

## \* 과제내용

-> 사진에 들어가 있어야 할 정보, GPS 정보, JPG 형태로 저장합니다.

-> 해당 사진의 GPS 정보를 해당 BOB 센터가 아닌, 다른 지역(서울 제외) 으로 조작합니다.

-> 분석 내용

1) JPG 파일에 들어가 있는 GPS 관련 정보, 형식 및 자신의 사진에서 실제 GPS가 있는 파일에서의 Offset 등을 꼭 포함해서 조사합니다.

2) 디지털 포렌식 관점에서 어떻게 하면 조작할 수 있는지 논리적인 근거하에 작성합니다.



[사진] 취약점 트랙 6기 교육생 황상두

날짜

2017년 7월 15일 오후 4:09

위치

대한민국 서울특별시 강남구 역삼1동  
800-61

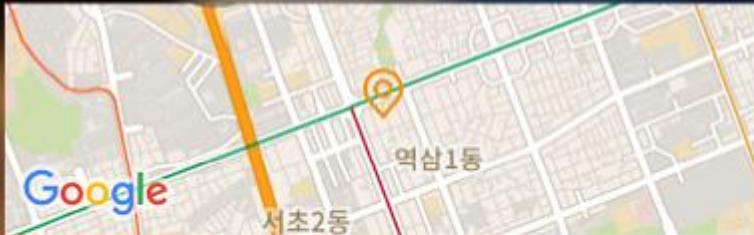


사진 종류

사람

추가정보

제목 : 20170715\_160942.jpg

크기 : 1.41MB

해상도 : 2592x1944

회전 : 270°

경로 : /내 디바이스/DCIM/Camera

[위치정보]대한민국 서울 특별시 강남구 역삼1동

# Python Imaging Library (PIL)

## Python 라이브러리를 이용

There is no GPS info in this picture

```
C:\Users\Hwang\Downloads\Imaging-1.1.7>python gps추출.py
There is GPS info in this picture located at 37.497222222,127.029722222
kml file created
```

'|

다운로드 > Imaging-1.1.7			▼	↺
이름	수정한 날짜	유형		
_imagingtk	2009-11-01 오전 9...	C Source file		
BUILDME	2009-11-02 오전 7...	파일		
CHANGES	2009-11-16 오전 1...	파일		
CONTENTS	2009-11-02 오후 8...	파일		
decode	2009-11-02 오후 8...	C Source file		
display	2009-11-01 오전 9...	C Source file		
doctest	2009-11-01 오전 9...	Python File		
encode	2009-11-02 오후 8...	C Source file		
gps추출	2017-07-15 오후 4...	Python File		
hsd	2017-07-15 오전 7...	JPG 파일		
hsd.jpg.kml	2017-07-15 오후 4...	KML 파일		
MANIFEST	2009-11-16 오전 3...	파일		
map	2009-11-01 오전 9...	C Source file		
outline	2009-11-01 오전 9...	C Source file		
path	2009-11-01 오전 9...	C Source file		
PIL.pth	2009-11-01 오전 9...	PTH 파일		
README	2009-11-16 오전 1...	파일		
selftest	2009-11-01 오전 9...	Python File		

193바이트

## KML Viewer



실제와 동일하게 나옴.

