# birdears Documentation

Release 0.1.1

**lacchus Mercurius** 

# **CONTENTS**

1	Insta	alling birdears	3
	1.1	Installing the dependencies	3
		1.1.1 Arch Linux	3
	1.2	Installing birdears	3
		1.2.1 In-depth installation	3
2	Usin	g birdears	5
_	2.1	What is Functional Ear Training	5
	2.2	The method	5
	2.3	birdears modes	5
	2.4	basic usage	5
	2.4	2.4.1 melodic	6
		2.4.1 harmonic	7
		2.4.2 dictation	8
			9
	2.5		9 10
	2.5	6 61	
		1	10
		2.5.2 Creating new preset files	10
3	birde		11
	3.1	Subpackages	11
		3.1.1 birdears.interfaces package	11
		3.1.2 birdears.questions package	12
	3.2	Submodules	18
	3.3	birdears.interval module	18
	3.4	birdears.logger module	20
	3.5		21
	3.6		22
	3.7		23
	3.8		24
	3.9		25
4	Incta	alling birdears	27
7	4.1	8	-, 27
	4.1		27 27
	4.2		27 27
	4.2		27 27
		4.2.1 III-ucpul iiistanauon	/ د
5		8	29
	5.1	What is Functional Ear Training	29

	5.2 5.3 5.4	The method . birdears modes . basic usage . 5.4.1 melodic . 5.4.2 harmonic . 5.4.3 dictation . 5.4.4 instrumental . Loading from config/preset files . 5.5.1 Pre-made presets . 5.5.2 Creating new preset files	29 29 30 31 32 33 34 34 34
6	API		35
7	7.1 7.2 7.3 7.4 7.5 7.6 7.7 7.8 7.9	Subpackages Submodules birdears.interval module birdears.prequestion module birdears.questionbase module birdears.resolution module birdears.scale module	37 37 37 39 40 41 42 43 44
8	8.1 8.2 8.3 8.4 8.5	Submodules birdears.questions.melodicinterval module birdears.questions.melodicinterval module birdears.questions.melodicinterval module birdears.questions.melodicinterval module	47 47 49 51 52
9	<b>birde</b> 9.1 9.2	ears.interfaces package Submodules	<b>55</b> 55 55
Ру	thon I	Module Index	57
In	dex		59

Welcome to birdears documentation.

birdears is a software written in Python 3 for ear training for musicians (musical intelligence, transcribing music, composing). It is a clone of the method used by Functional Ear Trainer app for Android.

It comes with four modes, or four kind of exercises, which are: melodic, harmonic, dictation and instrumental.

In resume, with the *melodic* mode two notes are played one after the other and you have to guess the interval; with the *harmonic* mode, two notes are played simoutaneously (harmonically) and you should guess the interval.

With the dictation mode, more than 2 notes are played (ie., a melodic dictation) and you should tell what are the intervals between them.

With the *instrumental* mode, it is a like the *dictation*, but you will be expected to play the notes on your instrument, *ie.*, birdears will not wait for a typed reply and you should prectice with your own judgement. The melody can be repeat any times and you can have as much time as you want to try it out.

CONTENTS 1

2 CONTENTS

**CHAPTER** 

ONE

### **INSTALLING BIRDEARS**

# 1.1 Installing the dependencies

### 1.1.1 Arch Linux

```
sudo pacman -Syu sox python-pip
```

# 1.2 Installing birdears

To install, simple do this command with pip3

```
pip3 install --user --upgrade --no-cache-dir birdears
```

# 1.2.1 In-depth installation

You can choose to use a virtualenv to use birdears; this should give you an idea on how to setup one virtualenv.

You should first install virtualenv (for python3) using your distribution's package (supposing you're on linux), then issue on terminal:

```
virtualenv -p python3 ~/.venv # use the directory ~/.venv/ for the virtualenv
source ~/.venv/bin/activate # activate the virtualenv; this should be done
# every time you may want to run the software
# installed here.

pip3 install birdears # this will install the software
birdears --help # and this will run it
```

**CHAPTER** 

**TWO** 

# **USING BIRDEARS**

# 2.1 What is Functional Ear Training

write me!

### 2.2 The method

We can use abc language to notate music withing the documentation, ok

```
X: 1
T: Banish Misfortune
R: jig
M: 6/8
L: 1/8
K: Dmix
fed cAG| A2d cAG| F2D DED| FEF GFG|
AGA cAG| AGA cde|fed cAG| Ad^c d3:|
f2d d^cd| f2g agf| e2c cBc|e2f gfe|
f2g agf| e2f gfe|fed cAG|Ad^c d3:|
f2g e2f| d2e c2d|ABA GAG| F2F GED|
c3 cAG| AGA cde| fed cAG| Ad^c d3:|
```

### 2.3 birdears modes

birdears actually has four modes:

- · melodic interval question
- harmonic interval question
- melodic dictation question
- instrumental dictation question
- load from config file

# 2.4 basic usage

To see the commands avaliable just invoke the command without any arguments:

birdears

```
Usage: birdears <command> [options]
 birdears - Functional Ear Training for Musicians!
Options:
 --debug / --no-debug Turns on debugging; instead you can set DEBUG=1.
                     Show this message and exit.
Commands:
 dictation
             Melodic dictation
              Harmonic interval recognition
 harmonic
 instrumental Instrumental melodic time-based dictation
              Loads exercise from .toml config file...
             Melodic interval recognition
 melodic
 You can use 'birdears <command> --help' to show options for a specific
 command.
 More info at https://github.com/iacchus/birdears
```

```
birdears <command> --help
```

### 2.4.1 melodic

In this exercise birdears will play two notes, the tonic and the interval melodically, ie., one after the other and you should reply which is the correct distance between the two.

```
birdears melodic --help
```

```
Usage: birdears melodic [options]
 Melodic interval recognition
Options:
                               Mode of the question.
 -m, --mode <mode>
                                Tonic of the question.
 -t, --tonic <tonic>
 -o, --octave <octave>
                               Octave of the question.
 -d, --descending
                               Whether the question interval is descending.
 -c, --chromatic
                                If chosen, question has chromatic notes.
 -n, --n_octaves <n max>
                               Maximum number of octaves.
 -v, --valid_intervals <1,2,..> A comma-separated list without spaces
                                 of valid scale degrees to be chosen for the
                                 question.
 -q, --user_durations <1,0.5,n..>
                                 A comma-separated list without
                                 spaces with PRECISLY 9 floating values. Or
                                 'n' for default
                                                             duration.
 -p, --prequestion_method prequestion_method>
                                 The name of a pre-question method.
 -r, --resolution_method <resolution_method>
                                The name of a resolution method.
 -h, --help
                                 Show this message and exit.
```

```
In this exercise birdears will play two notes, the tonic and the interval
melodically, ie., one after the other and you should reply which is the
correct distance between the two.

Valid values are as follows:

-m <mode> is one of: major, dorian, phrygian, lydian, mixolydian, minor,
locrian

-t <tonic> is one of: A, A#, Ab, B, Bb, C, C#, D, D#, Db, E, Eb, F, F#, G,
G#, Gb

-p prequestion_method> is one of: none, tonic_only, progression_i_iv_v_i
-r <resolution_method> is one of: nearest_tonic, repeat_only
```

#### 2.4.2 harmonic

In this exercise birdears will play two notes, the tonic and the interval harmonically, ie., both on the same time and you should reply which is the correct distance between the two.

birdears harmonic --help

```
Usage: birdears harmonic [options]
 Harmonic interval recognition
Options:
 -m, --mode <mode>
                                Mode of the question.
 -t, --tonic <note>
                                Tonic of the question.
 -o, --octave <octave>
                               Octave of the question.
 -d, --descending
                               Whether the question interval is descending.
                                If chosen, question has chromatic notes.
 -c, --chromatic
 -n, --n_octaves <n max> Maximum number of octaves.
 -v, --valid_intervals <1,2,...> A comma-separated list without spaces
                                 of valid scale degrees to be chosen for the
                                 question.
 -q, --user_durations <1,0.5,n..>
                                 A comma-separated list without
                                 spaces with PRECISLY 9 floating values. Or
                                 'n' for default
                                                              duration.
 -p, --prequestion_method prequestion_method>
                                 The name of a pre-question method.
 -r, --resolution_method <resolution_method>
                                 The name of a resolution method.
 -h, --help
                                 Show this message and exit.
 In this exercise birdears will play two notes, the tonic and the interval
 harmonically, ie., both on the same time and you should reply which is the
 correct distance between the two.
 Valid values are as follows:
 -m <mode> is one of: major, dorian, phrygian, lydian, mixolydian, minor,
 locrian
```

2.4. basic usage 7

```
-t <tonic> is one of: A, A#, Ab, B, Bb, C, C#, D, D#, Db, E, Eb, F, F#, G,
G#, Gb
-p prequestion_method> is one of: none, tonic_only, progression_i_iv_v_i
-r <resolution_method> is one of: nearest_tonic, repeat_only
```

#### 2.4.3 dictation

In this exercise birdears will choose some random intervals and create a melodic dictation with them. You should reply the correct intervals of the melodic dictation.

birdears dictation --help

```
Usage: birdears dictation [options]
    Melodic dictation
Options:
    -m, --mode <mode>
                                                                               Mode of the question.
    -i, --max_intervals <n max> Max random intervals for the dictation. 
-x, --n_notes <n notes> Number of notes for the dictation.
    -t, --tonic <note>
                                                                                Tonic of the question.
                                                                          Octave of the question.
    -o, --octave <octave>
    -d, --descending
                                                                              Wether the question interval is descending.
    -c, --chromatic
                                                                                If chosen, question has chromatic notes.
    -n, --n_octaves <n max> Maximum number of octaves.
    -v, --valid_intervals <1,2,..> A comma-separated list without spaces
                                                                                  of valid scale degrees to be chosen for the
                                                                                   question.
    -q, --user_durations <1,0.5,n..>
                                                                                  A comma-separated list without
                                                                                   spaces with PRECISLY 9 floating values. Or
                                                                                   'n' for default
                                                                                                                                                        duration.
    -p, --prequestion_method prequestion_method>
                                                                                  The name of a pre-question method.
    -r, --resolution_method <resolution_method>
                                                                                  The name of a resolution method.
    -h, --help
                                                                                  Show this message and exit.
    In this exercise birdears will choose some random intervals and create a
    melodic dictation with them. You should reply the correct intervals of the
    melodic dictation.
    Valid values are as follows:
    -m <mode> is one of: major, dorian, phrygian, lydian, mixolydian, minor,
    -t <tonic> is one of: A, A#, Ab, B, Bb, C, C#, D, D#, Db, E, Eb, F, F#, G,
    G#, Gb
    -p -p content on the content of the conte
    -r <resolution_method> is one of: nearest_tonic, repeat_only
```

#### 2.4.4 instrumental

In this exercise birdears will choose some random intervals and create a melodic dictation with them. You should play the correct melody in you musical instrument.

