Unittest for `media`

January 30, 2020

Contents

1 Test Information

1.1 Test Candidate Information

The Module media is designed to help on all issues with media files, like tags (e.g. exif, id3) and transformations. For more Information read the documentation.

Library Information	
Name	media
State	Released
Supported Interpreters	python3
Version	f44d258110e088423a3132bd52534f53
Dependencies	

1.2 Unittest Information

Unittest Information	
Version	8c004e5e652fd9d0db633e752ce36782
Testruns with	python 3.6.9 (final)

1.3 Test System Information

System Information	
Architecture	64bit
Distribution	LinuxMint 19.3 tricia
Hostname	ahorn
Kernel	5.3.0-28-generic (#30 18.04.1-Ubuntu SMP Fri Jan 17 06:14:09 UTC 2020)
Machine	x86_64
Path	/user_data/data/dirk/prj/unittest/media/unittest
System	Linux
Username	dirk

2 Statistic

2.1 Test-Statistic for testrun with python 3.6.9 (final)

Number of tests	**1**
Number of successfull tests	**1**
Number of possibly failed tests	**0**
Number of failed tests	**0**
Executionlevel	Full Test (all defined tests)
Time consumption	0.463s

2.2 Coverage Statistic

Module- or Filename	Line-Coverage	Branch-Coverage
media	98.6%	97.5%
media.__init__.py	100.0%	
media.metadata.py	98.4%	

3 Tested Requirements

3.1 Metadata

3.1.1 Method to get Metadata

Description

A Method shall return the metadata for a given media filename.

Testresult

This test was passed with the state: **Success**. See also full trace in section A.1.1!

Testrun:	python 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/unittest/media/unittest/src/tests/__init__.py (20)
Start-Time:	2020-01-30 22:06:34,253
Finished-Time:	2020-01-30 22:06:34,716
Time-Consumption	0.463s

Testsummary:

Success	Media data for unknown.txt is correct (Content None and Type is <class 'NoneType'>).
Success	Media data for audio.mp3 is correct (Content {'duration': 236.094694, 'bitrate': 290743, 'artist': 'Kaleo', 'title': 'No Good', 'album': 'A/B', 'track': 1, 'genre': 'Rock', 'year': 2016, 'size': 8580366, 'time': 1451606398, 'tm_is_subst': True} and Type is <class 'dict'>).
Success	Media data for audio_fail_conv.mp3 is correct (Content {'duration': 281.991837, 'bitrate': 228298, 'title': 'Video Games (Album Version Remastered)', 'artist': 'Lana Del Rey', 'album': 'Born To Die', 'genre': 'Pop', 'track': 4, 'year': 2012, 'size': 8047290, 'time': 1325375995, 'tm_is_subst': True} and Type is <class 'dict'>).
Success	Media data for image_exif_gps.jpg is correct (Content {'time': 1518783213, 'exposure_program': 'Program Normal', 'exposure_time': 0.000535, 'flash': 'Auto Off', 'aperture': 2.2, 'focal_length': 4.5, 'gps': {'lon': 12.140646934444444, 'lat': 53.68635940527778}, 'height': 2240, 'iso': 50, 'orientation': 0, 'width': 3968, 'size': 4342955, 'camera': 'HUAWEI: EVA-L09'} and Type is <class 'dict'>).
Success	Media data for image_exif_no_gps.jpg is correct (Content {'time': 1515143529, 'exposure_program': 'Program Normal', 'exposure_time': 0.03, 'flash': 'Fired', 'aperture': 2.2, 'focal_length': 4.5, 'height': 3968, 'iso': 160, 'orientation': 0, 'width': 2976, 'size': 2837285, 'camera': 'HUAWEI: EVA-L09'} and Type is <class 'dict'>).
Success	Media data for image_non_exif.jpg is correct (Content {'size': 1139092, 'time': 1449870515, 'tm_is_subst': True} and Type is <class 'dict'>).
Success	Media data for image_extraction_failed.jpg is correct (Content {'time': 1226149915, 'exposure_program': 'Program Normal', 'exposure_time': 0.008, 'flash': 'Fill Fired', 'aperture': 7.1, 'focal_length': 170.0, 'height': 2592, 'iso': 400, 'orientation': 1, 'width': 3888, 'size': 1301272, 'camera': 'Canon: Canon EOS 40D'} and Type is <class 'dict'>).
Success	Media data for video.3gp is correct (Content {'width': 800, 'height': 480, 'ratio': 1.6666666666666667, 'duration': 3.964, 'bitrate': 2341765, 'time': 1414948303, 'size': 1160345} and Type is <class 'dict'>).
Success	Media data for video.mp4 is correct (Content {'width': 1920, 'height': 1080, 'ratio': 1.7777777777777777, 'duration': 12.453, 'bitrate': 17883888, 'time': 1503125482, 'size': 27838508} and Type is <class 'dict'>).

