birdears Documentation

Release 0.1.1

lacchus Mercurius

TABLE OF 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		g birdears	-
	2.1	What is Functional Ear Training	5
	2.2	The method	5
	2.3	birdears modes	5
	2.4	basic usage	5
		2.4.1 melodic	6
		2.4.2 harmonic	7
		2.4.3 dictation	8
		2.4.4 instrumental	ç
	2.5	Loading from config/preset files	1(
			1(
			1(
2			
3		1	11
	3.1		11
			11
			12
	3.2		18
	3.3		18
	3.4		21
	3.5	1 1	22
	3.6	birdears.questionbase module	23
	3.7	birdears.resolution module	24
	3.8	birdears.scale module	25
	3.9	birdears.sequence module	26
4	Incto	alling birdears	29
7	4.1	8	29
	4.1		29 29
	4.2		
	4.2	E Company of the Comp	29
		4.2.1 In-depth installation	29
5	Using	g birdears	31
	5.1		3 1

	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	31 31 32 33 34 35 36 36 36
6	API		37
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	39 39 39 42 42 43 45 46 47
8	8.1 8.2 8.3 8.4 8.5	Submodules birdears.questions.harmonicinterval module birdears.questions.melodicdictation module birdears.questions.melodicdictation module birdears.questions.melodicinterval module	49 49 51 53 54
9	birde 9.1 9.2	Submodules	57 57 57
Ру	thon N	Module Index	59
In	dex		61

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.

TABLE OF CONTENTS 1

2 TABLE OF 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.

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"
```

play_question()

- **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.

```
__module__ = 'birdears.questions.instrumentaldictation'
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)
```

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

```
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.
     ___init___(mode='major',
                                tonic=None,
                                               octave=None,
                                                              descending=None,
                                                                                  chromatic=None,
                n octaves=None,
                                      valid intervals=None,
                                                                user durations=None,
                 tion_method='tonic_only', resolution_method='nearest_tonic', *args, **kwargs)
          Inits the class.
```

Parameters

3.1. Subpackages 17

- 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.

3.2 Submodules

3.3 birdears.interval module

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.

```
module = 'birdears.interval'
```

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.

```
__module__ = 'birdears.interval'

class birdears.interval.IntervalBase
    Bases: object

    __dict__ = mappingproxy({'__module__': 'birdears.interval', '__init__': <function In
    __init__()
        Base class for interval classes.

    __module__ = 'birdears.interval'
    __weakref__
        list of weak references to the object (if defined)
    return_simple (keys)</pre>
```

3.4 birdears.logger module

This submodule exports *logger* to log events.

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

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

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
INFO	diagnosing problems. 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.
     __dict__ = mappingproxy({'__module__': 'birdears.prequestion', '__init__': <function
      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 (ob i) – Question object from which to generate the
                  • sequence. (pre-question) -
     __module__ = 'birdears.prequestion'
      _weakref__
          list of weak references to the object (if defined)
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 $(ob\ j)$ – 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.

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:

'\n Base

```
"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.

```
__module__ = 'birdears.questionbase'
__weakref__
list of weak references to the object (if defined)

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.
```

```
\verb|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.
```

__dict__ = mappingproxy({'__module__': 'birdears.resolution', '__doc__': 'This class

```
\_\_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) -

__module__ = 'birdears.resolution'

__weakref__
list of weak references to the object (if defined)

birdears.resolution.nearest_tonic (question)

Resolution method that resolve the intervals to their nearest tonics.

Parameters question (obj) - Question object from which
```

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.

```
__module__ = 'birdears.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.

```
class birdears.scale.DiatonicScale (tonic, mode=None, octave=None, n_octaves=None, de-
                                           scending=None, dont_repeat_tonic=None)
     Bases: birdears.scale.ScaleBase
     Builds a musical diatonic scale.
     scale
```

array_type – The array of notes representing the scale.

```
descending=None,
         mode=None,
                          octave=None,
                                           n\_octaves=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.

```
__module__ = 'birdears.scale'
get_triad (index=0, degree=None)
```

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

list of weak references to the object (if defined)

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
    __dict__ = mappingproxy({'__module__': 'birdears.scale', '__init__': <function Scale
    init ()
    __module__ = 'birdears.scale'
     _weakref__
```

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.

```
__dict__ = mappingproxy({'__module__': 'birdears.sequence', '__doc__': 'Register a S __init__ (elements=[], duration=2, delay=1.5, pos_delay=1)
```

Inits the Sequence with an array and sets the default times for playing / pausing the elements.

Parameters

- **elements** (array_type) List of elements in this sequence. (notes or chords)
- **duration** (*float*) Default playing time for each element in the sequence.
- **delay** (float) Default waiting time to play the next element in the sequence.
- pos_delay (float) Waiting time after playing the last element in the sequence.