birdears instrumental --help

```
Usage: birdears instrumental [options]
 Instrumental melodic time-based dictation
Options:
 -m, --mode <mode>
                                 Mode of the question.
 -w, --wait_time <seconds>
                                Time in seconds for next question/repeat.
 -u, --n_repeats <times>
                                Times to repeat question.
 -i, --max_intervals <n max> Max random intervals for the dictation.
 -x, --n_notes <n notes>
                               Number of notes for the dictation.
 -t, --tonic <note>
                                Tonic of the question.
 -o, --octave <octave>
                                Octave of the question.
 -d, --descending
                                Wether the question interval is descending.
                                If chosen, question has chromatic notes.
 -c, --chromatic
                               Maximum number of octaves.
 -n, --n_octaves <n max>
 -v, --valid_intervals <1,2,...> A comma-separated list without spaces
                                 of valid scale degrees to be chosen for the
                                 question.
 -q, --user_durations <1,0.5,n..>
                                 A comma-separated list without
                                 spaces with PRECISLY 9 floating values. Or
                                 'n' for default
                                                              duration.
 -p, --prequestion_method prequestion_method>
                                 The name of a pre-question method.
 -r, --resolution_method <resolution_method>
                                 The name of a resolution method.
 -h, --help
                                 Show this message and exit.
 In this exercise birdears will choose some random intervals and create a
 melodic dictation with them. You should play the correct melody in you
 musical instrument.
 Valid values are as follows:
 -m <mode> is one of: major, dorian, phrygian, lydian, mixolydian, minor,
 -t <tonic> is one of: A, A#, Ab, B, Bb, C, C#, D, D#, Db, E, Eb, F, F#, G,
 G#, Gb
 -p -p requestion_method> is one of: none, tonic_only, progression_i_iv_v_i
 -r <resolution_method> is one of: nearest_tonic, repeat_only
```

2.4. basic usage 9

# 2.5 Loading from config/preset files

# 2.5.1 Pre-made presets

birdears cointains some pre-made presets in it's presets/subdirectory.

The study for beginners is recommended by following the numeric order of those files (000, 001, then 002 etc.)

#### Pre-made presets description

write me

# 2.5.2 Creating new preset files

You can open the files cointained in birdears premade presets/ folder to have an ideia on how config files are made; it is simply the command line options written in a form toml standard.

**CHAPTER** 

THREE

### **BIRDEARS PACKAGE**

BirdEars provides facilities to musical ear training exercises.

# 3.1 Subpackages

### 3.1.1 birdears.interfaces package

#### **Submodules**

#### birdears.interfaces.commandline module

birdears.interfaces.commandline.CommandLine (exercise, \*\*kwargs)
This function implements the birdears loop for command line.

#### **Parameters**

- **exercise** (str) The question name.
- \*\*kwargs (kwargs) FIXME: The kwargs can contain options for specific questions.

birdears.interfaces.commandline.center\_text (text, sep=True, nl=0)

This function returns input text centered according to terminal columns.

#### **Parameters**

- **text** (str) The string to be centered, it can have multiple lines.
- **sep** (bool) Add line separator after centered text (True) or not (False).
- nl (int) How many new lines to add after text.

birdears.interfaces.commandline.make\_input\_str(user\_input, keyboard\_index)

Makes a string representing intervals entered by the user.

This function is to be used by questions which takes more than one interval input as MelodicDictation, and formats the intervals already entered.

#### **Parameters**

- user\_input (array\_type) The list of keyboard keys entered by user.
- **keyboard\_index** (array\_type) The keyboard mapping used by question.

birdears.interfaces.commandline.print instrumental(response)

Prints the formatted response for 'instrumental' exercise.

**Parameters** response (dict) – A response returned by question's check\_question()

```
birdears.interfaces.commandline.print_question(question)

Prints the question to the user.
```

**Parameters question** (obj) – A Question class with the question to be printed.

```
birdears.interfaces.commandline.print_response(response)
Prints the formatted response.
```

**Parameters** response (dict) – A response returned by question's check\_question()

### 3.1.2 birdears.questions package

#### **Submodules**

#### birdears.questions.harmonicinterval module

```
class birdears.questions.harmonicinterval.HarmonicIntervalQuestion(mode='major',
                                                                                    tonic=None,
                                                                                    oc-
                                                                                    tave=None,
                                                                                    descend-
                                                                                    ing=None,
                                                                                    chro-
                                                                                    matic=None.
                                                                                    n octaves=None,
                                                                                    valid intervals=None,
                                                                                    user_durations=None,
                                                                                    preques-
                                                                                    tion method='none',
                                                                                    resolu-
                                                                                    tion method='nearest tonic',
                                                                                    *args,
                                                                                    **kwargs)
```

 $Bases: \ \textit{birdears.questionbase.QuestionBase}$ 

Implements a Harmonic Interval test.

```
__init__ (mode='major', tonic=None, octave=None, descending=None, chromatic=None, n_octaves=None, valid_intervals=None, user_durations=None, prequestion_method='none', resolution_method='nearest_tonic', *args, **kwargs')

Inits the class.
```

#### **Parameters**

- mode (str) A string representing the mode of the question. Eg., 'major' or 'minor'
- **tonic** (*str*) A string representing the tonic of the question, eg.: 'C'; if omitted, it will be selected randomly.
- **octave** (*int*) A scienfic octave notation, for example, 4 for 'C4'; if not present, it will be randomly chosen.
- **descending** (bool) Is the question direction in descending, ie., intervals have lower pitch than the tonic.
- **chromatic** (bool) If the question can have (True) or not (False) chromatic intervals, ie., intervals not in the diatonic scale of tonic/mode.
- n\_octaves (int) Maximum number of octaves of the question.

- **valid\_intervals** (*list*) A list with intervals (int) valid for random choice, 1 is 1st, 2 is second etc. Eg. [1, 4, 5] to allow only tonics, fourths and fifths.
- user\_durations (str) A string with 9 comma-separated int or float's to set the default duration for the notes played. The values are respectively for: pre-question duration (1st), pre-question delay (2nd), and pre-question pos-delay (3rd); question duration (4th), question delay (5th), and question pos-delay (6th); resolution duration (7th), resolution delay (8th), and resolution pos-delay (9th). duration is the duration in of the note in seconds; delay is the time to wait before playing the next note, and pos\_delay is the time to wait after all the notes of the respective sequence have been played. If any of the user durations is 'n, the default duration for the type of question will be used instead. Example:

```
"2,0.5,1,2,n,0,2.5,n,1"
```

- **prequestion\_method** (str) Method of playing a cadence or the exercise tonic before the question so to affirm the question musical tonic key to the ear. Valid ones are registered in the *birdears.prequestion.PREQUESION\_METHODS* global dict.
- **resolution\_method** (str) Method of playing the resolution of an exercise. Valid ones are registered in the *birdears.resolution.RESOLUTION\_METHODS* global dict.

```
check_question (user_input_char)
    Checks whether the given answer is correct.
make_pre_question (method)
make_question ()
make_resolution (method)
play_question ()
play_resolution ()
```

3.1. Subpackages 13

#### birdears.questions.instrumentaldictation module

class birdears.questions.instrumentaldictation.InstrumentalDictationQuestion(mode='major',

 $wait\_time=11,$  $n_repeats=1$ ,  $max_intervals=3$ ,  $n \ notes=4$ , tonic=None, octave=None, descending=None, chromatic=None, n octaves=None, valid intervals=Noi user durations=No. prequestion\_method='prog res-0lution\_method='reped \*args, \*\*kwargs)

Bases: birdears.questionbase.QuestionBase

Implements an instrumental dictation test.

```
__init__ (mode='major', wait_time=11, n_repeats=1, max_intervals=3, n_notes=4, tonic=None, octave=None, descending=None, chromatic=None, n_octaves=None, valid_intervals=None, user_durations=None, prequestion_method='progression_i_iv_v_i', resolution_method='repeat_only', *args, **kwargs'

Inits the class.
```

#### **Parameters**

- mode (str) A string representing the mode of the question. Eg., 'major' or 'minor'.
- wait\_time (float) Wait time in seconds for the next question or repeat.
- **n\_repeats** (*int*) Number of times the same dictation will be repeated before the end of the exercise.
- max\_intervals (int) The maximum number of random intervals the question will have.
- n notes (int) The number of notes the melodic dictation will have.
- **tonic** (*str*) A string representing the tonic of the question, eg.: 'C'; if omitted, it will be selected randomly.
- **octave** (*int*) A scienfic octave notation, for example, 4 for 'C4'; if not present, it will be randomly chosen.
- descending (bool) Is the question direction in descending, ie., intervals have lower pitch than the tonic.

- **chromatic** (bool) If the question can have (True) or not (False) chromatic intervals, ie., intervals not in the diatonic scale of tonic/mode.
- n\_octaves (int) Maximum number of octaves of the question.
- **valid\_intervals** (*list*) A list with intervals (int) valid for random choice, 1 is 1st, 2 is second etc. Eg. [1, 4, 5] to allow only tonics, fourths and fifths.
- user\_durations (str) A string with 9 comma-separated int or float's to set the default duration for the notes played. The values are respectively for: pre-question duration (1st), pre-question delay (2nd), and pre-question pos-delay (3rd); question duration (4th), question delay (5th), and question pos-delay (6th); resolution duration (7th), resolution delay (8th), and resolution pos-delay (9th). duration is the duration in of the note in seconds; delay is the time to wait before playing the next note, and pos\_delay is the time to wait after all the notes of the respective sequence have been played. If any of the user durations is 'n, the default duration for the type of question will be used instead. Example:

```
"2,0.5,1,2,n,0,2.5,n,1"
```

- **prequestion\_method** (*str*) Method of playing a cadence or the exercise tonic before the question so to affirm the question musical tonic key to the ear. Valid ones are registered in the *birdears.prequestion.PREQUESION\_METHODS* global dict.
- **resolution\_method** (str) Method of playing the resolution of an exercise. Valid ones are registered in the *birdears.resolution.RESOLUTION\_METHODS* global dict.

#### check\_question()

Checks whether the given answer is correct.