Success	Media data for video_special_time.avi is correct (Content {'width': 320, 'height': 240, 'ratio': 0.0, 'duration': 26.531264, 'bitrate': 840554, 'time': 1086778620, 'size': 2787622} and Type is <class 'dict'>).
Success	Media data for video_no_date.avi is correct (Content {'width': 640, 'height': 480, 'ratio': 1.3333333333333333, 'duration': 11.016, 'bitrate': 2153411, 'size': 2965248, 'time': 1158528375, 'tm_is_subst': True} and Type is <class 'dict'>).

A Trace for testrun with python 3.6.9 (final)

A.1 Tests with status Info (1)

A.1.1 Method to get Metadata

Description

A Method shall return the metadata for a given media filename.

Testresult

This test was passed with the state: **Success**.

Success	Media data for unknown.txt is correct (Content None and Type is <class 'NoneType'>).

```
Filetype not known: /user_data/data/dirk/prj/unittest/media/unittest/input_data/unknown.txt
Result (Media data for unknown.txt): None (<class 'NoneType'>)
Expectation (Media data for unknown.txt): result = None (<class 'NoneType'>)
```

Success	Media data for audio.mp3 is correct (Content {'duration': 236.094694, 'bitrate': 290743, 'artist': 'Kaleo', 'title': 'No Good', 'album': 'A/B', 'track': 1, 'genre': 'Rock', 'year': 2016, 'size': 8580366, 'time': 1451606398, 'tm_is_subst': True} and Type is <class 'dict'>).

```
Result (Media data for audio.mp3): { 'duration': 236.094694, 'bitrate': 290743, 'artist':
↪  'Kaleo', 'title': 'No Good', 'album': 'A/B', 'track': 1, 'genre': 'Rock', 'year': 2016,
↪  'size': 8580366, 'time': 1451606398, 'tm_is_subst': True } (<class 'dict'>)
Expectation (Media data for audio.mp3): result = { 'duration': 236.094694, 'bitrate': 290743,
↪  'artist': 'Kaleo', 'title': 'No Good', 'album': 'A/B', 'track': 1, 'genre': 'Rock',
↪  'year': 2016, 'time': 1451606398, 'tm_is_subst': True, 'size': 8580366 } (<class 'dict'>)
```

Success	Media data for audio_fail_conv.mp3 is correct (Content {'duration': 281.991837, 'bitrate': 228298, 'title': 'Video Games (Album Version Remastered)', 'artist': 'Lana Del Rey', 'album': 'Born To Die', 'genre': 'Pop', 'track': 4, 'year': 2012, 'size': 8047290, 'time': 1325375995, 'tm_is_subst': True} and Type is <class 'dict'>).

```
Can't convert 'N/A' (bitrate) for bitrate
Result (Media data for audio_fail_conv.mp3): { 'duration': 281.991837, 'bitrate': 228298,
↪  'title': 'Video Games (Album Version Remastered)', 'artist': 'Lana Del Rey', 'album':
↪  'Born To Die', 'genre': 'Pop', 'track': 4, 'year': 2012, 'size': 8047290, 'time':
↪  1325375995, 'tm_is_subst': True } (<class 'dict'>)
Expectation (Media data for audio_fail_conv.mp3): result = { 'duration': 281.991837,
↪  'bitrate': 228298, 'artist': 'Lana Del Rey', 'title': 'Video Games (Album Version
↪  Remastered)', 'album': 'Born To Die', 'track': 4, 'genre': 'Pop', 'year': 2012, 'time':
↪  1325375995, 'tm_is_subst': True, 'size': 8047290 } (<class 'dict'>)
```

