2f5")
        log_frame.pack(fill=tk.BOTH, expand=True, pady=5)

        tk.Label(log_frame, text="گزارش عملیات:", bg="#f0f2f5", font=default_font).pack(anchor=tk.W)

        self.log_area = scrolledtext.ScrolledText(log_frame, font=('Consolas', 9),
                                               height=10, wrap=tk.WORD)
        self.log_area.pack(fill=tk.BOTH, expand=True, pady=2)

        # بخش نمایش پیشرفت
        progress_frame = tk.Frame(main_frame, bg="#f0f2f5")
        progress_frame.pack(fill=tk.X, pady=5)

        self.progress_bar = ttk.Progressbar(progress_frame, orient=tk.HORIZONTAL, length=100, mode='determinate')
        self.progress_bar.pack(fill=tk.X, pady=2)

        self.progress_label = tk.Label(progress_frame, text="آماده استخراج لینک‌ها...",
                                     bg="#f0f2f5", font=default_font)
        self.progress_label.pack(anchor=tk.CENTER)

        # بخش دکمه‌های عملیات
        btn_frame = tk.Frame(main_frame, bg="#f0f2f5")
        btn_frame.pack(fill=tk.X, pady=10)

        self.stop_btn = tk.Button(btn_frame, text="توقف", command=self.stop_extraction_process,
                               bg="#ff4a4a", fg="white", font=default_font, padx=10, state=tk.DISABLED)
        self.stop_btn.pack(side=tk.LEFT)

        self.add_to_idm_btn = tk.Button(btn_frame, text="اضافه به IDM", command=self.add_to_idm,
                                     bg="#4aff7a", fg="black", font=default_font, padx=10, state=tk.DISABLED)
        self.add_to_idm_btn.pack(side=tk.RIGHT)

        # اعمال سبک RTL برای نمایش متون فارسی
        self.apply_rtl_style()

    def apply_rtl_style(self):
        # تغییر جهت المان‌ها برای پشتیبانی از RTL
        for widget in [self.url_entry, self.ext_entry, self.proxy_entry]:
            widget.configure(justify=tk.RIGHT)

    def toggle_proxy(self):
        if self.use_proxy_var.get():
            self.proxy_entry.configure(state=tk.NORMAL)
        else:
            self.proxy_entry.configure(state=tk.DISABLED)

    def log(self, message):
        self.log_area.configure(state=tk.NORMAL)
        self.log_area.insert(tk.END, message + "\n")
        self.log_area.see(tk.END)
        self.log_area.configure(state=tk.DISABLED)
        self.root.update_idletasks()

    def start_extraction(self):
        # تنظیم وضعیت اولیه
        self.stop_extraction.clear()
        self.extracted_links = []
        self.log_area.configure(state=tk.NORMAL)
        self.log_area.delete(1.0, tk.END)
        self.log_area.configure(state=tk.DISABLED)

        url = self.url_var.get().strip()

        if not url:
            messagebox.showerror("خطا", "لطفاً آدرس سایت را وارد کنید.")
            return

        if not (url.startswith('http://') or url.startswith('https://')):
            url = 'https://' + url
            self.url_var.set(url)

        # فعال/غیرفعال کردن دکمه‌ها
        self.extract_btn.configure(state=tk.DISABLED)
        self.stop_btn.configure(state=tk.NORMAL)
        self.add_to_idm_btn.configure(state=tk.DISABLED)

        self.progress_bar['value'] = 0
        self.progress_label.configure(text="در حال آماده‌سازی...")

        # شروع پردازش در ترد جداگانه
        thread = threading.Thread(target=self.extraction_process, args=(url,))
        thread.daemon = True
        thread.start()

    def extraction_process(self, url):
        self.log(f"[INFO] در حال استخراج لینک‌ها از {url}")

        try:
            # تنظیم پروکسی در صورت نیاز
            proxies = None
            if self.use_proxy_var.get() and self.proxy_var.get().strip():
                proxy = self.proxy_var.get().strip()
                proxies = {
                    'http': f'http://{proxy}',
                    'https': f'https://{proxy}'
                }
                self.log(f"[INFO] استفاده از پروکسی: {proxy}")