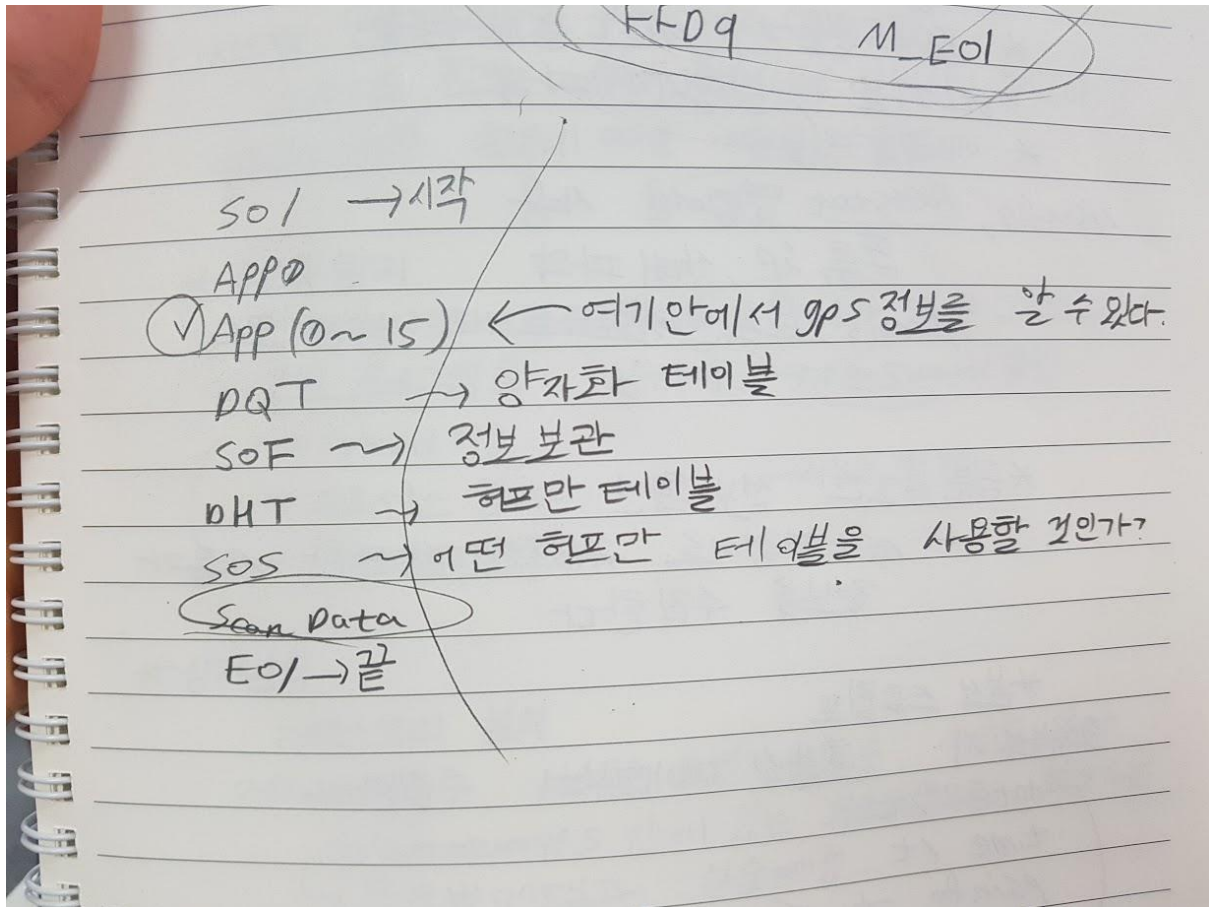
```
from PIL import Image
from PIL.ExifTags import TAGS
filename = "hsd.jpg"
extension = filename.split('.')[-1]
if (extension == 'jpg') | (extension == 'JPG') | (extension == 'jpeg') | (extension == 'JPEG'):
    try:
        img = Image.open(filename)
        info = img._getexif()
        exif = {}
        for tag, value in info.items():
            decoded = TAGS.get(tag, tag)
            exif[decoded] = value
        # from the exif data, extract gps
        exifGPS = exif['GPSInfo']
        latData = exifGPS[2]
        lonData = exifGPS[4]
        # calculate the lat / long
        latDeg = latData[0][0] / float(latData[0][1])
        latMin = latData[1][0] / float(latData[1][1])
        latSec = latData[2][0] / float(latData[2][1])
        lonDeg = lonData[0][0] / float(lonData[0][1])
        lonMin = lonData[1][0] / float(lonData[1][1])
        lonSec = lonData[2][0] / float(lonData[2][1])
        # correct the lat/lon based on N/E/W/S
        Lat = (latDeg + (latMin + latSec / 60.0) / 60.0)
        if exifGPS[1] == 'S': Lat = Lat * -1
        Lon = (lonDeg + (lonMin + lonSec / 60.0) / 60.0)
        if exifGPS[3] == 'W': Lon = Lon * -1
        # print file
        msg = "There is GPS info in this picture located at " + str(Lat) + "," + str(Lon)
        print msg
```

## \* 실행결과

```
C:\Users\Hwang\Downloads\Imaging-1.1.7>py gps추출.py hsd.jpg
There is GPS info in this picture located at 37.497222222,127.029722222
kml file created
```

gps정보가 추출된 것을 볼 수 있습니다.

## ● Jpg 헤더 분석



[참조] <http://f10ckf10ck.tistory.com/253>

Exif Format ← GPS정보가 있는 곳입니다.