This currently doesn't applies to instrumental dictation questions.

```
make_pre_question (method)
make_question (phrase_semitones)
make_resolution (method)
play_question()
```

3.1. Subpackages 15

#### birdears.guestions.melodicdictation module

```
\textbf{class} \  \, \texttt{birdears.questions.melodicdictation.MelodicDictationQuestion} \, (\textit{mode='major'}, \\
                                                                                               max_intervals=3,
                                                                                               n_notes=4,
                                                                                               tonic=None,
                                                                                               oc-
                                                                                               tave=None,
                                                                                               descend-
                                                                                               ing=None,
                                                                                               chro-
                                                                                               matic=None,
                                                                                               n octaves=None,
                                                                                                valid intervals=None,
                                                                                               user_durations=None,
                                                                                               preques-
                                                                                               tion method='progression i iv v
                                                                                               tion_method='repeat_only',
                                                                                                *args,
```

Bases: birdears.questionbase.QuestionBase

Implements a melodic dictation test.

```
__init__ (mode='major', max_intervals=3, n_notes=4, tonic=None, octave=None, de-
scending=None, chromatic=None, n_octaves=None, valid_intervals=None,
user_durations=None, prequestion_method='progression_i_iv_v_i', resolu-
tion_method='repeat_only', *args, **kwargs)
Inits the class.
```

#### **Parameters**

- mode (str) A string representing the mode of the question. Eg., 'major' or 'minor'.
- max\_intervals (int) The maximum number of random intervals the question will have
- **n\_notes** (*int*) The number of notes the melodic dictation will have.
- **tonic** (*str*) A string representing the tonic of the question, eg.: 'C'; if omitted, it will be selected randomly.
- **octave** (*int*) A scienfic octave notation, for example, 4 for 'C4'; if not present, it will be randomly chosen.
- **descending** (bool) Is the question direction in descending, ie., intervals have lower pitch than the tonic.
- **chromatic** (bool) If the question can have (True) or not (False) chromatic intervals, ie., intervals not in the diatonic scale of tonic/mode.
- **n\_octaves** (*int*) Maximum number of octaves of the question.
- **valid\_intervals** (*list*) A list with intervals (int) valid for random choice, 1 is 1st, 2 is second etc. Eg. [1, 4, 5] to allow only tonics, fourths and fifths.
- user\_durations (str) A string with 9 comma-separated int or float's to set the default duration for the notes played. The values are respectively for: pre-question duration (1st), pre-question delay (2nd), and pre-question pos-delay (3rd); question duration

\*\*kwargs)

(4th), question delay (5th), and question pos-delay (6th); resolution duration (7th), resolution delay (8th), and resolution pos-delay (9th). duration is the duration in of the note in seconds; delay is the time to wait before playing the next note, and pos\_delay is the time to wait after all the notes of the respective sequence have been played. If any of the user durations is 'n, the default duration for the type of question will be used instead. Example:

```
"2,0.5,1,2,n,0,2.5,n,1"
```

- **prequestion\_method** (str) Method of playing a cadence or the exercise tonic before the question so to affirm the question musical tonic key to the ear. Valid ones are registered in the *birdears.prequestion.PREQUESION\_METHODS* global dict.
- **resolution\_method** (str) Method of playing the resolution of an exercise. Valid ones are registered in the *birdears.resolution.RESOLUTION\_METHODS* global dict.

#### birdears.questions.melodicinterval module

make\_resolution (method)

```
class birdears.questions.melodicinterval.MelodicIntervalQuestion(mode='major',
                                                                                  tonic=None,
                                                                                  oc-
                                                                                  tave=None,
                                                                                  descend-
                                                                                  ing=None,
                                                                                  chro-
                                                                                  matic=None,
                                                                                  n octaves=None,
                                                                                  valid intervals=None,
                                                                                  user durations=None,
                                                                                  preques-
                                                                                  tion_method='tonic_only',
                                                                                  resolu-
                                                                                  tion_method='nearest_tonic',
                                                                                  *args,
                                                                                  **kwargs)
     Bases: birdears.questionbase.QuestionBase
     Implements a Melodic Interval test.
     check_question (user_input_char)
          Checks whether the given answer is correct.
     make_pre_question (method)
     make_question()
```

3.1. Subpackages 17

```
play_question()
play_resolution()
```

### 3.2 Submodules

### 3.3 birdears.interval module

Bases: birdears.interval.IntervalBase

Chooses a diatonic interval for the question.

#### tonic octave

int – Scientific octave for the tonic. For example, if the tonic is a 'C4' then tonic\_octave is 4.

#### interval octave

int – Scientific octave for the interval. For example, if the interval is a 'G5' then tonic\_octave is 5.

#### chromatic offset

int – The offset in semitones inside one octave; maybe it will be deprecated in favour of distance ['semitones'] which is the same.

#### note\_and\_octave

str – Note and octave of the interval, for example, if the interval is G5 the note name is 'G5'.

#### note\_name

str – The note name of the interval, for example, if the interval is G5 then the name is 'G'.

#### semitones

int – Semitones from tonic to octave. If tonic is C4 and interval is G5 the number of semitones is 19.

#### is\_chromatic

bool – If the current interval is chromatic (True) or if it exists in the diatonic scale which key is tonic.

#### is descending

bool – If the interval has a descending direction, ie., has a lower pitch than the tonic.

#### diatonic\_index

int – If the interval is chromatic, this will be the nearest diatonic interval in the direction of the resolution (closest tonic.) From II to IV degrees, it is the diatonic interval before; from V to VII it is the diatonic interval after.

#### distance

dict – A dictionary which the distance from tonic to interval, for example, if tonic is C4 and interval is G5:

```
{
   'octaves': 1,
   'semitones': 7
}
```

#### data

*tuple* – A tuple representing the interval data in the form of (semitones, short\_name, long\_name), for example:

```
(19, 'P12', 'Perfect Twelfth')
```

#### Todo:

- Maybe we should refactor some of the attributes with a tuple (note, octave)
- Maybe remove *chromatic\_offset* in favor of *distance['semitones']'*

\_\_init\_\_ (mode, tonic, octave, n\_octaves=None, descending=None, valid\_intervals=None)
Inits the class and choses a random interval with the given args.

#### **Parameters**

- mode (str) Diatonic mode for the interval. (eg.: 'major' or 'minor')
- tonic (str) Tonic of the scale. (eg.: 'Bb')
- octave (str) Scientific octave of the scale (eg.: 4)
- **interval** (*str*) Not implemented. The interval.
- **chromatic** (bool) Can have chromatic notes? (eg.: F# in a key of C; default: false)
- n\_octaves (int) Maximum number os octaves (eg. 2)
- **descending** (bool) Is the interval descending? (default: false)
- **valid\_intervals** (*int*) A list with inervals valid for random choice, 1 is 1st, 2 is second etc.

class birdears.interval.DiatonicInterval (mode, tonic, octave, n\_octaves=None, descending=None, valid intervals=None)

Bases: birdears.interval.IntervalBase

Chooses a diatonic interval for the question.

#### tonic\_octave

int – Scientific octave for the tonic. For example, if the tonic is a 'C4' then tonic\_octave is 4.

#### interval octave

int – Scientific octave for the interval. For example, if the interval is a 'G5' then tonic octave is 5.

#### chromatic offset

*int* – The offset in semitones inside one octave. Relative semitones to tonic.

#### note\_and\_octave

str – Note and octave of the interval, for example, if the interval is G5 the note name is 'G5'.

#### note\_name

str – The note name of the interval, for example, if the interval is G5 then the name is 'G'.

#### semitones

int – Semitones from tonic to octave. If tonic is C4 and interval is G5 the number of semitones is 19.

#### is\_chromatic

bool - If the current interval is chromatic (True) or if it exists in the diatonic scale which key is tonic.

#### is\_descending

bool – If the interval has a descending direction, ie., has a lower pitch than the tonic.

#### diatonic index

int – If the interval is chromatic, this will be the nearest diatonic interval in the direction of the resolution (closest tonic.) From II to IV degrees, it is the diatonic interval before; from V to VII it is the diatonic interval after.

#### distance

dict – A dictionary which the distance from tonic to interval, for example, if tonic is C4 and interval is G5:

```
{
   'octaves': 1,
   'semitones': 7
}
```

#### data

*tuple* – A tuple representing the interval data in the form of (semitones, short\_name, long\_name), for example:

```
(19, 'P12', 'Perfect Twelfth')
```

\_\_init\_\_ (mode, tonic, octave, n\_octaves=None, descending=None, valid\_intervals=None)
Inits the class and choses a random interval with the given args.

#### **Parameters**

- mode (str) Diatonic mode for the interval. (eg.: 'major' or 'minor')
- tonic (str) Tonic of the scale. (eg.: 'Bb')
- octave (str) Scientific octave of the scale (eg.: 4)
- n\_octaves (int) Maximum number os octaves (eg. 2)
- **descending** (bool) Is the interval descending? (default: false)
- **valid\_intervals** (*int*) A list with intervals (int) valid for random choice, 1 is 1st, 2 is second etc.

```
class birdears.interval.IntervalBase
    Bases: object
    __init__()
         Base class for interval classes.
```

```
return\_simple(keys)
```

This method returns a dict with only the values passed to keys.

# 3.4 birdears.logger module

This submodule exports *logger* to log events.

Logging messages which are less severe than *lvl* will be ignored:

Level	Numeric value
CRITICAL	50
ERROR	40
WARNING	30
INFO	20
DEBUG	10
NOTSET	0
Level	When it's used
DEBUG	Detailed information, typically of interest only when diagnosing problems.

```
INFO Confirmation that things are working as expected.

WARNING An indication that something unexpected happened, or indicative of some problem in the near future (e.g. 'disk space low').

The software is still working as expected.

ERROR Due to a more serious problem, the software has not been able to perform some function.

CRITICAL A serious error, indicating that the program itself may be unable to continue running.
```

```
birdears.logger.log_event (f, *args, **kwargs)
```

Decorator. Functions and method decorated with this decorator will have their signature logged when birdears is executed with *-debug* mode. Both function signature with their call values and their return will be logged.

# 3.5 birdears.prequestion module

This module implements pre-questions' progressions.

Pre questions are chord progressions or notes played before the question is played, so to affirmate the sound of the question's key.

For example a common cadence is chords I-IV-V-I from the diatonic scale, which in a key of C is CM-FM-GM-CM and in a key of A is AM-DM-EM-AM.

Pre-question methods should be decorated with *register\_prequestion\_method* decorator, so that they will be registered as a valid pre-question method.

```
class birdears.prequestion.PreQuestion (method, question)

Bases: object

__call__(*args, **kwargs)

Calls the resolution method and pass arguments to it.
```

Returns a birdears. Sequence object with the pre-question generated by the method.

```
__init__ (method, question)
```

This class implements methods for different types of pre-question progressions.

#### **Parameters**

- **method** (str) The method used in the pre question.
- question (obj) Question object from which to generate the
- sequence. (pre-question) -

birdears.prequestion.none(question, \*args, \*\*kwargs)

Pre-question method that return an empty sequence with no delay. :param question: Question object from which to generate the

pre-question sequence. (this is provided by the *Resolution* class when it is '\_\_call\_\_'ed)

```
birdears.prequestion.progression_i_iv_v_i (question, *args, **kwargs)
```

Pre-question method that play's a chord progression with triad chords built on the grades I, IV, V the I of the question key.

```
Parameters question (obj) – Question object from which to generate the pre-question sequence. (this is provided by the Resolution class when it is '__call__'ed)
```

```
birdears.prequestion.register_prequestion_method(f, *args, **kwargs)

Decorator for prequestion method functions.
```

Functions decorated with this decorator will be registered in the PREQUESTION METHODS global dict.

```
birdears.prequestion.tonic_only(question, *args, **kwargs)
```

Pre-question method that only play's the question tonic note before the question.

