TUBTUB TEAM

PYTUBE 분석

분배

- 박승권: api.py
- 서은정: model.py
- 유동현: util.py
- 박선배: jsinterp.py

TUBTUB TEAM

UTIL.PY -동현

```python
import argparse
import re

from os import path
from sys import stdout
from time import clock


class FullPaths(argparse.Action):
    """Expand user- and relative-paths"""
    def __call__(self, parser, namespace, values, option_string=None):
        setattr(namespace, self.dest, path.abspath(path.expanduser(values)))
```

```python
def truncate(text, max_length=200):
    return text[:max_length].rsplit(' ', 0)[0]
```

text를 자르는 메소드

```python
def safe_filename(text, max_length=200):
    """Sanitizes filenames for many operating systems.
    :params text: The unsanitized pending filename.
    """

    # Tidy up ugly formatted filenames.
    text = text.replace('_', ' ')
    text = text.replace(':', ' -')

    # NTFS forbids filenames containing characters in range 0-31 (0x00-0x1F)
    ntfs = [chr(i) for i in range(0, 31)]

    # Removing these SHOULD make most filename safe for a wide range of
    # operating systems.
    paranoid = ['\"', '\#', '\$', '\%', '\'', '\*', '\,', '\.', '\/', '\:',
                '\;', '\<', '\>', '\?', '\\', '\^', '\|', '\~', '\\\\']

    blacklist = re.compile('|'.join(ntfs + paranoid), re.UNICODE)
    filename = blacklist.sub('', text)
    return truncate(filename)
```

쓸때없는 문자를 정돈함

이런특수문자들 제거

```python
def sizeof(bytes):
    """Takes the size of file or folder in bytes and returns size formatted in
    KB, MB, GB, TB or PB.
    :params bytes:
        Size of the file in bytes
    """
    alternative = [
        (1024 ** 5, ' PB'),
        (1024 ** 4, ' TB'),
        (1024 ** 3, ' GB'),
        (1024 ** 2, ' MB'),
        (1024 ** 1, ' KB'),
        (1024 ** 0, (' byte', ' bytes')),
    ]

    for factor, suffix in alternative:
        if bytes >= factor:
            break
    amount = int(bytes / factor)
    if isinstance(suffix, tuple):
        singular, multiple = suffix
        if amount == 1:
            suffix = singular
        else:
            suffix = multiple
    return "%s%s" % (str(amount), suffix)
```

바이트단위를 계산해줌

```python
def print_status(progress, file_size, start):
    """
    This function - when passed as `on_progress` to `Video.download` - prints
    out the current download progress.
    :params progress:
        The lenght of the currently downloaded bytes.
    :params file_size:
        The total size of the video.
    :params start:
        The time when started
    """

    percent_done = int(progress) * 100. / file_size
    done = int(50 * progress / int(file_size))
    dt = (clock() - start)
    if dt > 0:
        stdout.write("\r  [%s%s][%3.2f%%] %s at %s/s\r " %
                     ('=' * done, ' ' * (50 - done), percent_done,
                      sizeof(file_size), sizeof(progress // dt)))
    stdout.flush()
```

파일다운로드시간