FF D8 FF E0 00 10 4A 46 49 46 00 01 01 00 00 01	yøya..JFIF....
00 01 00 00 FF E1 00 2A 45 78 69 66 00 00 49 49	....vá.*Exif..II
2A 00 08 00 00 00 01 00 09 92 02 00 07 00 00 00	*.....
1A 00 00 00 00 00 00 00 47 6F 6F 67 6C 65 00 00	.....Google..
FF DB 00 84 00 03 02 02 0A 0A 0A 0A 0A 0A 0A 0A	vũ.....

	name	value	start	size	color
✓	struct JPGFILE jpgfile		0h	1342Eh	Fg: Bg:
	enum M_ID SOIMarker	M_SOI (FFD8h)	0h	2h	Fg: Bg:
	> struct APP0 app0		2h	12h	Fg: Bg:
	> struct APP1 app1		14h	2Ch	Fg: Bg:
	> struct DQT dat		40h	86h	Fg: Bg:
	> struct SOF sof0		C6h	13h	Fg: Bg:
	> struct DHT dht[0]		D9h	1Fh	Fg: Bg:
	> struct DHT dht[1]		F8h	5Ch	Fg: Bg:
	> struct DHT dht[2]		154h	1Eh	Fg: Bg:
	> struct DHT dht[3]		172h	40h	Fg: Bg:
	> struct SOS scanStart		1B2h	Bh	Fg: Bg:
	> char scanData[78447]		1BDh	1326Fh	Fg: Bg:
	enum M_ID EOIMarker	M_EOI (FFD9h)	1342Ch	2h	Fg: Bg:



struct APP1 app1		2h	12h	Fg:	Bg:
▼ struct APP1 app1		14h	2Ch	Fg:	Bg:
enum M_ID marker	M_APP1 (FFE1h)	14h	2h	Fg:	Bg:
WORD szSection	42	16h	2h	Fg:	Bg:
char EXIF[6]	Exif	18h	6h	Fg:	Bg:
byte align[2]	11	1Eh	2h	Fg:	Bg:
WORD tagMark	42	20h	2h	Fg:	Bg:
DWORD offsetFirstIFD	8	22h	4h	Fg:	Bg:
struct IFD ifdMainImage		26h	12h	Fg:	Bg:

▼ struct IFD ifdMainImage		26h	12h	Fg:	Bg:
WORD nDirEntry	1	26h	2h	Fg:	Bg:
struct DIRENTRY dirEntry	Tag# = 0x9209 (Flash)	28h	Ch	Fg:	Bg:
DWORD nextIFDoffset	0	34h	4h	Fg:	Bg:
struct StrAscii strAscii	Google	38h	7h	Fg:	Bg:

## IFD(Image File Directory)의 구조

디렉토리 엔트리(Directory Entry)의 개수 (2bytes)			
태그 번호 (2bytes)	변수타입 (2bytes)	구성요소의 수 (4bytes)	데이터 or Offset (4bytes)
태그 번호 (2bytes)	변수타입 (2bytes)	구성요소의 수 (4bytes)	데이터 or Offset (4bytes)
태그 번호 (2bytes)	변수타입 (2bytes)	구성요소의 수 (4bytes)	데이터 or Offset (4bytes)
...	...	...	...
태그 번호 (2bytes)	변수타입 (2bytes)	구성요소의 수 (4bytes)	데이터 or Offset (4bytes)
다음 IFD 까지의 offset (4bytes)			

간단히 요약하자면

1. jpg헤더의 Maker를 통해 APP0의 위치를 알아낸다. (FFEx)의 형태를 가지고 있습니다.
2. 안에서 Exif를 찾아냅니다.
3. 이후 태그값으로 접근한다면 'GPSinfo'의 정보를 알아낼 수 있습니다.

```

0x8298: "Copyright",
0x829a: "ExposureTime",
0x829d: "FNumber",
0x8769: "ExifOffset",
0x8773: "InterColorProfile",
0x8822: "ExposureProgram",
0x8824: "SpectralSensitivity",
0x8825: "GPSInfo",
0x8827: "ISOSpeedRatings",
0x8828: "OECF",
0x8829: "Interlace",
0x882a: "TimeZoneOffset",
0x882b: "SelfTimerMode",
0x9000: "ExifVersion",
0x9003: "DateTimeOriginal",
0x9004: "DateTimeDigitized",
0x9101: "ComponentsConfiguration",
0x9102: "CompressedBitsPerPixel",
0x9201: "ShutterSpeedValue",
0x9202: "ApertureValue",
0x9203: "BrightnessValue",

