The Viewer is a python based tool for efficient analysis and visualization of Electrophsyiological signals. The viewer can be used as imported package inside the user's code where the user can call individual functions or from the GUI where the user can choose parameters for the analysis and visualization from an easy to use interface.

The viewer consists of two parts: the modules, and the GUI.

Modules:

- Modules contain all the analysis and visualiation functions. The modules functions can be imported inside the user's code and used directly in the code, or used through the GUI of the viewer.

1. **Load_data module**

    - Loads data from different formats for viewing and analysis. Supported formats are text files, numpy files, and csv files.

    - Functions:
        - Build_dataset_dir: extracts dataset from zipped file into a directory and returns the dataset directory.
        - Load_record: automatically detects the type of the file from the supported formats, loads the file, and returns its data as numpy array.
        - Load_dataset: takes the dataset directory and loads the complete dataset into a numpy matrix. The dataset files can be any type of the supported ones or a mix of them.

2. **Denoising_data module**

    - Denoises a dataset's records using the index-based discrete cosine transform (ibdct) filter or using the butterworth filter.

3. Segment_data module

- Using functions in this module the user can separate a record into separate channels, and divide data into segments of window size and step size.
- Functions:
    - Separate_data_chs: separates a record's channels into separate arrays.
    - Make_views: segments record into segments using custom window size and step size

4. Filters module

- Module contains filters that can be used by users to filter a signal or by the denoising function to remove noise from the dataset. Module also contains a function to create custom IIR and FIR filters which returns the filter coefficients and frequency response.

- Functions:
    - Ibdct: applies index-based discrete cosine transform filter on the input signal.
    - Butterworth_filter: applies sos butterworth_filter on the input signal.
    - Create_filter: allows user to create custom filters. The user can:
        - Create IIR of FIR filter
        - Create analog or digital filter
        - Create lowpass, highpass, bandpass, and bandstop filter
        - Choose output type as 'ba' or 'sos' or 'zpk'
        - Choose window type
        - Choose filter type (butter, ellip, chebyl,etc..)
    - Get_freq_response: calculates and returns the frequency response of any type of filter created from the create_filter function.

5. **Preprocessing module**

- From this module the user can:
    - resample a signal
    - get summary statistics of a signal
    - get summary statistics of a dataset

6. **Visualization module**

- Generates visualizations for multiple analysis techniques, and for filters designed through the filters module.

- Functions:
    - Plot_signal: plots a time-amplitude graph of the signal.
    - Plot_signals: uses plot_signal function to plot all signals in a dataset or a subset of a dataset.
    - Plot_random_samples: uses plot_signal to plot random samples from a dataset.
    - Plot_psd: plots PSD values of the signal.
    - Plot_spectrogram: plots spectrogram of the signal.
    - Plot_sscwt_cwt: plots either sscwt, cwt, or both sscwt and cwt of signals.
    - Plot_mag_response: calculates and plots magnitude response of a filter.

    - Plot_phase_response: calculates and plots phase response of a filter.
    - Plot_impulse_response: calculates and plots impulse response of a filter.
    - Plot_step_response: calculates and plots step response of a filter.
    - Plot_filter_prop: uses the magnitude, phase, impulse, and step response signals and plots all the responses of the filter.

        - Correlation_map: plots correlation map of input dataframe.

7. **Generate_signal module**

    - Generates user defined sine,cos,or exponential synthetic signals.

    - Features:
        - Define sine,cos,or exponential signal.
        - Apply Amplitude modulation. To apply set AM = True and set fa,ac, and ka to numerical values.
        - Apply Frequency modulation. To apply set FM = True and set fc and kf to numeric values.
        - Apply signal self-imposing. To apply set s_imp to true.
        - Add noise to signal. To apply set SNR to numeric value.

GUI:

- The GUI is built with pyqt5. The GUI uses the functions in the viewer module to perform analysis and visualization on signals, and to construct and test filters.

- The GUI allows easy analysis and visualization of data without needing to write code. Through the GUI the user can load signals and datasets, plot their properties, design filters and test them on synthetic signals, and apply the filters on the original signals.

- The GUI consists of two tabs:

1. **Signals Tab:**
    - Allows user to load a signal or a complete dataset from the dataset's directory through the file menu. In case of loading the dataset, all the records inside the dataset are displayed in a side menu and the user can choose a signal to plot by selecting the signal from the menu.
    - The user can visualize the signal properties by selecting a property to plot from the plot menu.
    - The user can also check the apply filter on signal checkbox to apply the filter designed in the filters tab on the loaded signal.

2. **Filters Tab:**
    - User can design a filter by setting the properties from the properties menu at the top right corner of the tab. When a filter is defined, its magnitude, phase, impulse, and step responses' graphs are generated automatically. The result filter is automatically applied on the synthetic signal generated in the same tab.
    - User can generate a synthetic signal to test the designed filter on. The user can choose the signal properties from the menu at the top left corner of the tab. The signal and the filtered signal graphs are automatically generated whenever the signal parameter changes.