**Parameters question** (obj) – Question object from which to generate the pre-question sequence. (this is provided by the *Resolution* class when it is '\_\_call\_\_'ed)

# 3.6 birdears.questionbase module

Bases: object

Base Class to be subclassed for Question classes.

This class implements attributes and routines to be used in Question subclasses.

```
__init__ (mode='major', tonic=None, octave=None, descending=None, chromatic=None, n_octaves=None, valid_intervals=None, user_durations=None, prequestion_method=None, resolution_method=None, default_durations=None, *args, **kwargs)
Inits the class.
```

#### **Parameters**

- mode (str) A string represnting the mode of the question. Eg., 'major' or 'minor'
- **tonic** (*str*) A string representing the tonic of the question, eg.: 'C'; if omitted, it will be selected randomly.
- **octave** (*int*) A scienfic octave notation, for example, 4 for 'C4'; if not present, it will be randomly chosen.
- **descending** (bool) Is the question direction in descending, ie., intervals have lower pitch than the tonic.
- **chromatic** (bool) If the question can have (True) or not (False) chromatic intervals, ie., intervals not in the diatonic scale of tonic/mode.
- n\_octaves (int) Maximum numbr of octaves of the question.
- **valid\_intervals** (*list*) A list with intervals (int) valid for random choice, 1 is 1st, 2 is second etc. Eg. [1, 4, 5] to allow only tonics, fourths and fifths.
- user\_durations (dict) A string with 9 comma-separated int or float's to set the default duration for the notes played. The values are respectively for: pre-question duration (1st), pre-question delay (2nd), and pre-question pos-delay (3rd); question duration (4th), question delay (5th), and question pos-delay (6th); resolution duration (7th), resolution delay (8th), and resolution pos-delay (9th). duration is the duration in of the note in seconds; delay is the time to wait before playing the next note, and pos\_delay is the time to wait after all the notes of the respective sequence have been played. If any of the user durations is 'n, the default duration for the type of question will be used instead. Example:

```
"2,0.5,1,2,n,0,2.5,n,1"
```

- **prequestion\_method** (str) Method of playing a cadence or the exercise tonic before the question so to affirm the question musical tonic key to the ear. Valid ones are registered in the *birdears.prequestion.PREQUESION\_METHODS* global dict.
- **resolution\_method** (str) Method of playing the resolution of an exercise Valid ones are registered in the *birdears.resolution.RESOLUTION\_METHODS* global dict.
- **user\_durations** Dictionary with the default durations for each type of sequence. This is provided by the subclasses.

#### check\_question()

This method should be overwritten by the question subclasses.

#### get valid semitones()

Returns a list with valid semitones for question.

#### make\_question()

This method should be overwritten by the question subclasses.

#### make resolution()

This method should be overwritten by the question subclasses.

#### play\_question()

This method should be overwritten by the question subclasses.

```
birdears.questionbase.register_question_class(f, *args, **kwargs)
```

Decorator for question classes.

Classes decorated with this decorator will be registered in the QUESTION\_CLASSES global.

### 3.7 birdears.resolution module

```
class birdears.resolution.Resolution(method, question)
    Bases: object
```

This class implements methods for different types of question resolutions.

A resolution is an answer to a question. It aims to create a mnemonic on how the inverval resvolver to the tonic.

```
__call__(*args, **kwargs)
```

Calls the resolution method and pass arguments to it.

Returns a birdears. Sequence object with the resolution generated by the method.

```
__init__ (method, question)
```

Inits the resolution class.

#### **Parameters**

- **method** (str) The method used in the resolution.
- question (obj) Question object from which to generate the
- sequence. (resolution) -

```
birdears.resolution.nearest_tonic(question)
```

Resolution method that resolve the intervals to their nearest tonics.

**Parameters question** (obj) – Question object from which to generate the resolution sequence. (this is provided by the *Prequestion* class when it is '\_\_call\_\_'ed)

birdears.resolution.register\_resolution\_method(f, \*args, \*\*kwargs)

Decorator for resolution method functions.

Functions decorated with this decorator will be registered in the RESOLUTION\_METHODS global dict.

birdears.resolution.repeat\_only(question)

Resolution method that only repeats the sequence elements with given durations.

**Parameters question** (obj) – Question object from which to generate the resolution sequence. (this is provided by the *Prequestion* class when it is '\_\_call\_\_'ed)

### 3.8 birdears.scale module

Bases: birdears.scale.ScaleBase

Builds a musical chromatic scale.

#### scale

*array\_type* – The array of notes representing the scale.

\_\_init\_\_(tonic, octave=None, n\_octaves=None, descending=None, dont\_repeat\_tonic=None)
Returns a chromatic scale from tonic.

#### **Parameters**

- tonic (str) The note which the scale will be built upon.
- **octave** (*int*) The scientific octave the scale will be built upon.
- n octaves (int) The number of octaves the scale will contain.
- **descending** (bool) Whether the scale is descending.
- **dont\_repeat\_tonic** (bool) Whether to skip appending the last note (octave) to the scale.

get\_triad (mode, index=0, degree=None)

Returns an array with notes from a scale's triad.

#### **Parameters**

- mode (str) Mode of the scale (eg. 'major' or 'minor')
- index (int) Triad index (eg.: 0 for 1st degree triad.)
- **degree** (*int*) Degree of the scale. If provided, overrides the *index* argument. (eg.: 1 for the 1st degree triad.)

Returns A list with three pitches (str), one for each note of the triad.

Bases: birdears.scale.ScaleBase

Builds a musical diatonic scale.

#### scale

*array\_type* – The array of notes representing the scale.

\_\_init\_\_ (tonic, mode=None, octave=None, n\_octaves=None, descending=None, dont repeat tonic=None)

Returns a diatonic scale from tonic and mode.

#### **Parameters**

- tonic (str) The note which the scale will be built upon.
- mode (str) The mode the scale will be built upon. ('major' or 'minor')
- **octave** (*int*) The scientific octave the scale will be built upon.
- **n\_octaves** (*int*) The number of octaves the scale will contain.
- **descending** (bool) Whether the scale is descending.
- dont\_repeat\_tonic (bool) Whether to skip appending the last note (octave) to the scale.

#### get\_triad (index=0, degree=None)

Returns an array with notes from a scale's triad.

#### **Parameters**

- index (int) triad index (eg.: 0 for 1st degree triad.)
- **degree** (*int*) Degree of the scale. If provided, overrides the *index* argument. (eg.: *1* for the 1st degree triad.)

Returns An array with three pitches, one for each note of the triad.

```
class birdears.scale.ScaleBase
    Bases: object
```

# 3.9 birdears.sequence module

```
class birdears.sequence (elements=[], duration=2, delay=1.5, pos_delay=1)
    Bases: object
```

Register a Sequence of notes and/or chords.

#### elements

array type – List of notes (strings) ou chords (list of strings) in this Sequence.

#### append (elements)

Appends *elements* to Sequence.elements

**Parameters** elements (array\_type) – Elements to be appended to the class.

```
async_play (callback, end_callback)
```

Plays the Sequence elements of notes and/or chords and wait for Sequence.pos\_delay seconds.

#### extend(elements)

Extends Sequence.elements with elements.

**Parameters elements** (array\_type) – elements extend the class with.

#### make\_chord\_progression (tonic, mode, degrees)

Appends triad chord(s) to the Sequence.

#### **Parameters**

• tonic (str) - Tonic note of the scale.

- mode(str) Mode of the scale from which build the triads upon.
- **degrees** (array\_type) List with integers represending the degrees of each triad.

play (callback=None, end\_callback=None)

play\_element (index)

Plays element sequence.elements[index].

**CHAPTER** 

**FOUR** 

### **INSTALLING BIRDEARS**

# 4.1 Installing the dependencies

### 4.1.1 Arch Linux

```
sudo pacman -Syu sox python-pip
```

# 4.2 Installing birdears

To install, simple do this command with pip3

```
pip3 install --user --upgrade --no-cache-dir birdears
```

# 4.2.1 In-depth installation

You can choose to use a virtualenv to use birdears; this should give you an idea on how to setup one virtualenv.

You should first install virtualenv (for python3) using your distribution's package (supposing you're on linux), then issue on terminal:

```
virtualenv -p python3 ~/.venv # use the directory ~/.venv/ for the virtualenv
source ~/.venv/bin/activate # activate the virtualenv; this should be done
# every time you may want to run the software
# installed here.

pip3 install birdears # this will install the software
birdears --help # and this will run it
```

**CHAPTER** 

**FIVE** 

# **USING BIRDEARS**

# 5.1 What is Functional Ear Training

write me!

## 5.2 The method

We can use abc language to notate music withing the documentation, ok

```
X: 1
T: Banish Misfortune
R: jig
M: 6/8
L: 1/8
K: Dmix
fed cAG| A2d cAG| F2D DED| FEF GFG|
AGA cAG| AGA cde|fed cAG| Ad^c d3:|
f2d d^cd| f2g agf| e2c cBc|e2f gfe|
f2g agf| e2f gfe|fed cAG|Ad^c d3:|
f2d e2f| d2e c2d|ABA GAG| F2F GED|
c3 cAG| AGA cde| fed cAG| Ad^c d3:|
```

### 5.3 birdears modes

birdears actually has four modes:

- · melodic interval question
- harmonic interval question
- melodic dictation question
- instrumental dictation question
- load from config file

# 5.4 basic usage

To see the commands avaliable just invoke the command without any arguments:

birdears

```
Usage: birdears <command> [options]
 birdears - Functional Ear Training for Musicians!
Options:
 --debug / --no-debug Turns on debugging; instead you can set DEBUG=1.
                     Show this message and exit.
Commands:
 dictation
             Melodic dictation
              Harmonic interval recognition
 harmonic
 instrumental Instrumental melodic time-based dictation
              Loads exercise from .toml config file...
             Melodic interval recognition
 melodic
 You can use 'birdears <command> --help' to show options for a specific
 command.
 More info at https://github.com/iacchus/birdears
```

```
birdears <command> --help
```

### 5.4.1 melodic

In this exercise birdears will play two notes, the tonic and the interval melodically, ie., one after the other and you should reply which is the correct distance between the two.

```
birdears melodic --help
```

```
Usage: birdears melodic [options]
 Melodic interval recognition
Options:
                               Mode of the question.
 -m, --mode <mode>
                                Tonic of the question.
 -t, --tonic <tonic>
 -o, --octave <octave>
                               Octave of the question.
 -d, --descending
                               Whether the question interval is descending.
 -c, --chromatic
                                If chosen, question has chromatic notes.
 -n, --n_octaves <n max>
                               Maximum number of octaves.
 -v, --valid_intervals <1,2,..> A comma-separated list without spaces
                                 of valid scale degrees to be chosen for the
                                 question.
 -q, --user_durations <1,0.5,n..>
                                 A comma-separated list without
                                 spaces with PRECISLY 9 floating values. Or
                                 'n' for default
                                                             duration.
 -p, --prequestion_method prequestion_method>
                                 The name of a pre-question method.
 -r, --resolution_method <resolution_method>
                                The name of a resolution method.
 -h, --help
                                 Show this message and exit.
```

```
In this exercise birdears will play two notes, the tonic and the interval
melodically, ie., one after the other and you should reply which is the
correct distance between the two.

Valid values are as follows:

-m <mode> is one of: major, dorian, phrygian, lydian, mixolydian, minor,
locrian

-t <tonic> is one of: A, A#, Ab, B, Bb, C, C#, D, D#, Db, E, Eb, F, F#, G,
G#, Gb

-p prequestion_method> is one of: none, tonic_only, progression_i_iv_v_i
-r <resolution_method> is one of: nearest_tonic, repeat_only
```

