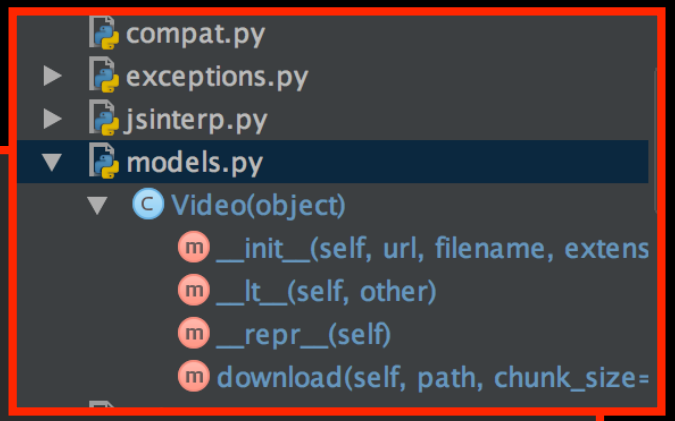
models.py

- pytube실행을 위한 동영상 다운로드 모듈
- 사용자가 지정한 youtube 영상의 데이터를 api.py 에서 실행 하고 다운로드 할 수 있게 하는 역할을 한다
- try - except: url을 여는 확장 라이브러리 모듈을 import (urlopen은 기본 내장된 라이브러리)
- Video 클래스 내에 __init__, download, __repr__, __it__ 총 네 개의 메소드가 생성되어 있다




```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from __future__ import unicode_literals
import os
from time import clock

try:
    from urllib2 import urlopen
except ImportError:
    from urllib.request import urlopen


class Video(object):
    """Class representation of a single instance of a YouTube video.
    """
    def __init__(self, url, filename, extension, resolution=None,
                 video_codec=None, profile=None, video_bitrate=None,
                 audio_codec=None, audio_bitrate=None):...

    def download(self, path, chunk_size=8 * 1024, on_progress=None,
                 on_finish=None, force_overwrite=False):...

    def __repr__(self):...

    def __lt__(self, other):...
```

```python
class Video(object):
    """Class representation of a single instance of a YouTube video.
    """
    def __init__(self, url, filename, extension, resolution=None,
                 video_codec=None, profile=None, video_bitrate=None,
                 audio_codec=None, audio_bitrate=None):
        """Sets-up the video object...."""
        self.url = url
        self.filename = filename
        self.extension = extension
        self.resolution = resolution
        self.video_codec = video_codec
        self.profile = profile
        self.video_bitrate = video_bitrate
        self.audio_codec = audio_codec
        self.audio_bitrate = audio_bitrate
```

```python
class Video(object):
    """Class representation of a single instance of a YouTube video.
    """
    def __init__(self, url, filename, extension, resolution=None,
                 video_codec=None, profile=None, video_bitrate=None,
                 audio_codec=None, audio_bitrate=None):
        """Sets-up the video object.

        :param str url:
            The url of the video. (e.g.: https://youtube.com/watch?v=..
        :param str filename:
            The filename (minus the extention) to save the video.
        :param str extention:
            The desired file extention (e.g.: mp4, flv, webm).
        :param str resolution:
            *Optional* The broadcasting standard (e.g.: 720p, 1080p).
        :param str video_codec:
            *Optional* The codec used to encode the video.
        :param str profile:
            *Optional* The arbitrary quality profile.
        :param str video_bitrate:
            *Optional* The bitrate of the video over sampling interval.
        :param str audio_codec:
            *Optional* The codec used to encode the audio.
        :param str audio_bitrate:
            *Optional* The bitrate of the video's audio over sampling i
        """
        self.url = url
        self.filename = filename
        self.extension = extension
        self.resolution = resolution
        self.video_codec = video_codec
        self.profile = profile
        self.video_bitrate = video_bitrate
        self.audio_codec = audio_codec
        self.audio_bitrate = audio_bitrate
```

def _init_()

- 다운로드 시 비디오에 저장되는 데이터 값을 호출하는 메소드.
- 해상도, 코덱 등 설정한 매개변수 값에 따라 하나의 youtube url에 들어있는 비디오 데이터의 내용들을 담고있다.