Success	Media data for image_exif_gps.jpg is correct (Content {'time': 1518783213, 'exposure_program': 'Program Normal', 'exposure_time': 0.000535, 'flash': 'Auto Off', 'aperture': 2.2, 'focal_length': 4.5, 'gps': {'lon': 12.140646934444444, 'lat': 53.68635940527778}, 'height': 2240, 'iso': 50, 'orientation': 0, 'width': 3968, 'size': 4342955, 'camera': 'HUAWEI: EVA-L09'} and Type is <class 'dict'>).

```
Result (Media data for image_exif_gps.jpg): { 'time': 1518783213, 'exposure_program':
↪  'Program Normal', 'exposure_time': 0.000535, 'flash': 'Auto Off', 'aperture': 2.2,
↪  'focal_length': 4.5, 'gps': { 'lon': 12.140646934444444, 'lat': 53.68635940527778 },
↪  'height': 2240, 'iso': 50, 'orientation': 0, 'width': 3968, 'size': 4342955, 'camera':
↪  'HUAWEI: EVA-L09' } (<class 'dict'>)
```

```
Expectation (Media data for image_exif_gps.jpg): result = { 'time': 1518783213,
↪  'exposure_program': 'Program Normal', 'exposure_time': 0.000535, 'flash': 'Auto Off',
↪  'aperture': 2.2, 'focal_length': 4.5, 'gps': { 'lon': 12.140646934444444, 'lat':
↪  53.68635940527778 }, 'height': 2240, 'iso': 50, 'orientation': 0, 'width': 3968,
↪  'camera': 'HUAWEI: EVA-L09', 'size': 4342955 } (<class 'dict'>)
```

Success Media data for image_exif_no_gps.jpg is correct (Content {'time': 1515143529, 'exposure_program': 'Program Normal', 'exposure_time': 0.03, 'flash': 'Fired', 'aperture': 2.2, 'focal_length': 4.5, 'height': 3968, 'iso': 160, 'orientation': 0, 'width': 2976, 'size': 2837285, 'camera': 'HUAWEI: EVA-L09'} and Type is <class 'dict'>).

```
Result (Media data for image_exif_no_gps.jpg): { 'time': 1515143529, 'exposure_program':
↪  'Program Normal', 'exposure_time': 0.03, 'flash': 'Fired', 'aperture': 2.2,
↪  'focal_length': 4.5, 'height': 3968, 'iso': 160, 'orientation': 0, 'width': 2976, 'size':
↪  2837285, 'camera': 'HUAWEI: EVA-L09' } (<class 'dict'>)
```

```
Expectation (Media data for image_exif_no_gps.jpg): result = { 'time': 1515143529,
↪  'exposure_program': 'Program Normal', 'exposure_time': 0.03, 'flash': 'Fired',
↪  'aperture': 2.2, 'focal_length': 4.5, 'height': 3968, 'iso': 160, 'orientation': 0,
↪  'width': 2976, 'camera': 'HUAWEI: EVA-L09', 'size': 2837285 } (<class 'dict'>)
```

Success Media data for image_non_exif.jpg is correct (Content {'size': 1139092, 'time': 1449870515, 'tm_is_subst': True} and Type is <class 'dict'>).

```
/user_data/data/dirk/prj/unittest/media/unittest/input_data/image_non_exif.jpg does not have
↪  any exif information
```

```
Result (Media data for image_non_exif.jpg): { 'size': 1139092, 'time': 1449870515,
↪  'tm_is_subst': True } (<class 'dict'>)
```

```
Expectation (Media data for image_non_exif.jpg): result = { 'time': 1449870515,
↪  'tm_is_subst': True, 'size': 1139092 } (<class 'dict'>)
```

Success Media data for image_extraction_failed.jpg is correct (Content {'time': 1226149915, 'exposure_program': 'Program Normal', 'exposure_time': 0.008, 'flash': 'Fill Fired', 'aperture': 7.1, 'focal_length': 170.0, 'height': 2592, 'iso': 400, 'orientation': 1, 'width': 3888, 'size': 1301272, 'camera': 'Canon: Canon EOS 40D'} and Type is <class 'dict'>).

```
GPS data extraction failed for {0: b'\x02\x02\x00\x00'}
```

```
Result (Media data for image_extraction_failed.jpg): { 'time': 1226149915,
↪  'exposure_program': 'Program Normal', 'exposure_time': 0.008, 'flash': 'Fill Fired',
↪  'aperture': 7.1, 'focal_length': 170.0, 'height': 2592, 'iso': 400, 'orientation': 1,
↪  'width': 3888, 'size': 1301272, 'camera': 'Canon: Canon EOS 40D' } (<class 'dict'>)
```

```
Expectation (Media data for image_extraction_failed.jpg): result = { 'time': 1226149915,
↪  'exposure_program': 'Program Normal', 'exposure_time': 0.008, 'flash': 'Fill Fired',
↪  'aperture': 7.1, 'focal_length': 170.0, 'height': 2592, 'iso': 400, 'orientation': 1,
↪  'width': 3888, 'camera': 'Canon: Canon EOS 40D', 'size': 1301272 } (<class 'dict'>)
```