#### 5.4.2 harmonic

In this exercise birdears will play two notes, the tonic and the interval harmonically, ie., both on the same time and you should reply which is the correct distance between the two.

birdears harmonic --help

```
Usage: birdears harmonic [options]
 Harmonic interval recognition
Options:
 -m, --mode <mode>
                                Mode of the question.
 -t, --tonic <note>
                                Tonic of the question.
                               Octave of the question.
 -o, --octave <octave>
 -d, --descending
                               Whether the question interval is descending.
                                If chosen, question has chromatic notes.
 -c, --chromatic
 -n, --n_octaves <n max> Maximum number of octaves.
 -v, --valid_intervals <1,2,...> A comma-separated list without spaces
                                 of valid scale degrees to be chosen for the
                                 question.
 -q, --user_durations <1,0.5,n..>
                                 A comma-separated list without
                                 spaces with PRECISLY 9 floating values. Or
                                 'n' for default
                                                              duration.
 -p, --prequestion_method prequestion_method>
                                 The name of a pre-question method.
 -r, --resolution_method <resolution_method>
                                 The name of a resolution method.
 -h, --help
                                 Show this message and exit.
 In this exercise birdears will play two notes, the tonic and the interval
 harmonically, ie., both on the same time and you should reply which is the
 correct distance between the two.
 Valid values are as follows:
 -m <mode> is one of: major, dorian, phrygian, lydian, mixolydian, minor,
 locrian
```

5.4. basic usage 31

```
-t <tonic> is one of: A, A#, Ab, B, Bb, C, C#, D, D#, Db, E, Eb, F, F#, G,
G#, Gb
-p prequestion_method> is one of: none, tonic_only, progression_i_iv_v_i
-r <resolution_method> is one of: nearest_tonic, repeat_only
```

#### 5.4.3 dictation

In this exercise birdears will choose some random intervals and create a melodic dictation with them. You should reply the correct intervals of the melodic dictation.

birdears dictation --help

```
Usage: birdears dictation [options]
    Melodic dictation
Options:
    -m, --mode <mode>
                                                                           Mode of the question.
    -i, --max_intervals <n max> Max random intervals for the dictation.
    -x, --n_notes <n notes> Number of notes for the dictation.
    -t, --tonic <note>
                                                                            Tonic of the question.
                                                                        Octave of the question.
    -o, --octave <octave>
    -d, --descending
                                                                           Wether the question interval is descending.
    -c, --chromatic
                                                                            If chosen, question has chromatic notes.
    -n, --n_octaves <n max> Maximum number of octaves.
    -v, --valid_intervals <1,2,..> A comma-separated list without spaces
                                                                              of valid scale degrees to be chosen for the
                                                                               question.
    -q, --user_durations <1,0.5,n..>
                                                                               A comma-separated list without
                                                                               spaces with PRECISLY 9 floating values. Or
                                                                               'n' for default
                                                                                                                                                  duration.
    -p, --prequestion_method prequestion_method>
                                                                              The name of a pre-question method.
    -r, --resolution_method <resolution_method>
                                                                               The name of a resolution method.
    -h, --help
                                                                               Show this message and exit.
    In this exercise birdears will choose some random intervals and create a
    melodic dictation with them. You should reply the correct intervals of the
    melodic dictation.
    Valid values are as follows:
    -m <mode> is one of: major, dorian, phrygian, lydian, mixolydian, minor,
    -t <tonic> is one of: A, A#, Ab, B, Bb, C, C#, D, D#, Db, E, Eb, F, F#, G,
    G#, Gb
    -p -p content on the content of the conte
    -r <resolution_method> is one of: nearest_tonic, repeat_only
```

### 5.4.4 instrumental

In this exercise birdears will choose some random intervals and create a melodic dictation with them. You should play the correct melody in you musical instrument.

birdears instrumental --help

```
Usage: birdears instrumental [options]
 Instrumental melodic time-based dictation
Options:
 -m, --mode < mode >
                                 Mode of the question.
 -w, --wait_time <seconds>
                                Time in seconds for next question/repeat.
 -u, --n_repeats <times>
                                Times to repeat question.
 -i, --max_intervals <n max> Max random intervals for the dictation.
 -x, --n_notes <n notes>
                               Number of notes for the dictation.
 -t, --tonic <note>
                                Tonic of the question.
 -o, --octave <octave>
                                Octave of the question.
 -d, --descending
                                Wether the question interval is descending.
                                If chosen, question has chromatic notes.
 -c, --chromatic
                               Maximum number of octaves.
 -n, --n_octaves <n max>
 -v, --valid_intervals <1,2,...> A comma-separated list without spaces
                                 of valid scale degrees to be chosen for the
                                 question.
 -q, --user_durations <1,0.5,n..>
                                 A comma-separated list without
                                 spaces with PRECISLY 9 floating values. Or
                                 'n' for default
                                                              duration.
 -p, --prequestion_method prequestion_method>
                                 The name of a pre-question method.
 -r, --resolution_method <resolution_method>
                                 The name of a resolution method.
 -h, --help
                                 Show this message and exit.
 In this exercise birdears will choose some random intervals and create a
 melodic dictation with them. You should play the correct melody in you
 musical instrument.
 Valid values are as follows:
 -m <mode> is one of: major, dorian, phrygian, lydian, mixolydian, minor,
 locrian
 -t <tonic> is one of: A, A#, Ab, B, Bb, C, C#, D, D#, Db, E, Eb, F, F#, G,
 G#, Gb
 -p -p requestion_method> is one of: none, tonic_only, progression_i_iv_v_i
 -r <resolution_method> is one of: nearest_tonic, repeat_only
```

5.4. basic usage 33

# 5.5 Loading from config/preset files

# 5.5.1 Pre-made presets

birdears cointains some pre-made presets in it's presets/subdirectory.

The study for beginners is recommended by following the numeric order of those files (000, 001, then 002 etc.)

## Pre-made presets description

write me

## 5.5.2 Creating new preset files

You can open the files cointained in birdears premade presets/ folder to have an ideia on how config files are made; it is simply the command line options written in a form toml standard.

CHAPTER
SIX

API

36 Chapter 6. API

## **BIRDEARS PACKAGE**

BirdEars provides facilities to musical ear training exercises.

# 7.1 Subpackages

## 7.2 Submodules

## 7.3 birdears.interval module

Bases: birdears.interval.IntervalBase

Chooses a diatonic interval for the question.

## tonic\_octave

int – Scientific octave for the tonic. For example, if the tonic is a 'C4' then tonic\_octave is 4.

#### interval octave

int – Scientific octave for the interval. For example, if the interval is a 'G5' then tonic\_octave is 5.

#### chromatic offset

int – The offset in semitones inside one octave; maybe it will be deprecated in favour of distance['semitones'] which is the same.

### note\_and\_octave

str – Note and octave of the interval, for example, if the interval is G5 the note name is 'G5'.

#### note name

str – The note name of the interval, for example, if the interval is G5 then the name is 'G'.

#### semitones

int – Semitones from tonic to octave. If tonic is C4 and interval is G5 the number of semitones is 19.

### is\_chromatic

bool - If the current interval is chromatic (True) or if it exists in the diatonic scale which key is tonic.

#### is descending

bool – If the interval has a descending direction, ie., has a lower pitch than the tonic.

#### diatonic\_index

int – If the interval is chromatic, this will be the nearest diatonic interval in the direction of the resolution (closest tonic.) From II to IV degrees, it is the diatonic interval before; from V to VII it is the diatonic interval after.

#### distance

dict – A dictionary which the distance from tonic to interval, for example, if tonic is C4 and interval is G5:

```
{
   'octaves': 1,
   'semitones': 7
}
```

#### data

*tuple* – A tuple representing the interval data in the form of (semitones, short\_name, long\_name), for example:

```
(19, 'P12', 'Perfect Twelfth')
```

#### Todo:

- Maybe we should refactor some of the attributes with a tuple (note, octave)
- Maybe remove *chromatic\_offset* in favor of *distance['semitones']'*

\_\_init\_\_ (mode, tonic, octave, n\_octaves=None, descending=None, valid\_intervals=None)
Inits the class and choses a random interval with the given args.

#### **Parameters**

- mode (str) Diatonic mode for the interval. (eg.: 'major' or 'minor')
- tonic (str) Tonic of the scale. (eg.: 'Bb')
- octave (str) Scientific octave of the scale (eg.: 4)
- **interval** (str) Not implemented. The interval.
- **chromatic** (bool) Can have chromatic notes? (eg.: F# in a key of C; default: false)
- n octaves (int) Maximum number os octaves (eg. 2)
- **descending** (bool) Is the interval descending? (default: false)
- **valid\_intervals** (*int*) A list with inervals valid for random choice, 1 is 1st, 2 is second etc.

class birdears.interval.DiatonicInterval (mode, tonic, octave, n\_octaves=None, descending=None, valid\_intervals=None)

 $Bases: \ \textit{birdears.interval.IntervalBase}$ 

Chooses a diatonic interval for the question.

#### tonic\_octave

int – Scientific octave for the tonic. For example, if the tonic is a 'C4' then tonic\_octave is 4.

### interval octave

int – Scientific octave for the interval. For example, if the interval is a 'G5' then tonic octave is 5.

#### chromatic offset

int – The offset in semitones inside one octave. Relative semitones to tonic.

### note\_and\_octave

str – Note and octave of the interval, for example, if the interval is G5 the note name is 'G5'.

### note\_name

str – The note name of the interval, for example, if the interval is G5 then the name is 'G'.

#### semitones

int – Semitones from tonic to octave. If tonic is C4 and interval is G5 the number of semitones is 19.

#### is chromatic

bool – If the current interval is chromatic (True) or if it exists in the diatonic scale which key is tonic.

#### is descending

bool – If the interval has a descending direction, ie., has a lower pitch than the tonic.

#### diatonic index

*int* – If the interval is chromatic, this will be the nearest diatonic interval in the direction of the resolution (closest tonic.) From II to IV degrees, it is the diatonic interval before; from V to VII it is the diatonic interval after.

