```
# Created 01/11/2021 by Nick Cheatwood for CAC 350
#  - Modified 02/14/2021 by Nick Cheatwood

# Import libraries/data
import numpy as np # for working with large sets of data
import pandas as pd # for working with files
import reverse_geocode # allows us to get location info based on
coordinates
import datetime # allows us to get current timestamp in ISO format (as
used in the .csv file)
import matplotlib.pyplot as plt # allows us to plot histograms
```

```
# Name and import the data
earthquake_data = pd.read_csv('earthquakes.csv')

# parse out magnitude
magnitude = np.array(earthquake_data['mag'])
```

```
# TO-DO: Create a report on all earthquakes that occurred globally in the
past 30 days

# No earthquakes occurred in the past 30 days (01/14/2021-02/14/2021), so
I we test 1 year and 30 days ago ( 01/14/2020 )

# Get the current timestamp (obsolete since none in 30 days)
current = datetime.datetime.utcnow().replace().isoformat()[:-3] # converts
to string, but will fix later
# Log the current timestamp
print('current:',current)

# Create a data frame.  This acts as sort of a filter.
df = pd.DataFrame(earthquake_data)
```

```
# time is in form 'yyyy-mm-ddThh:mm:ss.mmmZ'
# Find all earthquakes since 01/14/2020, parse out magnitude, type,
longitude, latitude, and time
past30Data = df.loc[df['time'] > '2020-01-14T00:00:00.000Z',
['mag','magType','longitude','latitude', "time"]].to_numpy()
# [ 0: mag, 1: magType, 2: longitude, 3: latitude, 4: time]

# Turn 2D array into array with dictionary in each index
past30DataArr = []
c = 0

for row in past30Data:
    # for each sub-array in the 2D array
    col_data = {
            "mag": row[0],
            "magType": row[1],
            "lon": row[2],
            "lat": row[3],
            "time": row[4]
    }

    # append to array (now have array with dictionary in each index)
    past30DataArr.append(col_data)
```

```
current: 2021-02-14T06:56:44.827
```

```
# Compute basic stats on the magnitude: max, min, median, mean, and 25th
and 75th percentiles
```

```
# Extract all magnitudes from the filtered data
# Init a new array
mag_data_arr = []

for i in past30DataArr:
    mag_data_arr.append(i.get("mag"))

magMax = np.nanmax(mag_data_arr)
magMin = np.nanmin(mag_data_arr)
magMed = np.nanmedian(mag_data_arr)
magMean = np.nanmean(mag_data_arr)
mag25 = np.nanpercentile(mag_data_arr,25)
mag75 = np.nanpercentile(mag_data_arr, 75)

# Convert to a dictionary for better readability
basic_stats = {
    "max": magMax,
    "min": magMin,
    "median": magMed,
    "mean": magMean,
    "25Perc": mag25,
    "75Perc": mag75
}

#  Log the results
print("Basic stats:",basic_stats)
```

```
Basic stats: {'max': 5.9, 'min': 2.45, 'median': 2.9, 'mean':
3.259553752535497, '25Perc': 2.64, '75Perc': 3.7}
```

```
# Average magnitude where type is ml

def getAverageMagnitudeForType(type, arr):
    # Gets the average magnitude in each array index based on magType
    typeValues = []
    avg = 0
    for data in arr:
        # for each dict in the data arr
        if ( data.get("magType") == type ):
            typeValues.append(data.get("mag"))

    # the average is the sum divided by the count
    avg = sum(typeValues) / len(typeValues)
    return avg

# Log the results
avgTypeML = getAverageMagnitudeForType("ml", past30DataArr)
print("Average where the type is ml:", avgTypeML)
```

```
Average where the type is ml: 2.8317977528089893
```

```
# Date and location of the largest magnitude

def getDataOfLargestMagnitude(arr):
    # given an array of dictionaries, returns the data (in array format
since there may be multiple) of the largest magnitude
    data = []
    largestMag = 0

    # get the largest mag
```

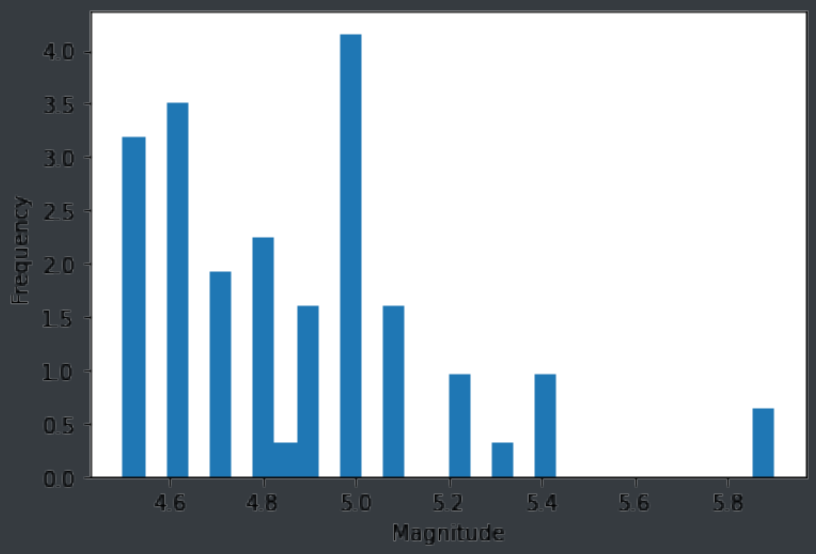
```
    for obj in arr:
        # loop through dicts in array
        if ( obj.get("mag") >= largestMag ):