```
__len__()
__module__ = 'birdears.sequence'
__weakref__
list of weak references to the object (if defined)
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:|
f2g 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 33

```
-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 35

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

38 Chapter 6. API

BIRDEARS PACKAGE

BirdEars provides facilities to musical ear training exercises.

7.1 Subpackages

7.2 Submodules

7.3 birdears.interval module

class birdears.interval.**ChromaticInterval** (*mode*, *tonic*, *octave*, *n_octaves=None*, *descend-ing=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; 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.

```
__module__ = 'birdears.interval'
```

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.

- 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.

```
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
          40
ERROR
WARNING
           3.0
TNFO
           20
DEBUG
           10
NOTSET
Level
           When it's used
DEBUG
           Detailed information, typically of interest only when
               diagnosing problems.
           Confirmation that things are working as expected.
TNFO
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.
   __dict__ = mappingproxy({'__module__': 'birdears.prequestion', '__init__': <function</pre>
```

```
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) -
```

```
__module__ = 'birdears.prequestion'
```

weakref

list of weak references to the object (if defined)

```
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

```
 \begin{array}{c} \textbf{class} \text{ birdears.questionbase.} \textbf{QuestionBase} (\textit{mode='major'}, & \textit{tonic=None}, & \textit{octave=None}, \\ \textit{descending=None}, & \textit{chromatic=None}, \\ \textit{n\_octaves=None}, & \textit{valid\_intervals=None}, \\ \textit{user\_durations=None}, & \textit{prequestion\_method=None}, \\ \textit{default\_durations=None}, & \textit{*args}, **kwargs) \\ \end{array}
```

Bases: object

Base Class to be subclassed for Question classes.

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

```
__dict__ = mappingproxy({'__module__': 'birdears.questionbase', '__doc__': '\n Base
```

__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.

```
__module__ = 'birdears.questionbase'
__weakref__
list of weak references to the object (if defined)

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.

```
__dict__ = mappingproxy({'__module__': 'birdears.resolution', '__doc__': 'This class
__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) -

```
__module__ = 'birdears.resolution'
weakref
```

list of weak references to the object (if defined)

```
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

```
class birdears.scale.ChromaticScale(tonic,
                                                         octave=None,
                                                                         n\_octaves=None,
                                                                                            descend-
                                                ing=None, dont_repeat_tonic=None)
     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.
       _module__ = 'birdears.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.

- 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.
     __module__ = 'birdears.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
     __dict__ = mappingproxy({'__module__': 'birdears.scale', '__init__': <function Scale
     init ()
     __module__ = 'birdears.scale'
     weakref
          list of weak references to the object (if defined)
7.9 birdears.sequence module
class birdears.sequence.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.
     __dict__ = mappingproxy({'__module__': 'birdears.sequence', '__doc__': 'Register a S
     ___init__ (elements=[], duration=2, delay=1.5, pos_delay=1)
          Inits the Sequence with an array and sets the default times for playing / pausing the elements.
              Parameters
                  • elements (array_type) – List of elements in this sequence. (notes or chords)
                  • duration (float) – Default playing time for each element in the sequence.
                  • delay (float) – Default waiting time to play the next element in the sequence.
                  • pos_delay (float) - Waiting time after playing the last element in the sequence.
     __len__()
      __module__ = 'birdears.sequence'
          list of weak references to the object (if defined)
     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)
```

 $Bases: \verb|birdears.question| base. Question Base|$

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.

8.3 birdears.questions.instrumentaldictation module

 $\textbf{class} \ \, \texttt{birdears.questions.instrumental} \\ \textbf{dictation.InstrumentalDictationQuestion} \ \, (\textit{mode='major'}, \\ \textbf{dictationQuestion}) \\ \textbf{dictationQuestion} \ \, (\textit{mode='major'}, \\ \textbf{dictationQuestion}) \\ \textbf{dictationQuestion} \ \, (\textit{mode='major'}, \\ \textbf{dictationQuestion}) \\ \textbf{dictationQuestion} \ \, (\textit{mode='major'}, \\ \textbf{dictationQuestion}) \\ \textbf{dictationQuestion}) \\ \textbf{dictationQuestion} \ \, (\textit{mode='major'}, \\ \textbf{dictationQuestion}) \\ \textbf{dictationQuestion}) \\ \textbf{dictationQuestion} \ \, (\textit{mode='major'}, \\ \textbf{dictationQuestion}) \\ \textbf{dictationQuestion}) \\ \textbf{dictationQuestion} \ \, (\textit{mode='major'}, \\ \textbf{dictationQuestion}) \\ \textbf{dictationQuestion}) \\ \textbf{dictationQuestion} \ \, (\textit{mode='major'}, \\ \textbf{dictationQuestion}) \\ \textbf{dictationQuestion}) \\ \textbf{dictationQuestion} \ \, (\textit{mode='major'}, \\ \textbf{dictationQuestion}) \\ \textbf{dictationQuestion}) \\ \textbf{dictationQuestion} \\ \textbf{dictationQuestion}) \\ \textbf{dictationQuestionQuestion} \\ \textbf{dictationQuestion} \\ \textbf{dictationQuestionQuestion}) \\ \textbf{dictationQuestionQuestionQuestion} \\ \textbf{dictationQuestionQuestionQuestionQuestion} \\ \textbf{dictationQuestio$

 $wait_time=11$, 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.

