In [1]:

```
from PIL import Image
import requests
from io import BytesIO
```

In [19]:

```
#!/usr/bin/env python

from PIL import Image

def get_exif(filename):
image = Image.open(filename)
image.verify()
return image._getexif()

exif = get_exif(r'C:\Users\rj100\OneDrive\Pictures\tablerock.jpeg')
print(exif)
```

{36864: b'0231', 37121: b'\x01\x02\x03\x00', 37377: 9.5264271069386, 36867: '2019:12:26 16:51:27',
36868: '2019:12:26 16:51:27', 37378: 2.5260688112781806, 37379: 9.155635150385162, 37380: 0.0, 37383: 5,
37385: 16, 37386: 1.54, 40961: 65535, 40962: 1024, 41988: 1.0223123732251522, 41989: 14, 41990: 0,
36880: '-06:00', 36881: '-06:00', 36882: '-06:00', 37521: '694', 37396: (2015, 1510, 2323, 1392), 37522: '694', 40963:
768, 41495: 2, 271: 'Apple', 272: 'iPhone 11 Pro Max', 33434: 0.0013568521031207597, 274: 1, 531: 1, 33437: 2.4,
41729: b'\x01', 282: 72.0, 283: 72.0, 34850: 2, 34853: {1: 'N', 2: (36.0, 34.0, 35.52), 3: 'W', 4: (93.0, 19.0, 12.0), 5:
b'\x00', 6: 282.0852412821416, 12: 'K', 13: 2.3566701406982804, 16: 'T', 17: 21.746246339791913, 23: 'T', 24:
21.746246339791913, 31: 18.766362291485997}, 34855: 20, 296: 2, 41986: 0, 40960: b'0100', 41987: 0, 305: '13.3',
42034: (1.5399999618512084, 6.0, 1.8, 2.4),
42035: 'Apple', 42036: 'iPhone 11 Pro Max back triple camera 1.54mm f/2.4', 306: '2019:12:26 16:51:27', 42080: 2,
34665: 212, 37500: b"Apple
iOS\x00\x00\x01MM\x00\x15\x00\x01\x00\t\x00\x00\x00\x01\x00\x00\x00\x0b\x00\x02\x00\x07\x00\x00\x02.\x00\x00\x01\x10\x
\x00\x00\x19\x00\t\x00\x00\x00\x01\x00\x00\x00\x02\x00\x1a\x00\x02\x00\x00\x00\x06\x00\x00\x03\xbe\x00\x1f\x00\t\x00\x00\
\x00\x02\x00\x00\x00%\x00\x00\x03\xc4\x00!\x00\n\x00\x00\x00\x01\x00\x00\x03\xea\x00%\x00\t\x00\x00\x00\x01\x00\x00B\x
\x00'\x001\x00,\x00(\x00=\x00y\x00\x82\x00\xa4\x00\xbd\x00\x8a\x00o\x00d\x00W\x00I\x00=\x00\x1d\x00 \x00
\x00'\x00*\x00.\x00?
\x00O\x00O\x00X\x00O\x00A\x007\x001\x00+\x00&\x00\x16\x00\x17\x00\x14\x00\x16\x00\x1b\x00\x19\x00\x1e\x00%\x00$\x0
\x002\x00*\x00\x1f\x00\x1c\x00\x1d\x00\x19\x00\r\x00\x0c\x00\x0c\x00\x0c\x00\x0e\x00\x12\x00\x14\x00\x15\x00\x13\x00\x10
/8=\x00\x00\x00\x00\x00\x00\x01\x01\x00\x00\x00\x00\x00\x00\x00\t\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
\xff\xff\x18A\x00\x00\xeap\xff\xff\xb8r\x00\x12\xb1\x8b\x00\x00\x1d\x0e\x00\x03\x02\xa3q825s\x008F2ECC1A-89D1-
484D-8D52-869390021ED8\x00\x00\x00\x01*A\x00\x00\xc4u\x00\x03~\xe2\x00\x00\x1b\x9154414414-88C0-4B26-
B0D7-3F4B7357C5ED\x00\x00"}