```

```

GPSTAGS = {
    0: "GPSVersionID",
    1: "GPSLatitudeRef",
    2: "GPSLatitude",
    3: "GPSLongitudeRef",
    4: "GPSLongitude",
    5: "GPSAltitudeRef",
    6: "GPSAltitude",
    7: "GPSTimeStamp",
    8: "GPSSatellites",
    9: "GPSStatus",
    10: "GPSMeasureMode",
    11: "GPSDOP",
    12: "GPSSpeedRef",
    13: "GPSSpeed",
    14: "GPSTrackRef",
    15: "GPSTrack",
    16: "GPSImgDirectionRef",
    17: "GPSImgDirection",
    18: "GPSMapDatum",
    19: "GPSDestLatitudeRef",
    20: "GPSDestLatitude",
    21: "GPSDestLongitudeRef",
    22: "GPSDestLongitude",
    23: "GPSDestBearingRef",
    24: "GPSDestBearing",
    25: "GPSDestDistanceRef",
    26: "GPSDestDistance"
}

```

```

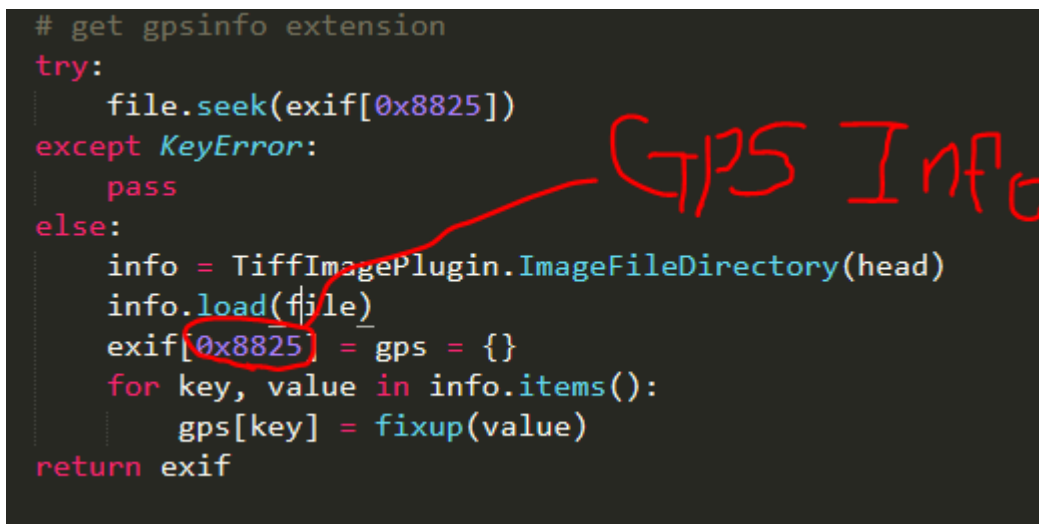
# from the exif data, extract gps
exifGPS = exif['GPSInfo']

latData = exifGPS[2]
lonData = exifGPS[4]
# calculate the lat / long

```

2번과 4번이 위도와 경도를 나타낸다 것을 알 수 있습니다.

조작을 시작하겠습니다.



Exif[0x8825] 안에 조작된 gps값을 넣는다면 조작을 할 수 있겠군요!



Lat = (latDeg + (latMin + latSec / 60.0) / 60.0)

수식을 보니 각각 **도 분 초**를 의미하는 것을 알 수 있습니다. (DMS방식)



$$\frac{latMin}{60} + \frac{latSec}{3600} = 0.xxx$$

$$x - int(x) = \text{소수점} = y$$

$$int(x) = 5$$

$$60y = latMin + \frac{latSec}{60}$$

$$60y - int(60y) = \text{소수점} = \frac{latSec}{60}$$

Degree를 DMS로 변환하기 위한 공식을 나름대로 세워보았습니다.

실제로 찾아보니 제가 생각한 게 맞는 듯 합니다.

[참조] <http://blog.naver.com/PostView.nhn?blogId=shoomi&logNo=90017563523>

이제 코딩을 하겠습니다.

```
x = 37.4087310
y = 127.1324040

Counterfeit_latDeg = int(x)
Counterfeit_latMin = (x - int(x))*60
Counterfeit_latSec = 60*int(Counterfeit_latMin - int(Counterfeit_latMin))

Counterfeit_lonDeg = int(y)
Counterfeit_lonMin = (y - int(y))*60
Counterfeit_lonSec = 60*int(Counterfeit_lonMin - int(Counterfeit_lonMin))

# calculae the lat / long
latDeg = Counterfeit_latDeg
latMin = Counterfeit_latMin
latSec = Counterfeit_latSec

lonDeg = Counterfeit_lonDeg
lonMin = Counterfeit_lonMin
lonSec = Counterfeit_lonSec
```

```
C:\Users\Hwang\Downloads\Imaging-1.1.7>python gps추출.py
There is GPS info in this picture located at 37.408731,127.132404
kml file created
```

조작값과 일치하게 나온 것을 확인할 수 있습니다.