Success Media data for video.3gp is correct (Content {'width': 800, 'height': 480, 'ratio': 1.6666666666666667, 'duration': 3.964, 'bitrate': 2341765, 'time': 1414948303, 'size': 1160345} and Type is <class 'dict'>).

```
Result (Media data for video.3gp): { 'width': 800, 'height': 480, 'ratio':
↪  1.6666666666666667, 'duration': 3.964, 'bitrate': 2341765, 'time': 1414948303, 'size':
↪  1160345 } (<class 'dict'>)
```

```
Expectation (Media data for video.3gp): result = { 'width': 800, 'height': 480, 'ratio':
↪  1.6666666666666667, 'duration': 3.964, 'bitrate': 2341765, 'time': 1414948303, 'size':
↪  1160345 } (<class 'dict'>)
```

Success Media data for video.mp4 is correct (Content {'width': 1920, 'height': 1080, 'ratio': 1.7777777777777777, 'duration': 12.453, 'bitrate': 17883888, 'time': 1503125482, 'size': 27838508} and Type is <class 'dict'>).

```
Result (Media data for video.mp4): { 'width': 1920, 'height': 1080, 'ratio':
↪  1.7777777777777777, 'duration': 12.453, 'bitrate': 17883888, 'time': 1503125482, 'size':
↪  27838508 } (<class 'dict'>)
```

```
Expectation (Media data for video.mp4): result = { 'width': 1920, 'height': 1080, 'ratio':
↪  1.7777777777777777, 'duration': 12.453, 'bitrate': 17883888, 'time': 1503125482, 'size':
↪  27838508 } (<class 'dict'>)
```

Success Media data for video_special_time.avi is correct (Content {'width': 320, 'height': 240, 'ratio': 0.0, 'duration': 26.531264, 'bitrate': 840554, 'time': 1086778620, 'size': 2787622} and Type is <class 'dict'>).

```
Can't convert 'N/A' (duration) for duration
```

```
Result (Media data for video_special_time.avi): { 'width': 320, 'height': 240, 'ratio': 0.0,
↪  'duration': 26.531264, 'bitrate': 840554, 'time': 1086778620, 'size': 2787622 } (<class
↪  'dict'>)
```

```
Expectation (Media data for video_special_time.avi): result = { 'width': 320, 'height': 240,
↪  'ratio': 0.0, 'duration': 26.531264, 'bitrate': 840554, 'time': 1086778620, 'size':
↪  2787622 } (<class 'dict'>)
```

Success Media data for video_no_date.avi is correct (Content {'width': 640, 'height': 480, 'ratio': 1.3333333333333333, 'duration': 11.016, 'bitrate': 2153411, 'size': 2965248, 'time': 1158528375, 'tm_is_subst': True} and Type is <class 'dict'>).

```
Result (Media data for video_no_date.avi): { 'width': 640, 'height': 480, 'ratio':
↪  1.3333333333333333, 'duration': 11.016, 'bitrate': 2153411, 'size': 2965248, 'time':
↪  1158528375, 'tm_is_subst': True } (<class 'dict'>)
Expectation (Media data for video_no_date.avi): result = { 'width': 640, 'height': 480,
↪  'ratio': 1.3333333333333333, 'duration': 11.016, 'bitrate': 2153411, 'time': 1158528375,
↪  'tm_is_subst': True, 'size': 2965248 } (<class 'dict'>)
```

B Test-Coverage

B.1 media

The line coverage for media was 98.6%
The branch coverage for media was 97.5%

B.1.1 media.__init__.py

The line coverage for media.__init__.py was 100.0%
The branch coverage for media.__init__.py was 97.5%

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#
"""
media (Media Tools)
===================

**Author:**

* Dirk Alders <sudo-dirk@mount-mockery.de>

**Description:**

    This module helps on all issues with media files, like tags (e.g. exif, id3) and
    transformations.