#### distance

dict – A dictionary which the distance from tonic to interval, for example, if tonic is C4 and interval is G5:

```
{
   'octaves': 1,
   'semitones': 7
}
```

#### data

*tuple* – A tuple representing the interval data in the form of (semitones, short\_name, long\_name), for example:

```
(19, 'P12', 'Perfect Twelfth')
```

\_\_init\_\_ (mode, tonic, octave, n\_octaves=None, descending=None, valid\_intervals=None)
Inits the class and choses a random interval with the given args.

#### **Parameters**

- mode (str) Diatonic mode for the interval. (eg.: 'major' or 'minor')
- tonic (str) Tonic of the scale. (eg.: 'Bb')
- octave (str) Scientific octave of the scale (eg.: 4)
- n\_octaves (int) Maximum number os octaves (eg. 2)
- **descending** (bool) Is the interval descending? (default: false)
- **valid\_intervals** (*int*) A list with intervals (int) valid for random choice, 1 is 1st, 2 is second etc.

```
class birdears.interval.IntervalBase
```

```
Bases: object
```

```
__init__()
```

Base class for interval classes.

## return\_simple(keys)

This method returns a dict with only the values passed to keys.

# 7.4 birdears.logger module

This submodule exports *logger* to log events.

Logging messages which are less severe than lvl will be ignored:

Level	Numeric value
CRITICAL	50
ERROR	40
WARNING	30
INFO	20
DEBUG	10
NOTSET	0
Level	When it's used
DEBUG	Detailed information, typically of interest only when diagnosing problems.
INFO	Confirmation that things are working as expected.
WARNING	An indication that something unexpected happened, or indicative of some problem in the near future (e.g. 'disk space low').  The software is still working as expected.
ERROR	Due to a more serious problem, the software has not been able to perform some function.
CRITICAL	A serious error, indicating that the program itself may be unable to continue running.

```
birdears.logger.log_event (f, *args, **kwargs)
```

Decorator. Functions and method decorated with this decorator will have their signature logged when birdears is executed with *-debug* mode. Both function signature with their call values and their return will be logged.

# 7.5 birdears.prequestion module

This module implements pre-questions' progressions.

Pre questions are chord progressions or notes played before the question is played, so to affirmate the sound of the question's key.

For example a common cadence is chords I-IV-V-I from the diatonic scale, which in a key of C is CM-FM-GM-CM and in a key of A is AM-DM-EM-AM.

Pre-question methods should be decorated with *register\_prequestion\_method* decorator, so that they will be registered as a valid pre-question method.

```
class birdears.prequestion.PreQuestion (method, question)
    Bases: object
    __call__(*args, **kwargs)
        Calls the resolution method and pass arguments to it.
        Returns a birdears.Sequence object with the pre-question generated by the method.
    __init__(method, question)
        This class implements methods for different types of pre-question progressions.
```

- **method** (str) The method used in the pre question.
- question (obj) Question object from which to generate the
- sequence. (pre-question) -

birdears.prequestion.none(question, \*args, \*\*kwargs)

Pre-question method that return an empty sequence with no delay. :param question: Question object from which to generate the

pre-question sequence. (this is provided by the *Resolution* class when it is '\_\_call\_\_'ed)

```
birdears.prequestion.progression_i_iv_v_i (question, *args, **kwargs)
```

Pre-question method that play's a chord progression with triad chords built on the grades I, IV, V the I of the question key.

**Parameters question** (obj) – Question object from which to generate the pre-question sequence. (this is provided by the *Resolution* class when it is '\_\_call\_\_'ed)

```
birdears.prequestion.register_prequestion_method(f, *args, **kwargs)
```

Decorator for prequestion method functions.

Functions decorated with this decorator will be registered in the PREQUESTION\_METHODS global dict.

```
birdears.prequestion.tonic only (question, *args, **kwargs)
```

Pre-question method that only play's the question tonic note before the question.

**Parameters question** (obj) – Question object from which to generate the pre-question sequence. (this is provided by the *Resolution* class when it is '\_\_call\_\_'ed)

# 7.6 birdears.questionbase module

Bases: object

Base Class to be subclassed for Question classes.

This class implements attributes and routines to be used in Question subclasses.

\_\_init\_\_ (mode='major', tonic=None, octave=None, descending=None, chromatic=None, n\_octaves=None, valid\_intervals=None, user\_durations=None, prequestion\_method=None, resolution\_method=None, default\_durations=None, \*args, \*\*kwargs)
Inits the class.

- mode (str) A string represnting the mode of the question. Eg., 'major' or 'minor'
- **tonic** (str) A string representing the tonic of the question, eg.: 'C'; if omitted, it will be selected randomly.
- **octave** (*int*) A scienfic octave notation, for example, 4 for 'C4'; if not present, it will be randomly chosen.
- **descending** (bool) Is the question direction in descending, ie., intervals have lower pitch than the tonic.
- **chromatic** (bool) If the question can have (True) or not (False) chromatic intervals, ie., intervals not in the diatonic scale of tonic/mode.

- n\_octaves (int) Maximum numbr of octaves of the question.
- **valid\_intervals** (*list*) A list with intervals (int) valid for random choice, 1 is 1st, 2 is second etc. Eg. [1, 4, 5] to allow only tonics, fourths and fifths.
- user\_durations (dict) A string with 9 comma-separated int or float's to set the default duration for the notes played. The values are respectively for: pre-question duration (1st), pre-question delay (2nd), and pre-question pos-delay (3rd); question duration (4th), question delay (5th), and question pos-delay (6th); resolution duration (7th), resolution delay (8th), and resolution pos-delay (9th). duration is the duration in of the note in seconds; delay is the time to wait before playing the next note, and pos\_delay is the time to wait after all the notes of the respective sequence have been played. If any of the user durations is 'n, the default duration for the type of question will be used instead. Example:

```
"2,0.5,1,2,n,0,2.5,n,1"
```

- **prequestion\_method** (str) Method of playing a cadence or the exercise tonic before the question so to affirm the question musical tonic key to the ear. Valid ones are registered in the *birdears.prequestion.PREQUESION\_METHODS* global dict.
- **resolution\_method** (str) Method of playing the resolution of an exercise Valid ones are registered in the *birdears.resolution.RESOLUTION\_METHODS* global dict.
- **user\_durations** Dictionary with the default durations for each type of sequence. This is provided by the subclasses.

#### check\_question()

This method should be overwritten by the question subclasses.

```
get_valid_semitones()
```

Returns a list with valid semitones for question.

```
make_question()
```

This method should be overwritten by the question subclasses.

```
make resolution()
```

This method should be overwritten by the question subclasses.

```
play_question()
```

This method should be overwritten by the question subclasses.

```
birdears.questionbase.register_question_class(f, *args, **kwargs)
```

Decorator for question classes.

Classes decorated with this decorator will be registered in the QUESTION\_CLASSES global.

## 7.7 birdears.resolution module

```
class birdears.resolution.Resolution(method, question)
```

Bases: object

This class implements methods for different types of question resolutions.

A resolution is an answer to a question. It aims to create a mnemonic on how the inverval resvolver to the tonic.

```
___call___(*args, **kwargs)
```

Calls the resolution method and pass arguments to it.

Returns a *birdears.Sequence* object with the resolution generated by the.method.

```
___init___(method, question)
```

Inits the resolution class.

#### **Parameters**

- **method** (str) The method used in the resolution.
- question (obj) Question object from which to generate the
- sequence. (resolution) -

```
birdears.resolution.nearest_tonic(question)
```

Resolution method that resolve the intervals to their nearest tonics.

**Parameters question** (obj) – Question object from which to generate the resolution sequence. (this is provided by the *Prequestion* class when it is '\_\_call\_\_'ed)

```
birdears.resolution.register_resolution_method(f, *args, **kwargs)
```

Decorator for resolution method functions.

Functions decorated with this decorator will be registered in the RESOLUTION\_METHODS global dict.

```
birdears.resolution.repeat only (question)
```

Resolution method that only repeats the sequence elements with given durations.

**Parameters question** (obj) – Question object from which to generate the resolution sequence. (this is provided by the *Prequestion* class when it is '\_\_call\_\_'ed)

## 7.8 birdears.scale module

Bases: birdears.scale.ScaleBase

Builds a musical chromatic scale.

#### scale

array\_type - The array of notes representing the scale.

\_\_init\_\_ (tonic, octave=None, n\_octaves=None, descending=None, dont\_repeat\_tonic=None)
Returns a chromatic scale from tonic.

#### **Parameters**

- tonic (str) The note which the scale will be built upon.
- octave (int) The scientific octave the scale will be built upon.
- **n\_octaves** (*int*) The number of octaves the scale will contain.
- **descending** (bool) Whether the scale is descending.
- **dont\_repeat\_tonic** (bool) Whether to skip appending the last note (octave) to the scale.

```
get_triad (mode, index=0, degree=None)
```

Returns an array with notes from a scale's triad.

- **mode** (str) Mode of the scale (eg. 'major' or 'minor')
- index (int) Triad index (eg.: 0 for 1st degree triad.)

• **degree** (*int*) – Degree of the scale. If provided, overrides the *index* argument. (eg.: *1* for the 1st degree triad.)

**Returns** A list with three pitches (str), one for each note of the triad.

Bases: birdears.scale.ScaleBase

Builds a musical diatonic scale.

### scale

*array\_type* – The array of notes representing the scale.

Returns a diatonic scale from tonic and mode.

#### **Parameters**

- tonic (str) The note which the scale will be built upon.
- mode (str) The mode the scale will be built upon. ('major' or 'minor')
- **octave** (*int*) The scientific octave the scale will be built upon.
- **n\_octaves** (*int*) The number of octaves the scale will contain.
- **descending** (bool) Whether the scale is descending.
- **dont\_repeat\_tonic** (bool) Whether to skip appending the last note (octave) to the scale.

```
get_triad (index=0, degree=None)
```

Returns an array with notes from a scale's triad.

#### **Parameters**

- index (int) triad index (eg.: 0 for 1st degree triad.)
- **degree** (*int*) Degree of the scale. If provided, overrides the *index* argument. (eg.: *1* for the 1st degree triad.)

**Returns** An array with three pitches, one for each note of the triad.

```
class birdears.scale.ScaleBase
    Bases: object
```

# 7.9 birdears.sequence module

```
class birdears.sequence (elements=[], duration=2, delay=1.5, pos_delay=1)
    Bases: object
```

Register a Sequence of notes and/or chords.

#### elements

array\_type - List of notes (strings) ou chords (list of strings) in this Sequence.

#### append (elements)

Appends elements to Sequence.elements

**Parameters elements** (array\_type) – Elements to be appended to the class.

### async\_play (callback, end\_callback)

Plays the Sequence elements of notes and/or chords and wait for Sequence.pos\_delay seconds.

### extend(elements)

Extends Sequence.elements with elements.

**Parameters** elements (array\_type) – elements extend the class with.

## make\_chord\_progression(tonic, mode, degrees)

Appends triad chord(s) to the Sequence.