```
__module__ = 'birdears.questions.instrumentaldictation'

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.
       _init___(mode='major',
                                 tonic=None,
                                                              descending=None,
                                               octave=None,
                                                                                  chromatic=None,
                 n octaves=None,
                                      valid intervals=None,
                                                                user_durations=None,
                                                                                          preques-
                 tion_method='tonic_only', 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.

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.

 $\verb|birdears.interfaces.commandline.center_text| (\textit{text}, \textit{sep=True}, \textit{nl} = 0)$

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()

 $\verb|birdears.interfaces.commandline.print_question| (question)$

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, 39
birdears.interfaces, 57
birdears.interfaces.commandline, 57
birdears.interval, 39
birdears.logger, 42
birdears.prequestion, 42
birdears.questionbase, 43
birdears.questions, 49
birdears.questions.harmonicinterval, 49
birdears.questions.instrumentaldictation, 51
birdears.questions.melodicdictation, 53
birdears.questions.melodicinterval, 54
birdears.resolution, 45
birdears.scale, 46
birdears.sequence, 47
```

60 Python Module Index

INDEX

Symbols	module (birdears.interval.IntervalBase attribute), 21,
call() (birdears.prequestion.PreQuestion method),	41
22, 42	module (birdears.prequestion.PreQuestion attribute),
call() (birdears.resolution.Resolution method), 24,	22, 43
45	module (birdears.questionbase.QuestionBase at-
dict (birdears.interval.IntervalBase attribute), 21, 41	tribute), 24, 44
dict (birdears.prequestion.PreQuestion attribute), 22,	module (birdears.questions.harmonicinterval.HarmonicIntervalQuestio
42	attribute), 13, 50
dict (birdears.questionbase.QuestionBase attribute),	module (birdears.questions.instrumentaldictation.InstrumentalDictation
23, 43	attribute), 15, 52
dict (birdears.resolution.Resolution attribute), 24, 45	module (birdears.questions.melodicdictation.MelodicDictationQuestion
dict (birdears.scale.ScaleBase attribute), 26, 47	attribute), 17, 54
dict (birdears.sequence.Sequence attribute), 27, 47	module (birdears.questions.melodicinterval.MelodicIntervalQuestion
init() (birdears.interval.ChromaticInterval method),	attribute), 18, 55
19, 40	module (birdears.resolution.Resolution attribute),
init() (birdears.interval.DiatonicInterval method),	25, 45
20, 41	module (birdears.scale.ChromaticScale attribute),
init() (birdears.interval.IntervalBase method), 21, 41	25, 46
init() (birdears.prequestion.PreQuestion method),	module (birdears.scale.DiatonicScale attribute), 26,
22, 43	47
init() (birdears.questionbase.QuestionBase method),	module (birdears.scale.ScaleBase attribute), 26, 47
23, 43	module (birdears.sequence.Sequence attribute), 27,
init() (birdears.questions.harmonicinterval.HarmonicIn	ntervalQuestion
method), 12, 49	weakref (birdears.interval.IntervalBase attribute),
$\underline{\hspace{0.3cm}} init\underline{\hspace{0.3cm}} () (birdears.questions.instrumental dictation.Instrumental $	nentalDictationQuestion
method), 14, 51	weakref (birdears.prequestion.PreQuestion at-
init() (birdears.questions.melodicdictation.MelodicDic	etationQuestion ¹¹⁰ , 22, 43
method), 16, 53	weakiei (biidears.questionbase.Questionbase
init() (birdears.questions.melodicinterval.MelodicInter	rvalQuestionute), 24, 44 weakref (birdears.resolution.Resolution attribute),
method), 17, 54	weakiei (birdears.resolution.Resolution attribute), 25, 45
init() (birdears.resolution.Resolution method), 24,	weakref (birdears.scale.ScaleBase attribute), 26, 47