**Submodules:**

* :func:`media.get_media_data`

**Unittest:**

    See also the :download:`unittest <../../media/_testresults_/unittest.pdf>` documentation.
"""
__DEPENDENCIES__ = []

import logging

logger_name = 'MEDIA'
logger = logging.getLogger(logger_name)


__DESCRIPTION__ = """The Module {\\tt %s} is designed to help on all issues with media files,
    like tags (e.g. exif, id3) and transformations.
For more Information read the documentation.""" % __name__.replace('_', '\_')
"""The Module Description"""
__INTERPRETER__ = (3, )
```

```
"""The Tested Interpreter-Versions"""


def get_media_data(full_path):
    from media import metadata
    ft = metadata.get_filetype(full_path)
    #
    if ft == metadata.FILETYPE_AUDIO:
        return metadata.get_audio_data(full_path)
    elif ft == metadata.FILETYPE_IMAGE:
        return metadata.get_image_data(full_path)
    elif ft == metadata.FILETYPE_VIDEO:
        return metadata.get_video_data(full_path)
    else:
        logger.warning('Filetype not known: %s', full_path)
```

B.1.2 media.metadata.py

The line coverage for media.metadata.py was 98.4%
The branch coverage for media.metadata.py was 97.5%

```
import logging
import os
from PIL import Image
import subprocess
import time


logger_name = 'MEDIA'
logger = logging.getLogger(logger_name)

FILETYPE_AUDIO = 'audio'
FILETYPE_IMAGE = 'image'
FILETYPE_VIDEO = 'video'

EXTENTIONS_AUDIO = ['.mp3', ]
EXTENTIONS_IMAGE = ['.jpg', '.jpeg', '.jpe', '.png', '.tif', '.tiff', '.gif', ]
EXTENTIONS_VIDEO = ['.avi', '.mpg', '.mpeg', '.mpe', '.mov', '.qt', '.mp4', '.webm', '.ogv', '.
    flv', '.3gp', ]

KEY_ALBUM = 'album'
KEY_APERTURE = 'aperture'
KEY_ARTIST = 'artist'
KEY_BITRATE = 'bitrate'
KEY_CAMERA = 'camera'
KEY_DURATION = 'duration'
KEY_EXPOSURE_PROGRAM = 'exposure_program'
KEY_EXPOSURE_TIME = 'exposure_time'
KEY_FLASH = 'flash'
KEY_FOCAL_LENGTH = 'focal_length'
KEY_GENRE = 'genre'
KEY_GPS = 'gps'
KEY_HEIGHT = 'height'
KEY_ISO = 'iso'
KEY_ORIENTATION = 'orientation'
KEY_RATIO = 'ratio'
KEY_SIZE = 'size'
KEY_TIME = 'time'   # USE time.localtime(value) or datetime.fromtimestamp(value) to convert the
    timestamp
KEY_TIME_IS_SUBSTITUTION = 'tm_is_subst'
KEY_TITLE = 'title'
KEY_TRACK = 'track'
KEY_WIDTH = 'width'
KEY_YEAR = 'year'
```