            # دریافت محتوای سایت
            headers = {
                'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36'
            }

            response = requests.get(url, headers=headers, proxies=proxies, timeout=30)
            response.raise_for_status()

            # پردازش محتوا با BeautifulSoup
            soup = BeautifulSoup(response.text, 'html.parser')

            # استخراج تمام لینک‌ها
            all_links = soup.find_all('a')
            total_links = len(all_links)

            # فیلتر کردن بر اساس پسوندهای مورد نظر
            extensions = [ext.strip().lower() for ext in self.extensions_var.get().split(',') if ext.strip()]

            self.log(f"[INFO] جستجوی لینک‌های با پسوند: {', '.join(extensions)}")

            base_url = url

            for i, link in enumerate(all_links):
                if self.stop_extraction.is_set():
                    self.log("[INFO] عملیات استخراج متوقف شد.")
                    break

                href = link.get('href')
                if not href:
                    continue

                # تبدیل به URL مطلق
                absolute_url = urljoin(base_url, href)

                # بررسی پسوند فایل
                if extensions:
                    parsed_url = urlparse(absolute_url)
                    path = parsed_url.path.lower()

                    if any(path.endswith(f'.{ext}') for ext in extensions):
                        self.log(f"[+] لینک یافت شد: {absolute_url}")
                        self.extracted_links.append(absolute_url)
                else:
                    # اگر پسوندی مشخص نشده باشد، همه لینک‌ها را اضافه می‌کنیم
                    self.log(f"[+] لینک یافت شد: {absolute_url}")
                    self.extracted_links.append(absolute_url)

                # به‌روزرسانی نوار پیشرفت
                progress = int((i + 1) / total_links * 100)
                self.progress_bar['value'] = progress
                self.progress_label.configure(text=f"پیشرفت: {progress}% - {len(self.extracted_links)} لینک یافت شده")
                self.root.update_idletasks()

            self.log(f"[INFO] استخراج لینک‌ها به پایان رسید. تعداد لینک‌های یافت شده: {len(self.extracted_links)}")

        except requests.exceptions.RequestException as e:
            self.log(f"[ERROR] خطا در دریافت محتوای سایت: {str(e)}")
        except Exception as e:
            self.log(f"[ERROR] خطای غیرمنتظره: {str(e)}")

        # بازگرداندن وضعیت دکمه‌ها
        self.extract_btn.configure(state=tk.NORMAL)
        self.stop_btn.configure(state=tk.DISABLED)

        if self.extracted_links:
            self.add_to_idm_btn.configure(state=tk.NORMAL)

        if not self.stop_extraction.is_set():
            self.progress_bar['value'] = 100
            self.progress_label.configure(text=f"استخراج تکمیل شد - {len(self.extracted_links)} لینک یافت شده")

    def stop_extraction_process(self):
        self.stop_extraction.set()
        self.log("[INFO] درخواست توقف عملیات استخراج...")

    def add_to_idm(self):
        """اضافه کند IDM سعی میکند لینک‌ها را به COM با استفاده از رابط"""
        if not self.extracted_links:
            messagebox.showinfo("اطلاعات", "هیچ لینکی برای اضافه کردن به IDM یافت نشد.")
            return

        try:
            # تلاش برای اتصال به IDM با روش‌های مختلف
            self.log("[INFO] در حال اتصال به IDM...")

            idm = None
            error_messages = []

            # روش 1: استفاده از COMObject استاندارد
            try:
                idm = win32com.client.Dispatch("IDMan.COMObject")
            except Exception as e:
                error_messages.append(f"روش 1 ناموفق: {str(e)}")

            # روش 2: امتحان کردن با COMObject.1
            if idm is None:
                try:
                    idm = win32com.client.Dispatch("IDMan.COMObject.1")
                except Exception as e:
                    error_messages.append(f"روش 2 ناموفق: {str(e)}")

            # روش 3: استفاده از dynamic dispatch
            if idm is None:
                try:
                    idm = win32com.client.dynamic.Dispatch("IDMan.COMObject")
                except Exception as e:
                    error_messages.append(f"روش 3 ناموفق: {str(e)}")

