Quick presentation

This section explains when MoviePy can be used and how it works.

Do I need MoviePy ?

Here are a few reasons why you may want to edit videos in Python:

- You have many videos to process or to compose in a complicated way.
- You want to automatize the creation of videos or GIFs on a web server (Django, Flask, etc.)
- You want to automatize tedious tasks, like title insertions tracking objects, cuting scenes, making end credits, subtitles, etc...
- You want to code your own video effects to do something no existing video editor can.
- You want to create animations from images generated by another python library (Matplotlib, Mayavi, Gizeh, scikit-images...)

And here are a few uses for which MoviePy is NOT the best solution:

- You only need to do frame-by-frame video analysis (with face detection or other fancy stuff). This could be done with MoviePy in association with other libraries, but really, just use imageio, OpenCV or SimpleCV, these are libraries that specialize in these tasks.
- You only want to convert a video file, or turn a series of image files into a movie. In this case it is better to directly call `ffmpeg` (or `avconv` or `mencoder` ...) it will be faster more memory-efficient than going through MoviePy.

Advantages and limitations

MoviePy has been developed with the following goals in mind:

- **Simple an intuitive**. Basic operations can be done in one line. The code is easy to learn and easy to understand for newcomers.
- **Flexible**. You have total control over the frames of the video and audio, and creating your own effects is easy as Py.
- **Portable**. The code uses very common software (Numpy and FFMPEG) and can run on (almost) any machine with (almost) any version of Python.

For the limitations: MoviePy cannot (yet) stream videos (read from a webcam, or render a video live on a distant machine), and is not really designed for video processing involving many successive frames of a movie (like video stabilization, you'll need another software for that). You can also have

memory problems if you use many video, audio, and image sources at the same time (>100), but this will be fixed in future versions.

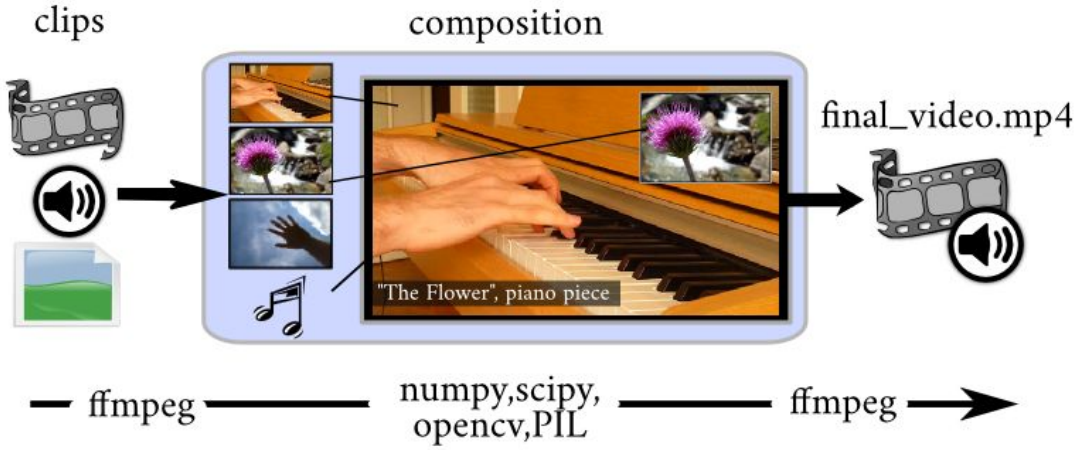
Example code

In a typical MoviePy script, you load video or audio files, modify them, put them together, and write the final result to a new video file. As an example, let us load a video of my last holidays, lower the volume, add a title in the center of the video for the first ten seconds, and write the result in a file:

```
# Import everything needed to edit video clips
from moviepy.editor import *

# Load myHolidays.mp4 and select the subclip 00:00:50 - 00:00:60
clip = VideoFileClip("myHolidays.mp4").subclip(50,60)

# Reduce the audio volume (volume x 0.8)
clip = clip.volumex(0.8)

# Generate a text clip. You can customize the font, color, etc.
txt_clip = TextClip("My Holidays 2013",fontsize=70,color='white')

# Say that you want it to appear 10s at the center of the screen
txt_clip = txt_clip.set_pos('center').set_duration(10)

# Overlay the text clip on the first video clip
video = CompositeVideoClip([clip, txt_clip])

# Write the result to a file (many options available !)
video.write_videofile("myHolidays_edited.webm")
```

How MoviePy works

MoviePy uses the software `ffmpeg` to read and to export video and audio files. It also (optionally) uses ImageMagick to generate texts and write GIF files. The processing of the different media is ensured by Python's fast numerical library Numpy. Advanced effects and enhancements use some of Python's numerous image processing libraries (PIL, Scikit-image, scipy, etc.).



Basic concepts

The central objects of MoviePy are *clips*, which can be `AudioClips` or `VideoClips`. They can be modified (cut, slowed down, darkened...) or put mixed with clips to form new clips, they can be previewed (using either PyGame or the IPython Notebook) and rendered to a file (as a MP4, a GIF, a MP3, etc.). `VideoClips` for instance can be created from a video file, an image, a text, or a custom animation. They can have an audio track (which is an `AudioClip`) and a mask (a special `VideoClip` indicating which parts of the clip to hide when the clip is mixed with other clips). See Creating and exporting video clips and Mixing clips for more details.

A clip can be modified using one of moviepy's numerous effects (like in `clip.resize(width="360")`, `clip.subclip(t1,t2)`, or `clip.fx(vfx.black_white)`) or using a user-implemented effect. MoviePy implements many functions (like `clip.fl`, `clip.fx`, etc.) which make it very easy to code your own effect in a few lines. See Clips transformations and effects for more.

You will also find a few advanced goodies in `moviepy.video.tools` to track objects in a video, draw simple shapes and color gradients (very useful for masks), generate subtitles and end credits, etc. See Advanced tools for a description of these.

Finally, although MoviePy has no graphical user interface, there are many ways to preview a clip which allow you to fine-tune your scripts and be sure that everything is perfect when you render you video in high quality. See How to be efficient with MoviePy.