In [21]:

```
from PIL.ExifTags import TAGS

def get_labeled_exif(exif):
labeled = {}
for (key,val) in exif.items():
labeled[TAGS.get(key)] = val

return labeled

exif = get_exif(r'C:\Users\rj100\OneDrive\Pictures\tablerock.jpeg')
labeled = get_labeled_exif(exif)
print(labeled)
```

{'ExifVersion': b'0231', 'ComponentsConfiguration': b'\x01\x02\x03\x00', 'ShutterSpeedValue':
9.5264271069386, 'DateTimeOriginal': '2019:12:26 16:51:27', 'DateTimeDigitized': '2019:12:26 16:51:27',
'ApertureValue': 2.5260688112781806, 'BrightnessValue': 9.155635150385162, 'ExposureBiasValue': 0.0,

'MeteringMode': 5, 'Flash': 16, 'FocalLength': 1.54, 'ColorSpace': 65535, 'ExifImageWidth': 1024,
'DigitalZoomRatio': 1.0223123732251522, 'FocalLengthIn35mmFilm': 14, 'SceneCaptureType': 0, None: 2,
'SubsecTimeOriginal': '694', 'SubjectLocation': (2015, 1510, 2323, 1392), 'SubsecTimeDigitized': '694',
'ExifImageHeight': 768, 'SensingMethod': 2, 'Make': 'Apple', 'Model': 'iPhone 11 Pro Max',
'ExposureTime': 0.0013568521031207597, 'Orientation': 1, 'YCbCrPositioning': 1, 'FNumber': 2.4,
'SceneType': b'\x01', 'XResolution': 72.0, 'YResolution': 72.0, 'ExposureProgram': 2, 'GPSInfo': {1: 'N', 2: (36.0, 34.0,
35.52), 3: 'W', 4: (93.0, 19.0, 12.0), 5: b'\x00', 6: 282.0852412821416, 12: 'K', 13: 2.3566701406982804, 16: 'T', 17:
21.746246339791913, 23: 'T', 24: 21.746246339791913, 31:
18.766362291485997}, 'ISOSpeedRatings': 20, 'ResolutionUnit': 2, 'ExposureMode': 0, 'FlashPixVersion': b'0100',
'WhiteBalance': 0, 'Software': '13.3', 'LensSpecification': (1.5399999618512084, 6.0, 1.8, 2.4), 'LensMake': 'Apple',
'LensModel': 'iPhone 11 Pro Max back triple camera 1.54mm f/2.4', 'DateTime':
'2019:12:26 16:51:27', 'ExifOffset': 212, 'MakerNote': b"Apple
iOS\x00\x00\x01MM\x00\x15\x00\x01\x00\t\x00\x00\x00\x01\x00\x00\x00\x0b\x00\x02\x00\x07\x00\x00\x02.\x00\x00\x01\x10\x
\x00\x00\x19\x00\t\x00\x00\x00\x01\x00\x00\x00\x02\x00\x1a\x00\x02\x00\x00\x00\x06\x00\x00\x03\xbe\x00\x1f\x00\t\x00\x00\
\x00\x02\x00\x00\x00%\x00\x00\x03\xc4\x00!\x00\n\x00\x00\x00\x01\x00\x00\x03\xea\x00%\x00\t\x00\x00\x00\x01\x00\x00B\x
\x00'\x001\x00,\x00(\x00=\x00y\x00\x82\x00\xa4\x00\xbd\x00\x8a\x00o\x00d\x00W\x00I\x00=\x00\x1d\x00 \x00
\x00'\x00*\x00.\x00?
\x00O\x00O\x00X\x00O\x00A\x007\x001\x00+\x00&\x00\x16\x00\x17\x00\x14\x00\x16\x00\x1b\x00\x19\x00\x1e\x00%\x00$\x0
\x002\x00*\x00\x1f\x00\x1c\x00\x1d\x00\x19\x00\r\x00\x0c\x00\x0c\x00\x0c\x00\x0e\x00\x12\x00\x14\x00\x15\x00\x13\x00\x10
/8=\x00\x00\x00\x00\x00\x00\x00\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00\t\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
\xff\xff\x18A\x00\x00\xeap\xff\xff\xb8r\x00\x12\xb1\x8b\x00\x00\x1d\x0e\x00\x03\x02\xa3q825s\x008F2ECC1A-89D1-
484D-8D52-869390021ED8\x00\x00\x00\x01*A\x00\x00\xc4u\x00\x03~\xe2\x00\x00\x1b\x9154414414-88C0-4B26-
B0D7-3F4B7357C5ED\x00\x00"}