            # روش 4: امتحان با نام‌های دیگر
            if idm is None:
                possible_names = ["IDManLib.IDManLib", "IDManLib.IDManLib.1", "IDM.COMObject", "IDM.COMObject.1"]
                for name in possible_names:
                    try:
                        idm = win32com.client.Dispatch(name)
                        self.log(f"[INFO] اتصال به IDM با {name} موفقیت‌آمیز بود.")
                        break
                    except Exception as e:
                        error_messages.append(f"تلاش برای {name} ناموفق: {str(e)}")

            # اگر همه روش‌ها ناموفق بودند، به خط فرمان متوسل می‌شویم
            if idm is None:
                self.log("[INFO] امکان اتصال به IDM از طریق رابط COM وجود ندارد. استفاده از روش خط فرمان...")
                return self.add_to_idm_by_commandline()

            # اگر به اینجا رسیدیم، یعنی موفق به اتصال به IDM شده‌ایم
            self.progress_bar['value'] = 0
            total_links = len(self.extracted_links)

            for i, link in enumerate(self.extracted_links):
                try:
                    # تغییر پارامتر آخر به 1 برای اضافه کردن مستقیم به صف دانلود - افزودن لینک به IDM
                    # پارامترهای AddURL:
                    # url, referer, cookie, user, pass, comment, group, flags
                    # flags=1 یعنی اضافه کردن مستقیم به صف بدون نمایش پنجره
                    idm.AddURL(link, "", "", "", "", "", "", "1")
                    self.log(f"[+] لینک به صف دانلود IDM اضافه شد: {link}")

                    # به‌روزرسانی نوار پیشرفت
                    progress = int((i + 1) / total_links * 100)
                    self.progress_bar['value'] = progress
                    self.progress_label.configure(text=f"افزودن به IDM: {progress}% - {i+1}/{total_links}")
                    self.root.update_idletasks()

                    # کمی تأخیر
                    time.sleep(0.1)

                except Exception as e:
                    self.log(f"[ERROR] خطا در افزودن لینک {link} به IDM: {str(e)}")

            self.log(f"[INFO] افزودن لینک‌ها به IDM به پایان رسید.")
            messagebox.showinfo("تکمیل", f"{total_links} لینک با موفقیت به صف دانلود IDM اضافه شد.")

        except Exception as e:
            error_msg = str(e)
            self.log(f"[ERROR] خطا در اتصال به IDM: {error_msg}")
            detailed_errors = "\n".join(error_messages)
            self.log(f"جزئیات خطاها:\n{detailed_errors}")

            # در صورت خطا، از روش خط فرمان استفاده می‌کنیم
            return self.add_to_idm_by_commandline()

        finally:
            self.progress_bar['value'] = 100
            self.progress_label.configure(text="عملیات تکمیل شد")

    def add_to_idm_by_commandline(self):
        """بدون نمایش پنجره دانلود IDM استفاده از خط فرمان برای اضافه کردن لینک‌ها به"""
        try:
            # پیدا کردن مسیر نصب IDM
            idm_paths = [
                r"C:\Program Files (x86)\Internet Download Manager\IDMan.exe",
                r"C:\Program Files\Internet Download Manager\IDMan.exe"
            ]

            idm_path = None
            for path in idm_paths:
                if os.path.exists(path):
                    idm_path = path
                    break

            if not idm_path:
                raise Exception("مسیر اجرایی IDM یافت نشد. لطفاً مطمئن شوید IDM نصب شده است.")

            self.log(f"[INFO] IDM در مسیر {idm_path} یافت شد. استفاده از روش خط فرمان.")

            self.progress_bar['value'] = 0
            total_links = len(self.extracted_links)

            for i, link in enumerate(self.extracted_links):
                try:
                    # استفاده از خط فرمان برای اضافه کردن لینک به IDM
                    # برای اضافه کردن مستقیم به صف دانلود بدون نمایش پنجره /a استفاده از پارامتر
                    import subprocess
                    cmd = f'"{idm_path}" /a /d "{link}"'
                    subprocess.Popen(cmd, shell=True)

                    self.log(f"[+] لینک به صف دانلود IDM اضافه شد (روش خط فرمان): {link}")

                    # به‌روزرسانی نوار پیشرفت
                    progress = int((i + 1) / total_links * 100)
                    self.progress_bar['value'] = progress
                    self.progress_label.configure(text=f"افزودن به IDM: {progress}% - {i+1}/{total_links}")
                    self.root.update_idletasks()

