Sonification-tool

Release 1.0.0

Szymon Duda

Jan 30, 2026

CONTENTS:

CHAPTER

ONE

SRC

1.1 src package

1.1.1 src.askers module

class src.askers.**Askers**

Bases: `object`

Container class for asker functions.

Askers are static methods that prompt user to input a value. In most cases they can come in form of a menu or a piece of code inquiring a value to assign to a variable.

settings_path

Path to a settings file.

Type

Path

notes_path

Path to a notes file.

Type

Path

static ask_action() → Literal['process_data', 'sonify', 'show_chart', 'show_histogram', 'settings', 'original_data', 'change_file', 'exit']

Get main menu action.

Returns

Actions in forms of pre-defined strings.

static ask_data_settings() → Literal['auto_normalization_at_load', 'auto_threshold_at_load', 'show_thold_chart', 'change_cutting_setting_paa', 'change_cutting_setting_dwelltimes', 'change_segmenting_setting_paa', 'change_segmenting_setting_dwelltimes', 'change_imfs_from'] | None

Get data settings menu action.

Returns
Actions in forms of pre-defined strings or None if returning.

`static ask_downsampling`(*is_initial: bool = False*) → int | None

Get downsampling n value from user.

Asks user to choose n value for downsampling that ranges from 1 to 10. If user doesn't want to downsample, returns None instead.

Parameters
is_initial (*bool*) – Determines if menu is for initial segmentation or not.

Returns
n value for downsampling in (1, 10) or None if not chosen.

`static ask_imf_num`(*lowest: int*, *highest: int*) → int | None

Get number of IMF to pick.

Parameters

- **lowest** (*int*) – Lowest possible IMF to pick.
- **highest** (*int*) – Highest possible IMF to pick.

Returns
Number of IMF in (min_num, max_num) or None if not chosen.

`static ask_lowest_note_anal`(*current_lowest_note_name: str*, *highest_lowest_note_possible: str*, *notes: list[str]*) → str | None

Get lowest note for analog sonification from user.

Calculates a range of notes that are possible to choose from, then prompts user to choose one. Checks if input is within range.

Parameters

- **current_lowest_note_name** (*str*) – Current lowest note name.
- **highest_lowest_note_possible** (*str*) – Highest lowest note name possible.
- **notes** (*list[str]*) – All notes available for sonification.

Returns
Lowest note for analog sonification or None if not chosen.

`static ask_new_imfs_from`() → int | None

Get number of IMF to start displaying from.

Returns
Number of IMF in (min_num, max_num) or None if not chosen.

`static ask_note_amount`(*available_notes_count: int*) → int | None

Get amount of notes for analog sonification from user.

Asks user for the amount of notes to be used in analog sonification. Checks if value fits within a specified range.

Parameters
`available_notes_count` (*int*) – Upper note threshold (just in case).

Returns
Amount of notes to be used in analog sonification or None if not chosen.

`static ask_note_binary(`*low_or_high: Literal['low', 'high']*`)` → str | None

Get high or low note for binary sonification.

Asks user to input a low or high note name depending on argument. Value is checked if exists in notes.json.

Parameters
`low_or_high` (*str*) – Low or high note for specific prints.

Returns
Note name of lowest/highest note or None if not chosen.

`static ask_note_duration()` → int | None

Get duration of a note in ms.

Asks user how much miliseconds a single note should last. Value is then checked if it is present within a range of acceptable answers.

Returns
Number of miliseconds that every sample lasts in output audio file or None if not chosen.

`static ask_path_filedialog(`*starting_path: str*`)` → str

Open file dialogbox and get txt/csv file.

Opens a file dialog box that propmpts user to choose a txt or csv file, starting on a starting_path path.

Parameters
`starting_path` (*str*) – Path where dialog box opens.

Returns
Path of the chosen file.

`static ask_process_data(`*is_normalized: str*, *is_threshold: str*, *is_binary: str*`)` → Literal['reverse_order', 'reverse_sign', 'normalization', 'calculate_threshold', 'downsample_data', 'apply_paa', 'convert_to_bin', 'convert_to_dwelltimes', 'convert_to_dwelltimes_reduced', 'appy_emd'] | None

Get processing menu action.

Returns
Actions in forms of pre-defined strings or None if returning.

static **ask_segment_value**(*data_length: int*, *segmenting_style: Literal['count', 'size']*) → int | None

Get segmenting value from user.