            # loops throw data array and determines largest magnitude
value across all indexes
            largestMag = obj.get("mag")

    # return data for largest mag
    for obj in arr:
        if ( obj.get("mag") == largestMag ):
          # If the current dictionary contains the largest magnitude, log
its respective data
            this_data = {
                "dateTime": obj.get("time"),
                "lat": obj.get("lat"),
                "lon": obj.get("lon"),
                "mag": obj.get("mag")
            }
            data.append(this_data)

    return data   # Pass out the data

# Now we have the row data for all days containing the largest magnitude
(5.9)
largestData = getDataOfLargestMagnitude(past30DataArr)

# print("Largest Magnitude Date and Coords:", largestData)

def appendLocationData(arr):
    # Given an array of dictionaries, appends to each dictionary location
data
    newArr = arr.copy()
```

```
    for i in newArr: # for each dictionary in the array:
        # get longitude and latitude as a tuple
        loc_coords = ( i.get("lat") , i.get("lon") ), # Requires ',' on
the end ???
        # print("Coords:", loc_coords)
        # convert to location data
        loc_data = reverse_geocode.search(loc_coords)
        # loc_data = ''
        # append to current dictionary
        i.update({'location': loc_data})

    return newArr

#rgTestCoords = ( -37.81, 144.96 )

#print(type(rgTestCoords))

# print(reverse_geocode.search(rgTestCoords))

# Now we have the row data for all days containing the largest magnitude
(5.9), but now with better location data
largestDataWithLocation = appendLocationData(largestData)
print("With Location Data:", largestDataWithLocation)

# Can we get only 'country code'?
# largestDataWithLocation[0].get('location')[0].get('country_code')
```

```
With Location Data: [{'dateTime': '2020-04-10T16:44:55.531Z', 'lat':
20.4267, 'lon': 122.1198, 'mag': 5.9, 'location': [{'country_code': 'PH',
'city': 'Basco', 'country': 'Philippines'}]}, {'dateTime': '2020-04-
05T18:37:10.487Z', 'lat': 1.4007, 'lon': 126.4427, 'mag': 5.9, 'location':
[{'country_code': 'ID', 'city': 'Susupu', 'country': 'Indonesia'}]}]
```

```
# Plot significant earthquakes (mag 4.5+) in a histogram

def getSignificantEarthquakes(arr):
    # init array
    sigMags = []
    for i in arr: # for each dict in the array
        if ( i.get('mag') >= 4.5): # test for significant earthquakes
            sigMags.append(i.get('mag'))

    return sigMags

# Now we have an array of significant earthquake magnitudes
significantRecords = getSignificantEarthquakes(past30DataArr)
# print(significantRecords)

# Plot the histogram
x = significantRecords # name the data according to the axis
# Not sure what x- and y-axis should be, so make generic
plt.hist(x, density=True, bins=30)
plt.ylabel('Frequency')
plt.xlabel('Magnitude')
```

```
Text(0.5, 0, 'Magnitude')
```




```
# Plot a histogram of only earthquakes that occurred in the US

# Read through all data and append location data
data_with_location = appendLocationData(past30DataArr)

# print(data_with_location)

# Read through each an add only those in the US to an array
def getDataForCountry(country_code, arr):
    # Init a fresh array
    magArr = []
    for i in arr: # for each dict in the array:
        code = i.get('location')[0].get('country_code') # test against the
provided code
        if ( code == country_code ):
            magArr.append(i.get('mag'))


    return magArr
```

```
# We now have an array of magnitudes for the U.S.
us_data = getDataForCountry('US', data_with_location)
# print(us_data)

x = us_data # name the data according to the axis
# Again, unsure of what axis should be, so make generic
plt.hist(x, density=True, bins=30)
plt.ylabel('Frequency')
plt.xlabel('Magnitude')
```

```
Text(0.5, 0, 'Magnitude')
```