In [22]:

```
from PIL.ExifTags import GPSTAGS

def get_geotagging(exif):
if not exif:
raise ValueError("No EXIF metadata found")

geotagging = {}
for (idx, tag) in TAGS.items():
if tag == 'GPSInfo':
if idx not in exif:
raise ValueError("No EXIF geotagging found")

for (key, val) in GPSTAGS.items():
if key in exif[idx]:
geotagging[val] = exif[idx][key]

return geotagging

exif = get_exif(r'C:\Users\rj100\OneDrive\Pictures\tablerock.jpeg')
geotags = get_geotagging(exif)
print(geotags)
```

{'GPSLatitudeRef': 'N', 'GPSLatitude': (36.0, 34.0, 35.52), 'GPSLongitudeRef': 'W', 'GPSLongitude': (93.0, 19.0, 12.0),
'GPSAltitudeRef': b'\x00', 'GPSAltitude': 282.0852412821416, 'GPSSpeedRef': 'K', 'GPSSpeed':
2.3566701406982804, 'GPSImgDirectionRef': 'T', 'GPSImgDirection': 21.746246339791913, 'GPSDestBearingRef':
'T', 'GPSDestBearing': 21.746246339791913, 'GPSHPositioningError':
18.766362291485997}

In [24]:

```
def get_decimal_from_dms(dms, ref):

degrees = dms[0]
```

```
minutes = dms[1] / 60.0
seconds = dms[2] / 3600.0

if ref in ['S', 'W']:
degrees = -degrees
minutes = -minutes
seconds = -seconds

return round(degrees + minutes + seconds, 5)

def get_coordinates(geotags):
lat = get_decimal_from_dms(geotags['GPSLatitude'], geotags['GPSLatitudeRef'])

lon = get_decimal_from_dms(geotags['GPSLongitude'], geotags['GPSLongitudeRef'])

return (lat,lon)

exif = get_exif(r'C:\Users\rj100\OneDrive\Pictures\tablerock.jpeg')
geotags = get_geotagging(exif)
print(get_coordinates(geotags))
```

(36.57653, -93.32)