Asks user to input a segmenting value. Prints depend on what style of segmenting is chosen. User input is checked to be in a correct range.

Parameters

- **data_length** (*int*) – Length of the data to be segmented.
- **segmenting_style** (*str*) – Type of segmentation to be performed.

Returns

Value of a segment (length or size) or None if not chosen.

static **ask_settings_type**() → Literal['data_settings', 'sonif_settings'] | None

Get settings type menu action.

Returns

Actions in forms of pre-defined strings or None if returning.

static **ask_similarity_threshold**() → float

Get similarity threshold from user.

Asks user for similarity threshold for note cutting during sonification. Checks if value fits within a specified range.

Returns

Similarity threshold for note cutting during sonification.

static **ask_sonif_settings**() → Literal['change_binary_low_note', 'change_binary_high_note', 'change_similarity_threshold'] | None

Get sonification settings menu action.

Returns

Actions in forms of pre-defined strings or None if returning.

static **ask_sonif_type**(*bin_available: bool*, *analog_available: bool*) → Literal['binary', 'analog'] | None

Get sonification type menu action.

Asks user to pick sonification type. Displays messages when it is not available. Picking unavailabe message makes program display the reason why it’s not available.

Parameters

- **bin_available** (*bool*) – Informs if binary sonification is available for choosing.
- **analog_available** (*bool*) – Informs if analog sonification is available for choosing.

Returns

Actions in forms of pre-defined strings or None if returning.

1.1.2 src.chunk module

class `src.chunk.`**`Chunk`**(*in_start: int*, *in_end: int*, *in_data_array: ndarray[float64] | ndarray[bool] | None = None*)

Bases: `object`

Representation of a data segment.

This is a helper class that assists with segmentation of data. It is capable of calculating data mean. Serves to make program more abstract and easier to understand.

`index_start`

Index in the main array where data from Chunk instance begins.

Type

int

`index_end`

Index in the main array where data from Chunk instance ends.

Type

int

`num_of_samples`

Number of samples present in Chunk.

Type

int

`__data_array`

Data present in Chunk.

Type

np.ndarray[np.float64] | np.ndarray[bool] | None

`data_mean`

Mean of all samples present in Chunk.

Type

np.float64 | float | None

`calculate_mean_from_data`() → None

Calculate and set mean from data array.

`del_data_array`() → None

Free up __data_array field space.

`get_data_mean`() → float64

Get data mean.

Returns

Mean of loaded data.

Raises

`Error` – If these's no mean calculated.

input_data_array(*new_array: ndarray[float64] | ndarray[bool]*) → None

Set new data array

Parameters
new_array – An array to become new __data_array field.

Raises
Error – If new array length doesn't match number of samples.

1.1.3 src.datasonif module

class src.datasonif.**DataSonif**

Bases: object

Representation of data to be sonified.

This class holds data array and information regarding it. It's capable of performing all processing and sonification operations on data as well as lesser operations such as showing charts and loading data.

file_path

Path to the original file with data.

Type
Path | None

data_array

Data that will be processed and sonified, loaded from a file.

Type
np.ndarray[np.float64] | np.ndarray[bool] | None

data_sign

Sign of data.

Type
str | None

is_og_order

Informs if data has original order or not.

Type
bool | None

is_og_sign

Informs if data has original sign or not.

Type
bool | None

min_val

Min val from data array.

Type
float | None

`max_val`

Max val from data array.

Type
float | None

`bins_count`

Amount of bins used for creating histogram and calculating data threshold.

Type
int

`threshold`

Data threshold for open/closed states.

Type
float | None

`is_normalized`

Informs if data is currently normalized.

Type
bool

`downsampling_performed`

Stores all values of used to perform downsampling of the data.

Type
list[int]

`is_converted_to_binary`

Informs if data is binary.

Type
bool

`settings_path`

Stores path to settings.json.

Type
Path

`notes_path`

Stores path to notes.json.

Type
Path

`sample_rate`

Output audio sample rate.

Type
int

analog_sonif_loop() → None

Asker loop for analog sonification.

Asker loop for analog sonification (not in Askers class because performs it operations on DataSonif). Prints information regarding sonification and allows changing note length, lowest note, amount of notes as well as analog sonification itself.

analog_sonification(*note_duration_milis: int*, *notes_used: list[str]*, *notes_dict: dict[str, float]*) → None

Perform analog sonification on data array.

