YouTube Video Downloader Manual
(Highest Available Quality - Single Video & Playlist)
==

1. Introduction

This manual explains how to download YouTube videos in the highest available resolution using Python on Windows. It covers:

- Downloading a single video (4K / highest available).
- Downloading an entire playlist.
- Using adaptive (video+audio separate) and fallback progressive downloads.
- Handling common issues.

You will use:
- Python
- The 'pytubefix' library
- FFmpeg (for merging video + audio when needed)

2. Prerequisites

Before running the scripts, make sure you have:

1) Python installed (3.x)
   - You can check with:
```
python --version
```

2) pip (Python package manager)
   - Usually comes with Python.
   - You can check with:
```
python -m pip --version
```

3) Internet connection (to download YouTube videos).

3. Install Required Python Package

We will use 'pytubefix', which is a maintained fork of pytube.

Open Command Prompt or PowerShell and run:

```
python -m pip install pytubefix
```

4. Install and Configure FFmpeg (for highest quality)

FFmpeg is used to merge video and audio streams when YouTube provides them separately (especially 1080p, 1440p, 4K resolutions).

High-level steps (Windows):
1) Download a pre-built FFmpeg zip for Windows (essentials build).
2) Extract it to a folder (e.g. C:\ffmpeg).
3) Inside that folder there will be a 'bin' directory containing 'ffmpeg.exe'.
4) Add that 'bin' path to your System PATH environment variable, for example:
   C:\ffmpeg\bin
5) Open a new Command Prompt and test:

```
ffmpeg -version
```

If you see version information, FFmpeg is correctly installed.

5. Single Video - Highest Available Quality (Safe Version)

Create a new file, e.g.:
    download_single_highest_safe.py

Paste the following code into it and save:

```
------------------------------------------------------------
from pytubefix import YouTube
import os
import re
import subprocess

# ========= CONFIG ============

# Put your YouTube video link here
VIDEO_URL = "https://www.youtube.com/watch?v=XXXXXXXXXXX"

# Folder where the video will be saved
OUTPUT_DIR = r"D:\Python Dsa\Single_Highest"

# ============================


def safe_filename(title: str) -> str:
    """Create a safe, short file name for Windows."""
    title = re.sub(r'[\\/*?:"<>|]', "", title)
    title = title.replace("\n", " ").replace("\r", " ")
    return (title.strip() or "video")[:80]


def run_ffmpeg_merge(video_path: str, audio_path: str, out_path: str):
    """Merge video + audio using ffmpeg (no re-encode)."""
    cmd = [
        "ffmpeg",
        "-y",
        "-i", video_path,
        "-i", audio_path,
        "-c", "copy",
        out_path,
    ]
    subprocess.run(cmd, check=True)


def download_with_retry(stream, output_path, filename, max_tries=3):
    """Download a stream with multiple retry attempts."""
    for attempt in range(1, max_tries + 1):
        try:
            print(f"   Downloading ({attempt}/{max_tries}) -> {filename}")
            return stream.download(output_path=output_path, filename=filename)
        except Exception as e:
            print("   Download error:", e)
            if attempt == max_tries:
                raise


def main():
    os.makedirs(OUTPUT_DIR, exist_ok=True)

    yt = YouTube(VIDEO_URL)
```

```
print("Title:", yt.title)

base = safe_filename(yt.title)
final_file = os.path.join(OUTPUT_DIR, base + ".mp4")

if os.path.exists(final_file):
    print("Already exists:", final_file)
    return

# 1) Try highest adaptive (best quality)
try:
    print("\nTrying highest adaptive (best quality)...")

    v_stream = (
        yt.streams
        .filter(adaptive=True, only_video=True, file_extension="mp4")
        .order_by("resolution")
        .desc()
        .first()
    )

    a_stream = (
        yt.streams
        .filter(adaptive=True, only_audio=True, file_extension="mp4")
        .order_by("abr")
        .desc()
        .first()
    )

    if v_stream and a_stream:
        print("Selected video:", v_stream.resolution, "audio:", a_stream.abr)

        temp_video_name = base + "_video.mp4"
        temp_audio_name = base + "_audio.mp4"
        temp_video = os.path.join(OUTPUT_DIR, temp_video_name)
        temp_audio = os.path.join(OUTPUT_DIR, temp_audio_name)

        download_with_retry(v_stream, OUTPUT_DIR, temp_video_name)
        download_with_retry(a_stream, OUTPUT_DIR, temp_audio_name)

        print("Merging (adaptive) with ffmpeg...")
        run_ffmpeg_merge(temp_video, temp_audio, final_file)

        os.remove(temp_video)
        os.remove(temp_audio)

        print("Done (adaptive):", final_file)
        return
    else:
        print("Adaptive streams not found, going to fallback...")

except Exception as e:
    print("Adaptive method failed:", e)
    print("Falling back to progressive (up to 720p)...")

# 2) Fallback: progressive (video+audio combined)
try:
    p_stream = (
        yt.streams
        .filter(progressive=True, file_extension="mp4")
        .order_by("resolution")
        .desc()
```

```
            .first()
        )

        if p_stream is None:
            print("No progressive stream found. Cannot download this video.")
            return

        print("Fallback progressive stream:", p_stream.resolution)
        download_with_retry(p_stream, OUTPUT_DIR, os.path.basename(final_file))

        print("Done (progressive fallback):", final_file)

    except Exception as e:
        print("Complete fail for this video:", e)


if __name__ == "__main__":
    main()
```

How this works:
- First, it tries to download the highest separate video + audio streams (adaptive).
- Then it merges them with FFmpeg for highest quality (up to 4K, if available).
- If that fails (for any reason), it falls back to the best progressive stream (video+audio together, usually up to 720p).