- url, 파일명, 확장자 외에 codec, profile, bitrate, audio,codec, audio bitrate 각각의 매개변수를 None 으로 초기화 하도록 설정

```
def download(self, path, chunk_size=8 * 1024, on_progress=None,
             on_finish=None, force_overwrite=False):
    """Downloads the video...."""
    path = os.path.normpath(path)
    if os.path.isdir(path):

        filename = "{0}.{1}".format(self.filename, self.extension)
        path = os.path.join(path, filename)
    # TODO: If it's not a path, this should raise an ``OSError``.
    # TODO: Move this into cli, this kind of logic probably shouldn't be
    # handled by the library.
    if os.path.isfile(path) and not force_overwrite:
        raise OSError("Conflicting filename:'{0}'".format(self.filename))
        # TODO: Split up the downloading and OS jazz into separate functions.
    response = urlopen(self.url)
    meta_data = dict(response.info().items())
    file_size = int(meta_data.get("Content-Length") or
                    meta_data.get("content-length"))
    self._bytes_received = 0
    start = clock()
    # TODO: Let's get rid of this whole try/except block, let ``OSErrors``
    # fail loudly.
```

1.
2.
3.
4.
5.

def download():

1. 특정 폴더가 존재할 경우 해당 폴더에 파일을 저장하고 폴더가 없으면 생성 후 그곳에 파일을 저장시킨다.
(경로를 검색하여 파일 존재 유무 확인 후 디렉토리에 저장)

2. os.path.isfile은 True를 리턴하는 메소드로 파일이 경로에 존재할 경우 에러를 발생시킨다

3. urlopen 메소드로 사용자가 지정한 YouTube 영상에 접근한 뒤 결과값을 저장한다.

4. meta_data에는 결과값으로 함께 받아온 정보들이 저장된다.

5.file_size 에는 Content-Length or content-length 두가지로 검색된 비디오 크기에 대한 결과 값이 저장된다.

```
        try:
            with open(path, 'wb') as dst_file:
                while True:
                    self._buffer = response.read(chunk_size)
                    # Check if the buffer is empty (aka no bytes remaining).
                    if not self._buffer:
                        if on_finish:
                            # TODO: We possibly want to flush the
                            # `_bytes_recieved`` buffer before we call
                            # ``on_finish()``.
                            on_finish(path)
                        break

                    self._bytes_received += len(self._buffer)
                    dst_file.write(self._buffer)
                    if on_progress:
                        on_progress(self._bytes_received, file_size, start)
```

try :
파일 입력을 시작
self.buffer 값으로 들어온 파일을 디렉토리 경로로 전달하기 시작한다.

```
        except KeyboardInterrupt:
            # TODO: Move this into the cli, ``KeyboardInterrupt`` handling
            # should be taken care of by the client. Also you should be allowed
            # to disable this.
            os.remove(path)
            raise KeyboardInterrupt(
                "Interrupt signal given. Deleting incomplete video.")
```

except :
만일 파일 전송 시 에러가 날 경우 KeyboardInterrupt로 예외처리가 되어
지정한 경로로 다운로드 되고 있는 파일을 제거한다.

```
    def __repr__(self):
        """A clean representation of the class instance."""
        return "<Video: {0} (.{1}) - {2} - {3}>".format(
            self.video_codec, self.extension, self.resolution, self.profile)

    def __lt__(self, other):
        """The "less than" (lt) method is used for comparing video object to
        one another. This useful when sorting.

        :param other:
            The instance of the other video instance for comparison.
        """
        if isinstance(other, Video):
            v1 = "{0} {1}".format(self.extension, self.resolution)
            v2 = "{0} {1}".format(other.extension, other.resolution)
            return (v1 > v2) - (v1 < v2) < 0
```

- def __repr__(): 기본적으로 __repr__은 문자열을 리턴하는 메소드이다. 여기에서도 str으로 선언된 비디오 코덱, 확장자, 해상도, 프로파일을 리턴하는 역할을 한다.

- def__lt__(): 다운로드 받은 Video 파일의 종류에 대해 유효성을 확인하는 메소드. isinstance 메소드에서 확장자, 해상도 값이 더 높은 데이터를 반환하여 현재 내가 사용하는 데이터타입이 무엇인지 비교하고 클래스의 실체인지 또는 클래스의 하위 클래스인지 점검한다.