Iteratively for every value in data array, decides frequency for sample by assigning it to a correct bin and finding corresponding value in the freqs array. Then creates an extended sine wave in an array, cuts it to match constant duration and adds it to audio array. When audio array is filled, it's sine waves are joined into a flac file and saved in outputs directory.

Parameters

- **note_duration_milis** (*int*) – Duration of note for a single sample.
- **notes_used** (*List[str]*) – Note names used for sonification.
- **notes_dict** – (dict[str, float]): Note to frequency dictionary.

apply_emd() → bool

Apply EMD decomposition on data array and let user choose IMF.

Applies EMD decomposition on data array and displays IMFs possible for choosing. Then asks user to select IMF to replace data array or return. If an IMF is chosen it also updates affected fields accordingly.

Returns

True if EMD has been applied correctly.

apply_paa_aggregation(*segment_value: int*, *segmenting_style: Literal['count', 'size']*) → None

Apply PAA aggregation on data array.

Applies PAA aggregation on data array by cutting it into segments and calculating their mean, from which a new array is created, replacing the old one after it's filled. Amount of segments and their size is based on function input.

Parameters

- **segment_value** (*int*) – Value of segment, either amount of segments or their size.
- **segmenting_style** (*str*) – Style of segmenting, "count" or "size".

binary_sonif_loop() → None

Asker loop for binary sonification.

Asker loop for binary sonification (not in Askers class because performs it operations on DataSonif). Prints information regarding sonification and allows changing note length as well as binary sonification itself.

`binary_sonification`(*note_duration_milis: int*, *low_note_freq: float*, *high_note_freq: float*) → None

Perform binary sonification on data array.

Iteratively for every value in data array, decides frequency for evey sample, creates an extended sine wave in an array, cuts it to match constant duration and adds it to audio array. When audio array is filled, it's sine waves are joined into a flac file and saved in outputs directory.

Parameters

- **note_duration_milis** (*int*) – Duration of note for a single sample.
- **low_note_freq** (*float*) – Frequency of high note.
- **high_note_freq** (*float*) – Frequency of low note.

`calculate_threshold`() → None

Calculate open/close threshold for data array.

Calculates open/close threshold for data array by creating a histogram and finding a midpoint between two peaks. Updates affected fields accordingly.

`convert_data_to_binary`() → None

Convert data array to binary.

Converts samples in data array to binary values by comparing them to state threshold.

`convert_to_dwell_times`(*segment_value: int*, *segmenting_style: str*) → None

Convert data array to dwell times.

`convert_to_dwell_times_REDUCED`(*segment_value: int*, *segmenting_style: str*) → None

Convert data array to reduced dwell times.

`downsample_data`(*n: int*) → None

Downsample data in data array.

Downsamples data array by remplacing it with a new one where every n-th value from original array will be saved.

Parameters

n (*int*) – Value of lines that will remain, where every n-th line will not be removed.

`get_datafile_path`(*filebox_startpath: str*) → bool

Ask and set path of data file.

Asks user for path to data file and sets its path as field.

Parameters
: **`filebox_startpath`** (*str*) – Path where dialog box should open.

Returns
: True if path has been set, else False.

`get_sample_count()` → int

Gets length of data array.

`load_data()` → bool

Load data from file path field.

Resets fields, then asks user if they want to downsample data. After that, attempts to load it from file. If succeeds, updates fields and normalizes data/calculates threshold if settings say to do so.

Returns
: True if data has been loaded, else False.

`normalize_data()` → None

Normalize data array.

Performs min max normalization on data array. Updates affected fields accordingly.

`reset_instance_fields()` → None

Reset most fields of instance.

Resets all instance fields besides file_path. Also doesn't reset class fields.

`reverse_data_order()` → None

Reverses data array.

`reverse_data_sign()` → None

Reverse sign of data.

Reverses data sign and updates affected fields accordingly.

`show_chart()` → None

Plot data array on a chart.

Plots data array on a chart. Can also show threshold.

`show_histogram()` → None

Plot data array histogram on a chart.

Creates a histogram with 200 bins and charts it. Can show threshold.

1.1.4 src.mainloop module

`src.mainloop.`**`mainloop()`** → None

Open main menu and handle user input.