6. Playlist - Highest Available Quality (Safe Version)

Now create another file, e.g.:
    download_playlist_highest_safe.py

Paste this code and save:

```
from pytubefix import Playlist
import os
import re
import subprocess

PLAYLIST_URL = "https://www.youtube.com/playlist?list=PLgUwDviBIf0rGlzIn_7rsaR2FQ5e6ZOL9"
OUTPUT_DIR = r"D:\Python Dsa\Highest_Safe_Videos"


def safe_filename(title: str) -> str:
    title = re.sub(r'[\\/*?:"<>|]', "", title)
    title = title.replace("\n", " ").replace("\r", " ")
    return (title.strip() or "video")[:80]


def run_ffmpeg_merge(video_path: str, audio_path: str, out_path: str):
    cmd = [
        "ffmpeg",
        "-y",
        "-i", video_path,
        "-i", audio_path,
        "-c", "copy",
        out_path,
    ]
    subprocess.run(cmd, check=True)
```

```
def download_with_retry(stream, output_path, filename, max_tries=3):
    for attempt in range(1, max_tries + 1):
        try:
            print(f"   Downloading ({attempt}/{max_tries}) -> {filename}")
            return stream.download(output_path=output_path, filename=filename)
        except Exception as e:
            print("   Download error:", e)
            if attempt == max_tries:
                raise


def main():
    os.makedirs(OUTPUT_DIR, exist_ok=True)

    playlist = Playlist(PLAYLIST_URL)
    print("Playlist:", playlist.title)
    print("Total videos:", len(playlist.videos))

    for idx, video in enumerate(playlist.videos, start=1):
        print("\n========================================")
        print(f"[{idx}/{len(playlist.videos)}] {video.title}")

        base = safe_filename(video.title)
        final_file = os.path.join(OUTPUT_DIR, base + ".mp4")

        if os.path.exists(final_file):
            print("Already downloaded, skipping:", final_file)
            continue

        # 1) Try highest adaptive
        try:
            print("Trying highest adaptive (best quality)...")
            v_stream = (
                video.streams
                .filter(adaptive=True, only_video=True, file_extension="mp4")
                .order_by("resolution")
                .desc()
                .first()
            )
            a_stream = (
                video.streams
                .filter(adaptive=True, only_audio=True, file_extension="mp4")
                .order_by("abr")
                .desc()
                .first()
            )

            if v_stream and a_stream:
                print("Selected video:", v_stream.resolution, "audio:", a_stream.abr)

                temp_video_name = base + "_video.mp4"
                temp_audio_name = base + "_audio.mp4"
                temp_video = os.path.join(OUTPUT_DIR, temp_video_name)
                temp_audio = os.path.join(OUTPUT_DIR, temp_audio_name)

                download_with_retry(v_stream, OUTPUT_DIR, temp_video_name)
                download_with_retry(a_stream, OUTPUT_DIR, temp_audio_name)

                print("Merging (adaptive) with ffmpeg...")
                run_ffmpeg_merge(temp_video, temp_audio, final_file)
                os.remove(temp_video)
                os.remove(temp_audio)

                print("Done (adaptive):", final_file)
                continue
            else:
                print("Adaptive streams not found, using fallback...")

        except Exception as e:
            print("Adaptive method failed:", e)
            print("Falling back to progressive (up to 720p)...")

        # 2) Fallback: progressive
        try:
            p_stream = (
                video.streams
                .filter(progressive=True, file_extension="mp4")
                .order_by("resolution")
                .desc()
                .first()
            )

            if p_stream is None:
                print("No progressive stream found for this video. Skipping.")
                continue

            print("Fallback progressive stream:", p_stream.resolution)
            download_with_retry(p_stream, OUTPUT_DIR, os.path.basename(final_file))

            print("Done (progressive fallback):", final_file)

        except Exception as e:
            print("Complete fail for this video:", e)


if __name__ == "__main__":
    main()
```
----

7. How to Run the Scripts

Single video script:
1) Edit VIDEO_URL and OUTPUT_DIR in download_single_highest_safe.py
2) In the terminal, run:

```
    python -u "D:\Python Dsa\download_single_highest_safe.py"
```

Playlist script:
1) Edit PLAYLIST_URL and OUTPUT_DIR in download_playlist_highest_safe.py
2) Run:

```
    python -u "D:\Python Dsa\download_playlist_highest_safe.py"
```

8. How to Stop the Program While Running

If you want to stop the download in the middle:

- In Command Prompt or PowerShell, press:

```
    Ctrl + C
```

This sends an interrupt signal to Python and stops the script.

```
9. Common Problems & Tips
-------------------------
1) Some videos fail to download
   - Video might be private, deleted, region-locked, or age-restricted.
   - In such cases, streams may not be accessible.

2) FFmpeg not found
   - If you see an error that 'ffmpeg' is not recognized:
     - Check that the 'bin' folder containing ffmpeg.exe is added to the PATH.
     - Open a new terminal and run:
       ffmpeg -version

3) Low resolution video
   - For highest quality:
     - The script first tries adaptive streams (up to 4K or whatever is available).
     - If that fails, it falls back to progressive (max ~720p).
   - If a particular video does not have higher resolution on YouTube,
     it cannot be downloaded in higher quality than available.

4) Long file names / invalid characters
   - Windows does not like some characters in file names.
   - The safe_filename() function removes problematic characters and shortens titles.


10. Summary
-----------
- Use 'pytubefix' to interact with YouTube videos and playlists.
- Use FFmpeg to merge separate video and audio streams for maximum quality.
- Single video script:
  - download_single_highest_safe.py
- Playlist script:
  - download_playlist_highest_safe.py
- Both scripts first attempt highest quality (adaptive) and then safely fall back
  to progressive downloads to minimize failures.
```