```
42
43 __KEY_CAMERA_VENDOR__ = 'camera_vendor'
44 __KEY_CAMERA_MODEL__ = 'camera_model'
45
46
47 def get_filetype(full_path):
48     ext = os.path.splitext(full_path)[1]
49     if ext in EXTENTIONS_AUDIO:
50         return FILETYPE_AUDIO
51     elif ext in EXTENTIONS_IMAGE:
52         return FILETYPE_IMAGE
53     elif ext in EXTENTIONS_VIDEO:
54         return FILETYPE_VIDEO
55
56
57 def get_audio_data(full_path):
58     conv_key_dict = {}
59     conv_key_dict['album'] = (str, KEY_ALBUM)
60     conv_key_dict['TAG:album'] = (str, KEY_ALBUM)
61     conv_key_dict['TAG:artist'] = (str, KEY_ARTIST)
62     conv_key_dict['artist'] = (str, KEY_ARTIST)
63     conv_key_dict['bit_rate'] = (__int_conv__, KEY_BITRATE)
64     conv_key_dict['duration'] = (float, KEY_DURATION)
65     conv_key_dict['TAG:genre'] = (str, KEY_GENRE)
66     conv_key_dict['genre'] = (str, KEY_GENRE)
67     conv_key_dict['TAG:title'] = (str, KEY_TITLE)
68     conv_key_dict['title'] = (str, KEY_TITLE)
69     conv_key_dict['TAG:track'] = (__int_conv__, KEY_TRACK)
70     conv_key_dict['track'] = (__int_conv__, KEY_TRACK)
71     conv_key_dict['TAG:date'] = (__int_conv__, KEY_YEAR)
72     conv_key_dict['date'] = (__int_conv__, KEY_YEAR)
73     return __adapt__data__(__get_xxprobe_data__(full_path, conv_key_dict), full_path)
74
75
76 def get_video_data(full_path):
77     conv_key_dict = {}
78     conv_key_dict['creation_time'] = (__vid_datetime_conv__, KEY_TIME)
79     conv_key_dict['TAG:creation_time'] = (__vid_datetime_conv__, KEY_TIME)
80     conv_key_dict['bit_rate'] = (__int_conv__, KEY_BITRATE)
81     conv_key_dict['duration'] = (float, KEY_DURATION)
82     conv_key_dict['height'] = (__int_conv__, KEY_HEIGHT)
83     conv_key_dict['width'] = (__int_conv__, KEY_WIDTH)
84     conv_key_dict['display_aspect_ratio'] = (__ratio_conv__, KEY_RATIO)
85     return __adapt__data__(__get_xxprobe_data__(full_path, conv_key_dict), full_path)
86
87
88 def get_image_data(full_path):
89     return __adapt__data__(__get_exif_data__(full_path), full_path)
90
91
92 def __adapt__data__(data, full_path):
93     data[KEY_SIZE] = os.path.getsize(full_path)
94     # Join Camera Vendor and Camera Model
95     if __KEY_CAMERA_MODEL__ in data and __KEY_CAMERA_VENDOR__ in data:
96         model = data.pop(__KEY_CAMERA_MODEL__)
97         vendor = data.pop(__KEY_CAMERA_VENDOR__)
98         data[KEY_CAMERA] = '%s: %s' % (vendor, model)
99     # Add time if not exists
100    if KEY_TIME not in data:
101        if KEY_YEAR in data and KEY_TRACK in data:
102            # Use a date where track 1 is the newest in the given year
```

```
103             minute = int(data[KEY_TRACK] / 60)
104             second = (data[KEY_TRACK] - 60 * minute) % 60
105             #
106             data[KEY_TIME] = int(time.mktime((data[KEY_YEAR], 1, 1, 0, 59 - minute, 59 - second,
    0, 0, 0)))
107             data[KEY_TIME_IS_SUBSTITUTION] = True
108         else:
109             data[KEY_TIME] = int(os.path.getmtime(full_path))
110             data[KEY_TIME_IS_SUBSTITUTION] = True
111     return data
112
113
114 def __get_xxprobe_data__(full_path, conv_key_dict):
115     def _ffprobe_command(full_path):
116         return ['ffprobe', '-v', 'quiet', '-show_format', '-show_streams', full_path]
117
118     def _avprobe_command(full_path):
119         return ['avprobe', '-v', 'quiet', '-show_format', '-show_streams', full_path]
120
121     try:
122         xxprobe_text = subprocess.check_output(_avprobe_command(full_path))
123     except FileNotFoundError:
124         try:
125             xxprobe_text = subprocess.check_output(_ffprobe_command(full_path))
126         except FileNotFoundError:
127             logger.warning('ffprobe and avprobe seem to be not installed')
128             return {}
129     #
130     rv = {}
131     for line in xxprobe_text.decode('utf-8').splitlines():
132         try:
133             key, val = [snippet.strip() for snippet in line.split('=')]
134         except ValueError:
135             continue
136         else:
137             if key in conv_key_dict:
138                 tp, name = conv_key_dict[key]
139                 try:
140                     rv[name] = tp(val)
141                 except ValueError:
142                     logger.log(logging.WARNING if val else logger.INFO, 'Can\'t convert %s (%s)
    for %s', repr(val), name, name)
143     return rv
144
145
146 def __get_exif_data__(full_path):
147     rv = {}
148     im = Image.open(full_path)
149     try:
150         exif = dict(im._getexif().items())
151     except AttributeError:
152         logger.debug('%s does not have any exif information', full_path)
153     else:
154         conv_key_dict = {}
155         # IMAGE
156         conv_key_dict[0x9003] = (__datetime_conv__, KEY_TIME)
157         conv_key_dict[0x8822] = (__exposure_program_conv__, KEY_EXPOSURE_PROGRAM)
158         conv_key_dict[0x829A] = (__num_denum_conv__, KEY_EXPOSURE_TIME)
159         conv_key_dict[0x9209] = (__flash_conv__, KEY_FLASH)
160         conv_key_dict[0x829D] = (__num_denum_conv__, KEY_APERTURE)
161         conv_key_dict[0x920A] = (__num_denum_conv__, KEY_FOCAL_LENGTH)
162         conv_key_dict[0x8825] = (__gps_conv__, KEY_GPS)
```