### **Parameters**

- tonic (str) Tonic note of the scale.
- **mode** (str) Mode of the scale from which build the triads upon.
- **degrees** (array\_type) List with integers represending the degrees of each triad.

play (callback=None, end\_callback=None)

### play\_element (index)

Plays element sequence.elements[index].

**CHAPTER** 

**EIGHT** 

## BIRDEARS.QUESTIONS PACKAGE

## 8.1 Submodules

# 8.2 birdears.questions.harmonicinterval module

```
class birdears.questions.harmonicinterval.HarmonicIntervalQuestion(mode='major',
                                                                                    tonic=None,
                                                                                    oc-
                                                                                    tave=None,
                                                                                    descend-
                                                                                    ing=None,
                                                                                    chro-
                                                                                    matic=None,
                                                                                    n\_octaves=None,
                                                                                    valid_intervals=None,
                                                                                    user_durations=None,
                                                                                    preques-
                                                                                    tion_method='none',
                                                                                    resolu-
                                                                                    tion method='nearest tonic',
                                                                                     *args,
                                                                                    **kwargs)
```

 $Bases: \ \textit{birdears.questionbase.QuestionBase}$ 

Implements a Harmonic Interval test.

```
__init__ (mode='major', tonic=None, octave=None, descending=None, chromatic=None, n_octaves=None, valid_intervals=None, user_durations=None, prequestion_method='none', resolution_method='nearest_tonic', *args, **kwargs')

Inits the class.
```

- mode (str) A string representing the mode of the question. Eg., 'major' or 'minor'
- **tonic** (*str*) A string representing the tonic of the question, eg.: 'C'; if omitted, it will be selected randomly.
- **octave** (*int*) A scienfic octave notation, for example, 4 for 'C4'; if not present, it will be randomly chosen.
- **descending** (bool) Is the question direction in descending, ie., intervals have lower pitch than the tonic.

- **chromatic** (bool) If the question can have (True) or not (False) chromatic intervals, ie., intervals not in the diatonic scale of tonic/mode.
- n\_octaves (int) Maximum number of octaves of the question.
- **valid\_intervals** (*list*) A list with intervals (int) valid for random choice, 1 is 1st, 2 is second etc. Eg. [1, 4, 5] to allow only tonics, fourths and fifths.
- user\_durations (str) A string with 9 comma-separated int or float's to set the default duration for the notes played. The values are respectively for: pre-question duration (1st), pre-question delay (2nd), and pre-question pos-delay (3rd); question duration (4th), question delay (5th), and question pos-delay (6th); resolution duration (7th), resolution delay (8th), and resolution pos-delay (9th). duration is the duration in of the note in seconds; delay is the time to wait before playing the next note, and pos\_delay is the time to wait after all the notes of the respective sequence have been played. If any of the user durations is 'n, the default duration for the type of question will be used instead. Example:

```
"2,0.5,1,2,n,0,2.5,n,1"
```

- **prequestion\_method** (str) Method of playing a cadence or the exercise tonic before the question so to affirm the question musical tonic key to the ear. Valid ones are registered in the *birdears.prequestion.PREQUESION\_METHODS* global dict.
- **resolution\_method** (str) Method of playing the resolution of an exercise. Valid ones are registered in the *birdears.resolution.RESOLUTION\_METHODS* global dict.

```
check_question (user_input_char)
```

Checks whether the given answer is correct.

```
make_pre_question (method)
make_question()
make_resolution (method)
play_question()
play_resolution()
```

# 8.3 birdears.questions.instrumentaldictation module

 $\textbf{class} \ \, \texttt{birdears.questions.instrumentaldictation.InstrumentalDictationQuestion} \, (\textit{mode='major'}, \\ \\ \textbf{and birdears.questions.instrumentaldictation.InstrumentalDictationQuestion} \, (\textit{mode='major'}, \\ \textbf{and birdears.questions.instrumentaldictation} \, (\textit{mode='major'}, \\ \textbf{and birdears.questions.question} \, (\textit{mode='major'}, \\ \textbf{and birdears.questions.q$ 

 $wait\_time=11$ , n repeats=1,  $max_intervals=3$ ,  $n \ notes=4$ , tonic=None, octave=None, descending=None, chromatic=None,  $n\_octaves=None,$ valid\_intervals=Noi user\_durations=No. prequestion\_method='prog res-0tion\_method='reped \*args, \*\*kwargs)

Bases: birdears.questionbase.QuestionBase

Implements an instrumental dictation test.

```
__init__ (mode='major', wait_time=11, n_repeats=1, max_intervals=3, n_notes=4, tonic=None, octave=None, descending=None, chromatic=None, n_octaves=None, valid_intervals=None, user_durations=None, prequestion_method='progression_i_iv_v_i', resolution_method='repeat_only', *args, **kwargs')

Inits the class.
```

- mode(str) A string representing the mode of the question. Eg., 'major' or 'minor'.
- wait\_time (float) Wait time in seconds for the next question or repeat.
- n\_repeats (int) Number of times the same dictation will be repeated before the end of the exercise.
- max\_intervals (int) The maximum number of random intervals the question will have.
- **n\_notes** (*int*) The number of notes the melodic dictation will have.
- **tonic** (str) A string representing the tonic of the question, eg.: 'C'; if omitted, it will be selected randomly.
- **octave** (*int*) A scienfic octave notation, for example, 4 for 'C4'; if not present, it will be randomly chosen.
- **descending** (bool) Is the question direction in descending, ie., intervals have lower pitch than the tonic.

- **chromatic** (bool) If the question can have (True) or not (False) chromatic intervals, ie., intervals not in the diatonic scale of tonic/mode.
- n\_octaves (int) Maximum number of octaves of the question.
- **valid\_intervals** (*list*) A list with intervals (int) valid for random choice, 1 is 1st, 2 is second etc. Eg. [1, 4, 5] to allow only tonics, fourths and fifths.
- user\_durations (str) A string with 9 comma-separated int or float's to set the default duration for the notes played. The values are respectively for: pre-question duration (1st), pre-question delay (2nd), and pre-question pos-delay (3rd); question duration (4th), question delay (5th), and question pos-delay (6th); resolution duration (7th), resolution delay (8th), and resolution pos-delay (9th). duration is the duration in of the note in seconds; delay is the time to wait before playing the next note, and pos\_delay is the time to wait after all the notes of the respective sequence have been played. If any of the user durations is 'n, the default duration for the type of question will be used instead. Example:

```
"2,0.5,1,2,n,0,2.5,n,1"
```

- **prequestion\_method** (str) Method of playing a cadence or the exercise tonic before the question so to affirm the question musical tonic key to the ear. Valid ones are registered in the *birdears.prequestion.PREQUESION\_METHODS* global dict.
- **resolution\_method** (str) Method of playing the resolution of an exercise. Valid ones are registered in the *birdears.resolution.RESOLUTION\_METHODS* global dict.

#### check\_question()

Checks whether the given answer is correct.

This currently doesn't applies to instrumental dictation questions.

```
make_pre_question (method)
make_question (phrase_semitones)
make_resolution (method)
play_question()
```

# 8.4 birdears.questions.melodicdictation module

class birdears.questions.melodicdictation.MelodicDictationQuestion(mode='major',

 $max_intervals=3$ ,  $n \ notes=4$ , tonic=None, tave=None, descending=None, chromatic=None, n octaves=None, *valid\_intervals=None*, user\_durations=None, prequestion\_method='progression\_i\_iv\_v\_ tion\_method='repeat\_only', \*args, \*\*kwargs)

Bases: birdears.questionbase.QuestionBase

Implements a melodic dictation test.

```
__init__ (mode='major', max_intervals=3, n_notes=4, tonic=None, octave=None, de-
scending=None, chromatic=None, n_octaves=None, valid_intervals=None,
user_durations=None, prequestion_method='progression_i_iv_v_i', resolu-
tion_method='repeat_only', *args, **kwargs')
Inits the class.
```

- mode (str) A string representing the mode of the question. Eg., 'major' or 'minor'.
- max\_intervals (int) The maximum number of random intervals the question will have.
- **n\_notes** (*int*) The number of notes the melodic dictation will have.
- **tonic** (str) A string representing the tonic of the question, eg.: 'C'; if omitted, it will be selected randomly.
- octave (int) A scienfic octave notation, for example, 4 for 'C4'; if not present, it will be randomly chosen.
- **descending** (bool) Is the question direction in descending, ie., intervals have lower pitch than the tonic.
- **chromatic** (bool) If the question can have (True) or not (False) chromatic intervals, ie., intervals not in the diatonic scale of tonic/mode.
- n\_octaves (int) Maximum number of octaves of the question.
- **valid\_intervals** (*list*) A list with intervals (int) valid for random choice, 1 is 1st, 2 is second etc. Eg. [1, 4, 5] to allow only tonics, fourths and fifths.
- **user\_durations** (str) A string with 9 comma-separated int or float's to set the default duration for the notes played. The values are respectively for: pre-question duration (1st), pre-question delay (2nd), and pre-question pos-delay (3rd); question duration

(4th), question delay (5th), and question pos-delay (6th); resolution duration (7th), resolution delay (8th), and resolution pos-delay (9th). duration is the duration in of the note in seconds; delay is the time to wait before playing the next note, and pos\_delay is the time to wait after all the notes of the respective sequence have been played. If any of the user durations is 'n, the default duration for the type of question will be used instead. Example:

```
"2,0.5,1,2,n,0,2.5,n,1"
```

- **prequestion\_method** (str) Method of playing a cadence or the exercise tonic before the question so to affirm the question musical tonic key to the ear. Valid ones are registered in the *birdears.prequestion.PREQUESION\_METHODS* global dict.
- **resolution\_method** (str) Method of playing the resolution of an exercise. Valid ones are registered in the *birdears.resolution.RESOLUTION\_METHODS* global dict.

# 8.5 birdears.questions.melodicinterval module

```
class birdears.questions.melodicinterval.MelodicIntervalQuestion(mode='major',
                                                                                  tonic=None,
                                                                                  oc-
                                                                                  tave=None,
                                                                                  descend-
                                                                                  ing=None,
                                                                                  chro-
                                                                                  matic=None,
                                                                                  n\_octaves=None,
                                                                                  valid intervals=None,
                                                                                  user_durations=None,
                                                                                  preques-
                                                                                  tion_method='tonic_only',
                                                                                  resolu-
                                                                                  tion_method='nearest_tonic',
                                                                                  *args,
                                                                                  **kwargs)
     Bases: birdears.questionbase.QuestionBase
     Implements a Melodic Interval test.
     check_question (user_input_char)
         Checks whether the given answer is correct.
     make_pre_question (method)
     make_question()
     make_resolution (method)
```

play\_question()
play\_resolution()

**CHAPTER** 

NINE

## **BIRDEARS.INTERFACES PACKAGE**

## 9.1 Submodules

## 9.2 birdears.interfaces.commandline module

birdears.interfaces.commandline.CommandLine(exercise, \*\*kwargs)
This function implements the birdears loop for command line.