CHecking

ln [27]:

```
from PIL import Image
from PIL.ExifTags import TAGS, GPSTAGS

def get_exif_data(image):
"""Returns a dictionary from the exif data of an PIL Image item. Also converts the GPS Tags""" exif_data = {}
info = image._getexif()
if info:
for tag, value in info.items():
decoded = TAGS.get(tag, tag)
if decoded == "GPSInfo":
gps_data = {}
for t in value:
sub_decoded = GPSTAGS.get(t, t)
gps_data[sub_decoded] = value[t]

exif_data[decoded] = gps_data
else:
exif_data[decoded] = value

return exif_data

def _get_if_exist(data, key):
if key in data:
return data[key]

return None

def _convert_to_degress(value):
"""Helper function to convert the GPS coordinates stored in the EXIF to degress in float format""" d0 = value[0]

d = float(d0)

m1 = value[1]
m = float(m1)

s0 = value[2]
s = float(s0)

return d + (m / 60.0) + (s / 3600.0)

def get_lat_lon(exif_data):
"""Returns the latitude and longitude, if available, from the provided exif_data (obtained through get_exif_data
above)""" lat = None
lon = None
```

```
if "GPSInfo" in exif_data:
gps_info = exif_data["GPSInfo"]

gps_latitude = _get_if_exist(gps_info, "GPSLatitude")
gps_latitude_ref = _get_if_exist(gps_info, 'GPSLatitudeRef')
gps_longitude = _get_if_exist(gps_info, 'GPSLongitude')
gps_longitude_ref = _get_if_exist(gps_info, 'GPSLongitudeRef')

if gps_latitude and gps_latitude_ref and gps_longitude and gps_longitude_ref:
lat = _convert_to_degress(gps_latitude)
if gps_latitude_ref != "N":
lat = 0 - lat

lon = _convert_to_degress(gps_longitude)
if gps_longitude_ref != "E":
lon = 0 - lon

return lat, lon

#################
# Example ######
#################
if __name__ == "__main__":
image = Image.open(r'C:\Users\rj100\OneDrive\Pictures\tablerock.jpeg')
exif_data = get_exif_data(image)
print(get_lat_lon(exif_data))
```

(36.57653333333334, -93.32)

Removing Exif From Files

```
In [34]:

image = Image.open(r'C:\Users\rj100\OneDrive\Pictures\tablerock.jpeg')

# next 3 lines strip exif
data = list(image.getdata())
image_without_exif = Image.new(image.mode, image.size)
image_without_exif.putdata(data)

image_without_exif.save('image_file_without_exif.jpeg')

In [43]:

def make_thumbnail(filename):
img = Image.open(filename)

(width, height) = img.size
if width > height:
ratio = 50.0 / width
else:
ratio = 50.0 / height

img.thumbnail((round(width * ratio), round(height * ratio)), Image.LANCZOS)
img.save(filename)

In [44]:

thumbnail = make_thumbnail(r'C:\Users\rj100\OneDrive\Pictures\tablerock.jpeg')
```

```
In [
]:
```

```
In [1]:
from mpl_toolkits.basemap import  Basemap
```

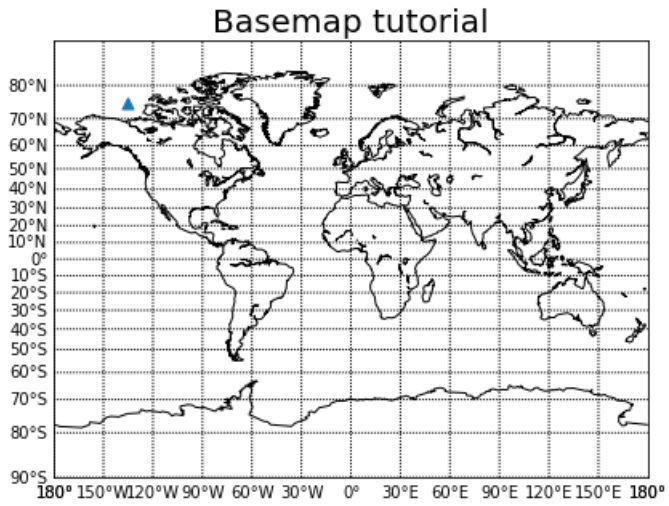
In [21]:

```
import matplotlib.pyplot as plt
import numpy as np

fig = plt.figure(figsize=(12,5))

m = Basemap(projection='mill',
            llcrnrlat=-90,
            urcrnrlat = 90,
            llcrnrlon = -180,
            urcrnrlon = 180,
            resolution = 'c')
m.drawcoastlines()

m.drawparallels(np.arange(-90,90,10), labels=[True,False,False,False])
m.drawmeridians(np.arange(-180,180,30), labels = [0,0,0,1])
m.scatter(-135,75, latlon = True, s=50, marker='^')
m.scatter(-36,93,latlon=True,s=60000,marker='^')
plt.title('Basemap tutorial', fontsize=20)
plt.show()
```



In [ ]: