```
import pandas as pd
import scipy.signal as scisig
import os
import numpy as np


def get_user_input(prompt):
    try:
        return raw_input(prompt)
    except NameError:
        return input(prompt)


def getInputLoadFile():
    '''Asks user for type of file and file path. Loads corresponding data.

    OUTPUT:
        data:   DataFrame, index is a list of timestamps at 8Hz, columns include
                AccelZ, AccelY, AccelX, Temp, EDA, filtered_eda
    '''
    print("Please enter information about your EDA file... ")
    dataType = get_user_input("\tData Type (e4, q, shimmer, or misc): ")
    if dataType=='q':
        filepath = get_user_input("\tFile path: ")
        filepath_confirm = filepath
        data = loadData_Qsensor(filepath)
    elif dataType=='e4':
        filepath = get_user_input("\tPath to E4 directory: ")
        filepath_confirm = os.path.join(filepath,"EDA.csv")
        data = loadData_E4(filepath)
    elif dataType=='shimmer':
        filepath = get_user_input("\tFile path: ")
        filepath_confirm = filepath
        data = loadData_shimmer(filepath)
    elif dataType=="misc":
        filepath = get_user_input("\tFile path: ")
        filepath_confirm = filepath
        data = loadData_misc(filepath)
    else:
        print("Error: not a valid file choice")

    return data, filepath_confirm

def getOutputPath():
    print("")
    print("Where would you like to save the computed output file?")
    outfile = get_user_input('\tFile name: ')
    outputPath = get_user_input('\tFile directory (./ for this directory): ')
    fullOutputPath = os.path.join(outputPath,outfile)
    if fullOutputPath[-4:] != '.csv':
        fullOutputPath = fullOutputPath+'.csv'
    return fullOutputPath

def loadData_Qsensor(filepath):
    '''
    This function loads the Q sensor data, uses a lowpass butterworth filter on the EDA signal
    Note: currently assumes sampling rate of 8hz, 16hz, 32hz; if sampling rate is 16hz or 32hz the signal is downsampled
    INPUT:
        filepath:       string, path to input file

    OUTPUT:
        data:           DataFrame, index is a list of timestamps at 8Hz, columns include AccelZ, AccelY, AccelX, Temp, EDA, filtered_eda
    '''
    # Get header info
    try:
        header_info = pd.io.parsers.read_csv(filepath, nrows=5)
    except IOError:
        print("Error!! Couldn't load file, make sure the filepath is correct and you are using a csv from the q sensor software\n\n")
        return

    # Get sample rate
    sampleRate = int((header_info.iloc[3,0]).split(":")[1].strip())

    # Get the raw data
    data = pd.io.parsers.read_csv(filepath, skiprows=7)
    data = data.reset_index()

    # Reset the index to be a time and reset the column headers
    data.columns = ['AccelZ','AccelY','AccelX','Battery','Temp','EDA']

    # Get Start Time
    startTime = pd.to_datetime(header_info.iloc[4,0][12:-10])
    # Make sure data has a sample rate of 8Hz
    data = interpolateDataTo8Hz(data,sampleRate,startTime)

    # Remove Battery Column
    data = data[['AccelZ','AccelY','AccelX','Temp','EDA']]

    # Get the filtered data using a low-pass butterworth filter (cutoff:1hz, fs:8hz, order:6)
    data['filtered_eda'] =  butter_lowpass_filter(data['EDA'], 1.0, 8, 6)

    return data

def _loadSingleFile_E4(filepath,list_of_columns, expected_sample_rate,freq):
    # Load data
    data = pd.read_csv(filepath)

    # Get the startTime and sample rate
    startTime = pd.to_datetime(float(data.columns.values[0]),unit="s")
    sampleRate = float(data.iloc[0][0])
    data = data[data.index!=0]
    data.index = data.index-1

    # Reset the data frame assuming expected_sample_rate
    data.columns = list_of_columns
    if sampleRate != expected_sample_rate:
        print('ERROR, NOT SAMPLED AT {0}HZ. PROBLEMS WILL OCCUR\n'.format(expected_sample_rate))

    # Make sure data has a sample rate of 8Hz
    data = interpolateDataTo8Hz(data,sampleRate,startTime)
    return data


def loadData_E4(filepath):
    # Load EDA data
    eda_data = _loadSingleFile_E4(os.path.join(filepath,'EDA.csv'),["EDA"],4,"250L")
    # Get the filtered data using a low-pass butterworth filter (cutoff:1hz, fs:8hz, order:6)
    eda_data['filtered_eda'] =  butter_lowpass_filter(eda_data['EDA'], 1.0, 8, 6)

    # Load ACC data
    acc_data = _loadSingleFile_E4(os.path.join(filepath,'ACC.csv'),["AccelX","AccelY","AccelZ"],32,"31250U")
    # Scale the accelometer to +-2g
    acc_data[["AccelX","AccelY","AccelZ"]] = acc_data[["AccelX","AccelY","AccelZ"]]/64.0

    # Load Temperature data
    temperature_data = _loadSingleFile_E4(os.path.join(filepath,'TEMP.csv'),["Temp"],4,"250L")

    data = eda_data.join(acc_data, how='outer')
    data = data.join(temperature_data, how='outer')

    # E4 sometimes records different length files - adjust as necessary
    min_length = min(len(acc_data), len(eda_data), len(temperature_data))

    return data[:min_length]

def loadData_shimmer(filepath):
    data = pd.read_csv(filepath, sep='\t', skiprows=(0,1))
```

```
orig_cols = data.columns
rename_cols = {}

for search, new_col in [['Timestamp','Timestamp'],
                       ['Accel_LN_X', 'AccelX'], ['Accel_LN_Y', 'AccelY'], ['Accel_LN_Z', 'AccelZ'],
                       ['Skin_Conductance', 'EDA']]:
    orig = [c for c in orig_cols if search in c]
    if len(orig) == 0:
        continue
    rename_cols[orig[0]] = new_col

data.rename(columns=rename_cols, inplace=True)

# TODO: Assuming no temperature is recorded
data['Temp'] = 0

# Drop the units row and unnecessary columns
data = data[data['Timestamp'] != 'ms']
data.index = pd.to_datetime(data['Timestamp'], unit='ms')
data = data[['AccelZ', 'AccelY', 'AccelX', 'Temp', 'EDA']]

for c in ['AccelZ', 'AccelY', 'AccelX', 'Temp', 'EDA']:
    data[c] = pd.to_numeric(data[c])

# Convert to 8Hz
data = data.resample("125L").mean()
data.interpolate(inplace=True)

# Get the filtered data using a low-pass butterworth filter (cutoff:1hz, fs:8hz, order:6)
```

```
    data['filtered_eda'] = butter_lowpass_filter(data['EDA'], 1.0, 8, 6)

    return data


def loadData_getColNames(data_columns):
    print("Here are the data columns of your file: ")
    print(data_columns)

    # Find the column names for each of the 5 data streams
    colnames = ['EDA data','Temperature data','Acceleration X','Acceleration Y','Acceleration Z']
    new_colnames = ['','','','','']

    for i in range(len(new_colnames)):
        new_colnames[i] = get_user_input("Column name that contains "+colnames[i]+": ")
        while (new_colnames[i] not in data_columns):
            print("Column not found. Please try again")
            print("Here are the data columns of your file: ")
            print(data_columns)

            new_colnames[i] = get_user_input("Column name that contains "+colnames[i]+": ")

    # Get user input on sample rate
    sampleRate = get_user_input("Enter sample rate (must be an integer power of 2): ")
    while (sampleRate.isdigit()==False) or (np.log(int(sampleRate))/np.log(2) !=
np.floor(np.log(int(sampleRate))/np.log(2))):
        print("Not an integer power of two")
        sampleRate = get_user_input("Enter sample rate (must be a integer power of 2): ")
    sampleRate = int(sampleRate)
    # Get user input on start time
    startTime = pd.to_datetime(get_user_input("Enter a start time (format: YYYY-MM-DD HH:MM:SS): "))
    while type(startTime)==str:
        print("Not a valid date/time")
        startTime = pd.to_datetime(get_user_input("Enter a start time (format: YYYY-MM-DD HH:MM:SS):
"))


    return sampleRate, startTime, new_colnames



def loadData_misc(filepath):
    # Load data
    data = pd.read_csv(filepath)

    # Get the correct colnames
    sampleRate, startTime, new_colnames = loadData_getColNames(data.columns.values)

    data.rename(columns=dict(zip(new_colnames,['EDA','Temp','AccelX','AccelY','AccelZ'])), inplace=True)
    data = data[['AccelZ','AccelY','AccelX','Temp','EDA']]

    # Make sure data has a sample rate of 8Hz
    data = interpolateDataTo8Hz(data,sampleRate,startTime)

    # Get the filtered data using a low-pass butterworth filter (cutoff:1hz, fs:8hz, order:6)
    data['filtered_eda'] =  butter_lowpass_filter(data['EDA'], 1.0, 8, 6)

    return data
```

```
def interpolateDataTo8Hz(data,sample_rate,startTime):
    if sample_rate<8:
        # Upsample by linear interpolation
        if sample_rate==2:
            data.index = pd.date_range(start=startTime, periods=len(data), freq='500L')
        elif sample_rate==4:
            data.index = pd.date_range(start=startTime, periods=len(data), freq='250L')
        data = data.resample("125L").mean()
    else:
        if sample_rate>8:
            # Downsample
            idx_range = list(range(0,len(data))) # TODO: double check this one
            data = data.iloc[idx_range[0::int(int(sample_rate)/8)]]
        # Set the index to be 8Hz
        data.index = pd.date_range(start=startTime, periods=len(data), freq='125L')

    # Interpolate all empty values
    data = interpolateEmptyValues(data)
    return data

def interpolateEmptyValues(data):
    cols = data.columns.values
    for c in cols:
        data.loc[:, c] = data[c].interpolate()

    return data

def butter_lowpass(cutoff, fs, order=5):
    # Filtering Helper functions
    nyq = 0.5 * fs
    normal_cutoff = cutoff / nyq
    b, a = scisig.butter(order, normal_cutoff, btype='low', analog=False)
    return b, a


def butter_lowpass_filter(data, cutoff, fs, order=5):
    # Filtering Helper functions
    b, a = butter_lowpass(cutoff, fs, order=order)
    y = scisig.lfilter(b, a, data)
    return y
```