#### **Parameters**

- **exercise** (str) The question name.
- \*\*kwargs (kwargs) FIXME: The kwargs can contain options for specific questions.

birdears.interfaces.commandline.center\_text (text, sep=True, nl=0)

This function returns input text centered according to terminal columns.

#### **Parameters**

- **text** (str) The string to be centered, it can have multiple lines.
- **sep** (bool) Add line separator after centered text (True) or not (False).
- **nl** (*int*) How many new lines to add after text.

birdears.interfaces.commandline.make\_input\_str(user\_input, keyboard\_index)

Makes a string representing intervals entered by the user.

This function is to be used by questions which takes more than one interval input as MelodicDictation, and formats the intervals already entered.

### Parameters

- **user\_input** (*array\_type*) The list of keyboard keys entered by user.
- **keyboard\_index** (array\_type) The keyboard mapping used by question.

birdears.interfaces.commandline.print\_instrumental(response)

Prints the formatted response for 'instrumental' exercise.

**Parameters** response (dict) – A response returned by question's check\_question()

birdears.interfaces.commandline.print\_question(question)

Prints the question to the user.

**Parameters** question (obj) – A Question class with the question to be printed.

birdears.interfaces.commandline.print\_response(response)

Prints the formatted response.

**Parameters** response (dict) – A response returned by question's check\_question()

# **PYTHON MODULE INDEX**

## b

```
birdears, 37
birdears.interfaces, 55
birdears.interfaces.commandline, 55
birdears.interval, 37
birdears.logger, 39
birdears.prequestion, 40
birdears.questionbase, 41
birdears.questions, 47
birdears.questions.harmonicinterval, 47
birdears.questions.instrumentaldictation, 49
birdears.questions.melodicdictation, 51
birdears.questions.melodicinterval, 52
birdears.resolution, 42
birdears.scale, 43
birdears.sequence, 44
```

58 Python Module Index

# **INDEX**

Symbolscall() (birdears.prequestion.PreQuestion method),21, 40call() (birdears.resolution.Resolution method), 23,42	birdears.questions.melodicdictation (module), 16, 51 birdears.questions.melodicinterval (module), 17, 52 birdears.resolution (module), 23, 42 birdears.scale (module), 24, 43 birdears.sequence (module), 25, 44
init() (birdears.interval.ChromaticInterval method), 19, 38	С
init() (birdears.interval.DiatonicInterval method), 20, 39	center_text() (in module bird- ears.interfaces.commandline), 11, 55
init() (birdears.interval.IntervalBase method), 20, 39 init() (birdears.prequestion.PreQuestion method),	check_question() (birdears.questionbase.QuestionBase method), 23, 42
21, 40	check_question() (bird-
init() (birdears.questionbase.QuestionBase method), 22, 41	ears.questions.harmonicinterval.HarmonicIntervalQuestion method), 13, 48
init() (birdears.questions.harmonicinterval.HarmonicIn	
method), 12, 47	ears.questions.instrumentaldictation.InstrumentalDictationQuestic
init() (birdears.questions.instrumentaldictation.Instrumenthod), 14, 49	check_question() (bird-
	ctationQuestions.melodicdictation.MelodicDictationQuestion
method), 16, 51	method), 17, 52
init() (birdears.resolution.Resolution method), 23,	check_question() (bird- ears.questions.melodicinterval.MelodicIntervalQuestion
init() (birdears.scale.ChromaticScale method), 24,	method), 17, 52
43	chromatic_offset (birdears.interval.ChromaticInterval at-
init() (birdears.scale.DiatonicScale method), 24, 44	tribute), 18, 37 chromatic_offset (birdears.interval.DiatonicInterval at-
A	tribute), 19, 38
append() (birdears.sequence.Sequence method), 25, 44	ChromaticInterval (class in birdears.interval), 18, 37
async_play() (birdears.sequence.Sequence method), 25,	ChromaticScale (class in birdears.scale), 24, 43
44	CommandLine() (in module bird- ears.interfaces.commandline), 11, 55
В	
birdears (module), 11, 37	D
birdears.interfaces (module), 11, 55	data (birdears.interval.ChromaticInterval attribute), 18, 38
birdears.interfaces.commandline (module), 11, 55	data (birdears.interval.DiatonicInterval attribute), 20, 39
birdears.interval (module), 18, 37	diatonic_index (birdears.interval.ChromaticInterval attribute), 18, 37
birdears progression (module), 20, 39	diatonic_index (birdears.interval.DiatonicInterval at-
birdears.prequestion (module), 21, 40 birdears.questionbase (module), 22, 41	tribute), 19, 39
birdears.questionbase (module), 22, 41 birdears.questions (module), 12, 47	DiatonicInterval (class in birdears.interval), 19, 38
birdears.questions (module), 12, 47	DiatonicScale (class in birdears.scale), 24, 44
hirdears questions instrumental dictation (module) 14 49	

distance (birdears.interval.ChromaticInterval attribute),	make_pre_question() (bird-
18, 37 distance (birdears.interval.DiatonicInterval attribute), 19,	ears.questions.melodicinterval.MelodicIntervalQuestion method), 17, 52
39	make_question() (birdears.questionbase.QuestionBase
E	method), 23, 42
	make_question() (birdears.questions.harmonicinterval.HarmonicIntervalQu
elements (birdears.sequence.Sequence attribute), 25, 44 extend() (birdears.sequence.Sequence method), 25, 45	method), 13, 48 make_question() (birdears.questions.instrumentaldictation.InstrumentalDict
	method), 15, 50
G	$make\_question() \\ (birdears.questions.melodic dictation. Melodic Dictation Question) \\ (birdears.questions.melodic dictation) \\ (birdears.question) \\ (b$
get_triad() (birdears.scale.ChromaticScale method), 24,	method), 17, 52
43	make_question() (birdears.questions.melodicinterval.MelodicIntervalQuestimethod), 17, 52
get_triad() (birdears.scale.DiatonicScale method), 25, 44 get_valid_semitones() (bird-	make_resolution() (birdears.questionbase.QuestionBase
ears.questionbase.QuestionBase method),	method), 23, 42
23, 42	make_resolution() (bird-
Н	ears.questions.harmonicinterval.HarmonicIntervalQuestion method), 13, 48
	make_resolution() (bird-
HarmonicIntervalQuestion (class in bird- ears.questions.harmonicinterval), 12, 47	ears.questions.instrumentaldictation.InstrumentalDictationQuestimethod), 15, 50
	make_resolution() (bird-
InstrumentalDictationQuestion (class in bird-	ears. questions. melodic dictation. Melodic Dictation Question
ears.questions.instrumentaldictation), 14,	method), 17, 52
49	make_resolution() (bird- ears.questions.melodicinterval.MelodicIntervalQuestion
IntervalBase (class in birdears.interval), 20, 39	method), 17, 52
is_chromatic (birdears.interval.ChromaticInterval attribute), 18, 37	MelodicDictationQuestion (class in bird- ears.questions.melodicdictation), 16, 51
is_chromatic (birdears.interval.DiatonicInterval at-	MelodicIntervalQuestion (class in bird-
tribute), 19, 39 is_descending (birdears.interval.ChromaticInterval	ears.questions.melodicinterval), 17, 52
attribute), 18, 37	N
is_descending (birdears.interval.DiatonicInterval at-	
tribute), 19, 39	nearest_tonic() (in module birdears.resolution), 23, 43 none() (in module birdears.prequestion), 21, 40
I	note_and_octave (birdears.interval.ChromaticInterval at-
log avent() (in module hirdeers logger) 21, 40	tribute), 18, 37
log_event() (in module birdears.logger), 21, 40	note_and_octave (birdears.interval.DiatonicInterval at-
M	tribute), 19, 38
make_chord_progression() (birdears.sequence.Sequence	note_name (birdears.interval.ChromaticInterval attribute), 18, 37
method), 25, 45	note_name (birdears.interval.DiatonicInterval attribute),
make_input_str() (in module bird-	19, 38
ears.interfaces.commandline), 11, 55 make_pre_question() (bird-	P
ears.questions.harmonicinterval.HarmonicInterva	ol Question play() (birdears.sequence.Sequence method), 26, 45
method), 13, 48	play_element() (birdears.sequence.Sequence method), 20, 43
make_pre_question() (bird-	26.45
ears.questions.instrumentaldictation.Instrumental	DictationQuestion (birdears.questionbase.QuestionBase
method), 15, 50 make_pre_question() (bird-	method), 23, 42
ears.questions.melodicdictation.MelodicDictation	play_question() (birdears.questions.harmonicinterval.HarmonicIntervalQuenQuestion method), 13, 48
method), 17, 52	play_question() (birdears.questions.instrumentaldictation.InstrumentalDicta
	method), 15, 50

60 Index

```
play question() (birdears.questions.melodicdictation.MelodicDictationOuestion
          method), 17, 52
play_question() (birdears.questions.melodicinterval.MelodicIntervalQuestion
          method), 17, 52
play_resolution()
                                                   (bird-
         ears.questions.harmonicinterval.HarmonicIntervalQuestion
         method), 13, 48
play resolution()
                                                   (bird-
         ears.questions.melodicdictation.MelodicDictationQuestion
         method), 17, 52
play_resolution()
                                                   (bird-
         ears. questions. melodic interval. Melodic Interval Question\\
         method), 18, 53
PreQuestion (class in birdears.prequestion), 21, 40
print_instrumental()
                                     module
                           (in
                                                    bird-
         ears.interfaces.commandline), 11, 55
print_question()
                        (in
                                    module
                                                    bird-
         ears.interfaces.commandline), 11, 55
print_response()
                        (in
                                    module
                                                    bird-
         ears.interfaces.commandline), 12, 55
progression_i_iv_v_i() (in module birdears.prequestion),
         21, 41
Q
QuestionBase (class in birdears.questionbase), 22, 41
R
register prequestion method()
                                  (in
                                         module
                                                    bird-
         ears.prequestion), 21, 41
register_question_class()
                                       module
                                                    bird-
                              (in
         ears.questionbase), 23, 42
register_resolution_method()
                                        module
                                                    bird-
                                 (in
         ears.resolution), 24, 43
repeat only() (in module birdears.resolution), 24, 43
Resolution (class in birdears.resolution), 23, 42
return simple() (birdears.interval.IntervalBase method),
          20, 39
S
scale (birdears.scale.ChromaticScale attribute), 24, 43
scale (birdears.scale.DiatonicScale attribute), 24, 44
ScaleBase (class in birdears.scale), 25, 44
semitones (birdears.interval.ChromaticInterval attribute),
          18.37
semitones (birdears.interval.DiatonicInterval attribute),
          19, 38
Sequence (class in birdears.sequence), 25, 44
Т
tonic octave (birdears.interval.ChromaticInterval at-
          tribute), 18, 37
tonic_octave (birdears.interval.DiatonicInterval attribute),
tonic only() (in module birdears.prequestion), 22, 41
```

Index 61