This is a program's strating point and main menu. It begins with prompting user to choose a txt/csv file to load data from. Then, main menu begins where user is presented with various options to choose from. Their options are: - Process data, which opens a menu asking them for specific processing technique and responding accordingly. - Sonify data, which opens a menu to choose correct sonificatin type and handling responses accordingy. - Show chart, which charts loaded data. - Show histogram, which prints data values on a histogram with 200 bins. - Settings, which opens settings loop. - Load original data, which loads data again from the same file. - Change file, which loads data from a different file chosen by user. - Exit, which exits the program.

1.1.5 src.note module

`class src.note.Note`(*freq: float*, *og_sample_amount: int*, *lowest_note_wavelen_samples_roundup: int*)

Bases: `object`

Representation of a frequency (note) in time.

This is a representation of a note that is used for sonification. Class handles generating a sine wave in an array as well as handling tone extension with lowest note wave length and cutting it to math previous sine wave.

`freq`

Frequency.

Type
float

`og_sample_amount`

Original amount of samples.

Type
int

`lowest_note_wavelen_samples_roundup`

Amount of samples needed for calculating full wavelength of the lowest possible note.

Type
int

`time_vector`

Constatnly rising function for calculation sine wave.

Type
np.ndarray[np.float64]

`tone`

Sine wave sample values array.

Type
np.ndarray[np.float64] | None

`last_freq`

Last frequency of calculated tone.

Type
float | None

`curr_sample_amount`
Amount of samples in current tone.

Type
int

`sample_rate`
Sample rate of sound.

Type
int

`similatiry_threshold`
Threshold dictating when two frequencies are similar.

Type
float

`are_freqs_similar`(*freq1: float*, *freq2: float*) → bool
Check if notes are similar.

Checks if notes are similar according to a similarity threshold.

`calculate_time_vector`() → None
Calculate and set time vector of instance.

`calculate_tone`() → None
Calculate and set tone of instance.

Calculates a sine wave of an instance, then sets it and last frequency.

`cut_tone_to_match`(*is_freq_rising: bool*, *prev_note_last_freq: float*) → None
Cut (extended) tone to match the previous sine wave.

Cuts (extended) tone so that it matches the previous sine wave. For this, it needs to: - Match its last frequency according to similarity threshold - Also be rising or falling

Parameters

- **is_freq_rising** (*bool*) – Informs if last value of previous sine wave was higher than it's predecessor.
- **prev_note_last_freq** (*float*) – Value of the last frequency present in previous sine wave.

`extend_with_lowest_note`() → None
Extend time vector and tone with lowest note length.

Sets up a new amount of samples to make tone from by adding a length of the lowest possible note in samples. Then recalculates time vector and tone of an instance.

get_last_freq() → float

Get last frequency field of an instance.

get_tone() → ndarray

Get tone field of an instance.

is_freq_rising_end() → bool

Check if tone is rising at the end.

sample_rate = 44100

similatiry_threshold = 0.015

1.1.6 src.settings_loop module

src.settings_loop.**settings_loop**() → None

Open settings menu and handle user input.

Opens a menu to choose settings type. Then, depending on the answer returns to main menu, opens processing settings or sonification settings. After that it opens a menu for specific settings type and allows user to pick the value they want to change. When values are on/off toggle style, picking them just changes their state. When values are non-boolean, user is prompted to input a value or return. User stays in function loop as long as they want to until inputing ‘r’ to return to main loop.

1.1.7 src.utils module

class src.utils.**Utils**

Bases: `object`

Container class for various functions.

settings_path

Stores path to settings.json.

Type

Path

notes_path

Stores path to notes.json.

Type

Path

static change_setting_to_opposite(*json_key: str*) → None

Change binary settings to opposite.

Changes two-state settings to opposite to what is currently set. Is able to change bool values, size/count and fixed/user.

Parameters

json_key – Key to change the value of.

static draw_tone(*tone: ndarray*) → None

Plots a soundwave on a chart.

Parameters

tone (*np.ndarray*) – Soundwave to draw.

static fix_value_in_json(*adress: Path*, *json_key: str*, *default_val: bool | str | int | float*) → None

Fix a value in json if it's broken.

Chcecks if a key is present in settings. If not, adds it with default value given.

