Editing Video with Python + MoviePy

Prerequisites

- **Basics of Python 3**
- **Basic knowledge about video files**
- **Python 3.7.4**
- **MoviePy 1.0.3**
- **PyCharm 2019.3.5**
- **Windows 10 or above**

MoviePy

Python module for editing videos. It can cut and arrange clips, add video effects, and edit audio. It can work like a toolbox, if you only make changes to one clip. If you have to edit lots of clips, MoviePy can automate that process.

Installing MoviePy

- **First, set up a folder with a new Python script moviepy_test.py.**
- **Then install the package moviepy.**
- **In your terminal, you can do this with the command pip install moviepy**

Downloading the raw video

- **Download a mp4 file and name as sample-mp4-file.mp4 format**
- **Move sample-mp4-file.mp4 to the folder with moviepy_test.py.**

Basic editing

```
from moviepy.editor import *

clip = VideoFileClip("sample-mp4-file.mp4")

clip = VideoFileClip("sample-mp4-file.mp4").subclip(56, 66)
```

Basic editing

Put these clips together in a list, and chain them together with concatenate_videoclip

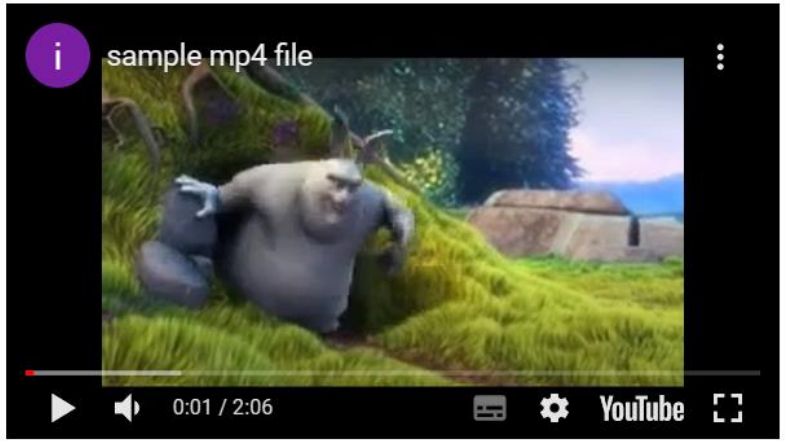
```
final_clip = concatenate_videoclips([clip, clip2, clip3, clip4])
```

Finally, to get our finished video, we use the write_videofile method on final_clip

```
from moviepy.editor import *
# clip is the video from 00:56 to 01:06
clip = VideoFileClip("sample-mp4-file.mp4").subclip(56, 66)
clip2 = VideoFileClip("sample-mp4-file.mp4").subclip(70, 76)
clip3 = VideoFileClip("sample-mp4-file.mp4").subclip(50, 52)
clip4 = VideoFileClip("sample-mp4-file.mp4").subclip(30, 35)
final_clip = concatenate_videoclips([clip, clip2, clip3, clip4])

final_clip.write_videofile("output_1.mp4")
```

Output :



to ->

Reduced original clip size

Adding effects

MoviePy can do more than just cut and rearrange video. The module moviepy.editor also contains submodules. vfx holds video effects, and afx holds audio effects

```
clip = (VideoFileClip("sample-mp4-file.mp4").subclip(56, 66)
    .fx(vfx.colorx, 0.7))

# clip_edited = clip.fx(vfx.colorx, 0.7)

clip = (VideoFileClip("sample-mp4-file.mp4").subclip(56, 66)
        .fx(vfx.colorx, 1.2)  # 20% brighter
        .fx(vfx.lum_contrast, 0, 40, 127))  # and increase the contrast
```

Adding effects

with clip 2 and invert_colors, and output our results

```
from moviepy.editor import *

clip = (VideoFileClip("sample-mp4-file.mp4").subclip(56, 66)
        .fx(vfx.colorx, 1.2)  # 20% brighter
        .fx(vfx.lum_contrast, 0, 40, 127))  # and increase the contrast
clip2 = (VideoFileClip("sample-mp4-file.mp4").subclip(70, 76)
        .fx(vfx.invert_colors))
clip3 = VideoFileClip("sample-mp4-file.mp4").subclip(50, 52)
clip4 = VideoFileClip("sample-mp4-file.mp4").subclip(30, 35)

final_clip = concatenate_videoclips([clip, clip2, clip3, clip4])

final_clip.write_videofile("output_2.mp4")
```

Output :

to ->

Reduced original clip size

Audio effects

can import the clip and excerpt the first 10 seconds:

```
musicclip = AudioFileClip("Study and Relax.mp3").subclip(0, 6)

audioclip = clip.audio
```

Audio effects are in the afx submodule:

```
audioclip = (clip.audio).afx(afx.volumex, 1.2).afx(afx.audio_fadein, 1.0)
# Make the sound 20% louder, and fade it in over 1 second
```

Audio effects

The set_audio function replaces a video clip's audio with a new audio clip.

```
musicclip = AudioFileClip("Study and Relax.mp3").subclip(0, 6)
audioclip = (clip.audio).fx(afx.volumex, 1.2).fx(afx.audio_fadein, 1.0)
# Make the sound 20% louder, and fade it in over 1 second
clip_v2 = clip.set_audio(audioclip)  # new first clip

final_clip = concatenate_videoclips([clip_v2, clip2, clip3, clip4])

final_clip.write_videofile("output_3.mp4")
```

Putting it all together

```
composite_start_of_video = CompositeVideoClip([clip_v2,
                                               clip4.fx(vfx.resize, 0.6)])
# clip4 is smaller (60% original size), and on top of clip_v2

composite_start_of_video = CompositeVideoClip([clip_v2,
                                               clip4.fx(vfx.resize, 0.6).fx(afx.volumex, 0.0)])
# clip4 is smaller (60% original size), and on top of clip_v2

musicclip = AudioFileClip("Study and Relax.mp3").subclip(0, 6)
audioclip = (clip.audio).fx(afx.volumex, 1.2).fx(afx.audio_fadein, 1.0)
# Make the sound 20% louder, and fade it in over 1 second
clip_v2 = clip.set_audio(audioclip)  # new first clip

final_clip.write_videofile("output_4.mp4")
```

Putting it all together

```
composite_start_of_video = CompositeVideoClip([clip_v2,
                                                clip4.fx(vfx.resize, 0.6).fx(afx.volumex, 0.0)])
# clip4 is smaller (60% original size), and on top of clip_v2

clip2_audio = (clip2.audio).fx(afx.volumex, 1.5)  # 50% louder, so we can hear over our music
composite_second_clip_audio = CompositeAudioClip([clip2_audio,
                                                   (musicclip).fx(afx.volumex, 0.3)])  # 70% quieter
clip2_v2 = clip2.set_audio(composite_second_clip_audio)

final_clip = concatenate_videoclips([composite_start_of_video, clip2_v2, clip3, clip4])
```

Conclusion

Edited and remixed a video, all in a Python program