                    # کمی تأخیر برای جلوگیری از فشار بیش از حد
                    time.sleep(0.5)

                except Exception as e:
                    self.log(f"[ERROR] خطا در افزودن لینک {link} به IDM: {str(e)}")

            self.log(f"[INFO] افزودن لینک‌ها به IDM به پایان رسید.")
            messagebox.showinfo("تکمیل", f"{total_links} لینک با موفقیت به صف دانلود IDM اضافه شد.")
            return True

        except Exception as e:
            error_msg = str(e)
            self.log(f"[ERROR] خطا در استفاده از روش خط فرمان: {error_msg}")
            messagebox.showerror("خطا", f"خطا در افزودن لینک‌ها به IDM با روش خط فرمان: {error_msg}")
            return False


if __name__ == "__main__":
    root = tk.Tk()
    app = IDMDownloaderApp(root)
    root.mainloop()
```

راهنمای نصب و استفاده

مراحل نصب

1. نصب پایتون: ابتدا پایتون را نصب کنید (نسخه 3.6 یا بالاتر).
2. نصب کتابخانه‌های مورد نیاز: دستور زیر را در ترمینال یا خط فرمان اجرا کنید:

```
pip install requests beautifulsoup4 pywin32
```

3. ذخیره کد برنامه: کد بالا را در یک فایل با پسوند `.py` ذخیره کنید (مثلاً `idm_downloader.py`).
4. نصب IDM: اطمینان حاصل کنید که نرم‌افزار Internet Download Manager نصب شده باشد.

نحوه استفاده

1. اجرای برنامه: فایل `idm_downloader.py` را اجرا کنید.

```
python idm_downloader.py
```

2. وارد کردن آدرس سایت: آدرس سایت مورد نظر را در فیلد مربوطه وارد کنید.
3. تنظیم پسوندهای مورد نظر: پسوندهای فایل‌هایی که می‌خواهید دانلود کنید را مشخص کنید (مثلاً `zip,rar,pdf,mp3,mp4,exe`).
4. استفاده از پروکسی (اختیاری): در صورت نیاز، تیک گزینه "استفاده از پروکسی" را فعال کرده و آدرس و پورت پروکسی را وارد کنید.
5. شروع استخراج: دکمه "استخراج" را کلیک کنید تا فرآیند استخراج لینک‌ها آغاز شود.
6. اضافه کردن به IDM: پس از اتمام استخراج، دکمه "اضافه به IDM" را کلیک کنید تا لینک‌ها به صف دانلود IDM اضافه شوند.

رفع مشکلات احتمالی

خطای "No module named 'requests'"

این خطا نشان می‌دهد که کتابخانه requests نصب نشده است. با اجرای دستور زیر آن را نصب کنید:

```
pip install requests
```

خطای "Object creation failed"

این خطا معمولاً زمانی رخ می‌دهد که برنامه نتواند به IDM متصل شود. اطمینان حاصل کنید که:

- نرم‌افزار IDM نصب شده است
- IDM در حال اجراست
- کتابخانه pywin32 به درستی نصب شده است

نمایش نادرست متون فارسی

اگر متون فارسی به درستی نمایش داده نمی‌شوند، می‌توانید:

- از فونت‌های پشتیبانی‌کننده زبان فارسی استفاده کنید (مانند Tahoma)
- کدگذاری ترمینال خود را به UTF-8 تغییر دهید
- در ابتدای کد پایتون خود، خط زیر را اضافه کنید:

```
# -*- coding: utf-8 -*-
```

مشکل در استخراج لینک‌ها

برخی سایت‌ها ممکن است لینک‌های خود را به صورت پویا بارگذاری کنند یا از JavaScript استفاده کنند که با BeautifulSoup قابل استخراج نیستند. در این موارد می‌توانید:

- از کتابخانه‌های پیشرفته‌تر مانند Selenium استفاده کنید
- به API سایت (در صورت وجود) متصل شوید

این برنامه به منظور استفاده‌های آموزشی ایجاد شده است. لطفاً از آن برای اهداف قانونی استفاده کنید.