```
        conv_key_dict[0xA003] = (__int_conv__, KEY_HEIGHT)
        conv_key_dict[0x8827] = (__int_conv__, KEY_ISO)
        conv_key_dict[0x010F] = (str, __KEY_CAMERA_VENDOR__)
        conv_key_dict[0x0110] = (str, __KEY_CAMERA_MODEL__)
        conv_key_dict[0x0112] = (__int_conv__, KEY_ORIENTATION)
        conv_key_dict[0xA002] = (__int_conv__, KEY_WIDTH)
        for key in conv_key_dict:
            if key in exif:
                tp, name = conv_key_dict[key]
                value = tp(exif[key])
                if value is not None:
                    rv[name] = value
    return rv


# TODO: Join datetime converter __datetime_conv__ and __vid_datetime_conv_
def __datetime_conv__(dt):
    format_string = "%Y:%m:%d %H:%M:%S"
    return int(time.mktime(time.strptime(dt, format_string)))


def __vid_datetime_conv__(dt):
    try:
        dt = dt[:dt.index('.')]
    except ValueError:
        pass  # time string seems to have no '.'
    dt = dt.replace('T', ' ').replace('/', '').replace('\\', '')
    if len(dt) == 16:
        dt += ':00'
    format_string = '%Y-%m-%d %H:%M:%S'
    return int(time.mktime(time.strptime(dt, format_string)))


def __exposure_program_conv__(n):
    return {
        0: 'Unidentified',
        1: 'Manual',
        2: 'Program Normal',
        3: 'Aperture Priority',
        4: 'Shutter Priority',
        5: 'Program Creative',
        6: 'Program Action',
        7: 'Portrait Mode',
        8: 'Landscape Mode'
    }.get(n, None)


def __flash_conv__(n):
    return {
        0: 'No',
        1: 'Fired',
        5: 'Fired (?)',  # no return sensed
        7: 'Fired (!)',  # return sensed
        9: 'Fill Fired',
        13: 'Fill Fired (?)',
        15: 'Fill Fired (!)',
        16: 'Off',
        24: 'Auto Off',
        25: 'Auto Fired',
        29: 'Auto Fired (?)',
        31: 'Auto Fired (!)',
```

```
224         32: 'Not Available'
225     }.get(n, None)
226
227
228 def __int_conv__(value):
229     try:
230         return int(value)
231     except ValueError:
232         for c in ['.', '/', '-']:
233             p = value.find(c)
234             if p >= 0:
235                 value = value[:p]
236     return int(value)
237
238
239 def __num_denum_conv__(data):
240     num, denum = data
241     return num / denum
242
243
244 def __gps_conv__(data):
245     def lat_lon_cal(lon_or_lat):
246         lon_lat = 0.
247         fac = 1.
248         for num, denum in lon_or_lat:
249             lon_lat += float(num) / float(denum) * fac
250             fac *= 1. / 60.
251         return lon_lat
252     try:
253         lon = lat_lon_cal(data[0x0004])
254         lat = lat_lon_cal(data[0x0002])
255         if lon != 0 or lat != 0:    # do not use lon and lat equal 0, caused by motorola gps
    weakness
256             return {'lon': lon, 'lat': lat}
257     except KeyError:
258         logger.warning('GPS data extraction failed for %s', repr(data))
259
260
261 def __ratio_conv__(ratio):
262     ratio = ratio.replace('\\', '')
263     num, denum = ratio.split(':')
264     return float(num) / float(denum)
```