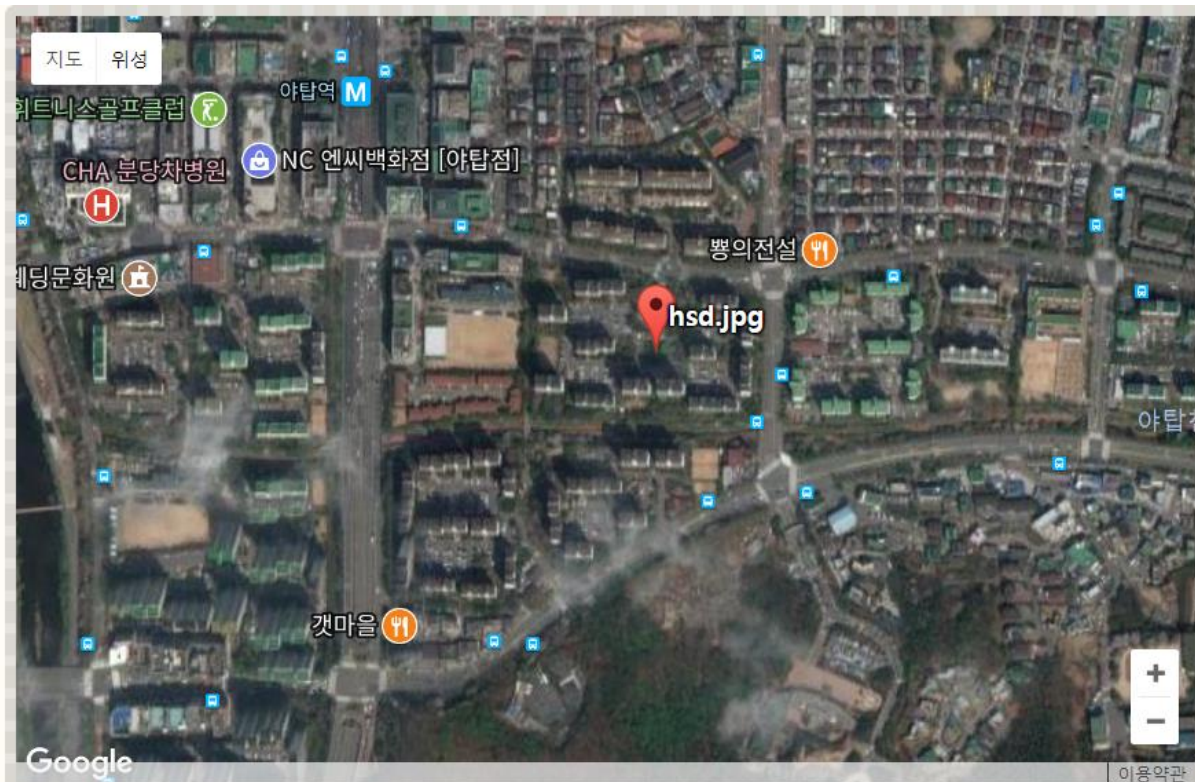
<http://ivanrublev.me/kml/>

## KML Viewer



Drop .kml file here.

File will be processed locally in your browser



저희 집 위치입니다.  
변조에 성공하였습니다.

## ● 변조방법

[참조] <http://fl0ckfl0ck.tistory.com/254>

1. 보여드린 방법처럼 위경도만 변경시키는 방법이 있습니다.
2. 툴(Exif Viewer , Exif Tool , Opanda PowerExif )등을 이용할 수도 있습니다.

그러나 **주의할 사항**이 있습니다.

Exif 필드를 변경했을 때 Modify-Time 필드시간이 변경됩니다. 즉 위경도를 변경한 이후에 Modify-time 필드값까지 변경해주어야 합니다.

바이너리를 조작함에 따라 SHA-1과 MD5의 값이 변경되게 됩니다. 다시 말하면 원본이 있다면 무결성 검증이 가능하게 되기 때문에 조작 후에 원본을 꼭 삭제해야 합니다.

단지 변조하는 것이 아닌 내가 보내고자 하는 수신자가 있다면 개인키와 공개키를 사용하는 PKI 방식을 사용하는 것도 좋은 방법입니다.

다시 말하면 IFD를 개인키로 암호화하고 공개키로 열어볼 수 있도록 하는 것입니다. 공개키를 가지고 있지 않은 이들은 결코 열어볼 수 없을 것입니다.