Parameters

- **adress** (*Path*) – Path to json file.
- **json_key** (*str*) – Key to fix.
- **default_val** (*bool | str | int | float*) – Value for key if fixing is needed.

static get_curr_time_to_name() → str

Get current time.

Gets current time to add to audio file name.

Returns

Current time string.

static get_dict_from_json(*json_adress: Path*) → dict[str]

Get dictionary from json file.

Parameters

json_adress (*Path*) – Adress of the json file.

Returns

Dictionary extracted from json file.

static get_highest_lowest_note_possible_for_amount(*notes: list[str]*, *notes_amount: int*) → str

Get highest possible name of the lowest note.

Returns the highest possible lowest note name that can be selected when choosing a fixed number of notes from an ordered note list.

Parameters

- **notes** (*list[str]*) – List of note names.
- **notes_amount** (*int*) – Amount of notes for analog sonification.

Returns

Highest possible name of the lowest note.

`static get_highest_note_anal_safe`(*notes: list[str]*, *lowest_note: str*, *notes_amount: int*) → str | None

Get highest note name for analog sonification.

Checks if it is possible to get the highest note. Then, calculates the highest note name and returns it or returns None if check fails.

Parameters

- `notes` (*list[str]*) – List of note names.
- `lowest_note` (*str*) – Low note name.
- `notes_amount` (*int*) – Amount of notes for analog sonification.

Returns

Highest possible note name.

`static get_keys_from_json`(*json_adress: Path*) → list[str]

Get keys from json file.

Parameters

`json_adress` (*Path*) – Adress of the json file.

Returns

List of keys from chosen json file.

`static get_notes_used_list`(*notes: list[str]*, *lowest_note: str*, *notes_amount: int*) → list[str]

Get a list of notes to be used in analog sonification.

Gets a list of notes to be used in analog sonification based on amount ouf notes and lowest note name.

Parameters

- `notes` (*list[str]*) – List of note names.
- `lowest_note` (*str*) – Low note name.
- `notes_amount` (*int*) – Amount of notes for analog sonification.

Returns

List of names of notes to be used in analog sonification.

`static get_val_from_json`(*adress: Path*, *json_key: str*) → str | bool | int | float

Get value from json file for given key.

Parameters

- `adress` (*str*) – Path to json file.
- `json_key` (*str*) – Key to get value from.

static get_val_from_settings_fix(*json_key: Literal['AUTOMATIC_NORMALIZATION_AT_LOAD', 'AUTOMATIC_THRESHOLD_AT_LOAD', 'SHOW_THRESHOLD_ON_CHARTS', 'CUT_REMAINDER_SAMPLES_PAA', 'CUT_REMAINDER_SAMPLES_DWELLTIMES', 'SEGMENTING_STYLE_PAA', 'SEGMENTING_STYLE_DWELLTIMES', 'EMD_CONSIDER_IMFS_FROM', 'SAMPLE_RATE', 'BINARY_SONIF_LOW_NOTE', 'BINARY_SONIF_HIGH_NOTE', 'BINARY_SONIF_NOTE_DURATION_MILIS', 'ANAL_SONIF_NOTE_DURATION_MILIS', 'ANAL_SONIF_AMOUNT_OF_USED_NOTES', 'ANAL_SONIF_LOWEST_NOTE', 'SONIF_SIMILARITY_THRESHOLD']*, *default_val: str | bool | int | float = None*) → str | bool | int | float

Get val from settings, fix with automatic if needed.

Attempts to get a value from settings for given literal key. If fails, attempts to fix it with pre defined values.

Parameters

- **json_key** (*str*) – A literal key to get values from.
- **default_val** (*str|bool|int|float*) – Optional value for assigning non-predefined values.

Returns

Value read from settings or default if there were problems.

static human_read_milis(*milis: int*) → str

Convert milis to human readable.

Parameters

milis (*int*) – Miliseconds to convert to readable.

Returns

Miliseconds in human readable.

notes_path = PosixPath('/home/root-user/git-clones/soni/src/settings.json')

static save_value_to_settings(*json_key: str*, *json_val: bool | str | int | float*) → None

Save value with key to settings.json.

Parameters

- **json_key** (*str*) – Key for value to be saved.

- `json_val` (*bool | str | int | float*) – Value to be saved.

settings_path = PosixPath('/home/root-user/git-clones/soni/src/settings.json')