45	weakref (birdears.sequence.Sequence attribute), 20, 47weakref (birdears.sequence.Sequence attribute), 27,
init() (birdears.scale.ChromaticScale method), 25,	weakiei (birdears.sequence.sequence attribute), 27,
46	47
init() (birdears.scale.DiatonicScale method), 26, 46	A
init() (birdears.scale.ScaleBase method), 26, 47	
init() (birdears.sequence.Sequence method), 27, 47	append() (birdears.sequence.Sequence method), 27, 47
len() (birdears.sequence.Sequence method), 27, 47	async_play() (birdears.sequence.Sequence method), 27,
module (birdears.interval.ChromaticInterval at-	48
tribute), 20, 40	В
module (birdears.interval.DiatonicInterval attribute),	
21, 41	birdears (module), 11, 39

birdears.interfaces (module), 11, 57	E
birdears.interfaces.commandline (module), 11, 57	elements (birdears.sequence.Sequence attribute), 26, 47
birdears.interval (module), 18, 39	extend() (birdears.sequence.Sequence method), 27, 48
birdears.logger (module), 21, 42	
birdears.prequestion (module), 22, 42	G
birdears.questionbase (module), 23, 43	get_triad() (birdears.scale.ChromaticScale method), 25,
birdears.questions (module), 12, 49	46
birdears.questions.harmonicinterval (module), 12, 49	get_triad() (birdears.scale.DiatonicScale method), 26, 47
birdears questions instrumental dictation (module), 14, 51	get_valid_semitones() (bird-
birdears.questions.melodicdictation (module), 16, 53 birdears.questions.melodicinterval (module), 17, 54	ears.questionbase.QuestionBase method),
birdears.resolution (module), 24, 45	24, 44
birdears.scale (module), 25, 46	Н
birdears.sequence (module), 26, 47	
(modulo), 20, 17	HarmonicIntervalQuestion (class in bird-
C	ears.questions.harmonicinterval), 12, 49
center_text() (in module bird-	1
ears.interfaces.commandline), 11, 57	
check_question() (birdears.questionbase.QuestionBase	InstrumentalDictationQuestion (class in bird-
method), 24, 44	ears.questions.instrumentaldictation), 14,
check_question() (bird-	51
ears.questions.harmonicinterval.HarmonicInterva	intervalBase (class in birdears.interval), 21, 41 ilQuestion is_chromatic (birdears.interval.ChromaticInterval at-
method), 13, 50	tribute), 19, 39
check_question() (bird-	
ears.questions.instrumentaldictation.Instrumental	is chromatic (birdears.interval.DiatonicInterval at- DictationQuestion tribute), 20, 41
method), 15, 52	is_descending (birdears.interval.ChromaticInterval
check_question() (bird-	ottributa) 10 20
ears.questions.melodicdictation.MelodicDictation	is_descending (birdears.interval.DiatonicInterval at-
method), 17, 54 check_question() (bird-	tribute), 20, 41
ears.questions.melodicinterval.MelodicIntervalQ	uestion
method), 18, 55	upsiton
chromatic_offset (birdears.interval.ChromaticInterval at-	log_event() (in module birdears.logger), 22, 42
tribute), 19, 39	NA.
chromatic_offset (birdears.interval.DiatonicInterval at-	M
tribute), 20, 40	make_chord_progression() (birdears.sequence.Sequence
ChromaticInterval (class in birdears.interval), 18, 39	method), 27, 48
ChromaticScale (class in birdears.scale), 25, 46	make_input_str() (in module bird-
CommandLine() (in module bird-	ears.interfaces.commandline), 11, 57
ears.interfaces.commandline), 11, 57	make_pre_question() (bird-
D	ears.questions.harmonicinterval.HarmonicIntervalQuestion
	method), 13, 50 make_pre_question() (bird-
data (birdears.interval.ChromaticInterval attribute), 19, 40	ears.questions.instrumentaldictation.InstrumentalDictationQuestion
data (birdears.interval.DiatonicInterval attribute), 20, 41	method), 15, 52
diatonic_index (birdears.interval.ChromaticInterval at-	make_pre_question() (bird-
tribute), 19, 39	ears.questions.melodicdictation.MelodicDictationQuestion
diatonic_index (birdears.interval.DiatonicInterval attribute), 20, 41	method), 17, 54
DiatonicInterval (class in birdears.interval), 20, 40	make_pre_question() (bird-
DiatonicScale (class in birdears.scale), 26, 46	ears.questions.melodicinterval.MelodicIntervalQuestion
distance (birdears.interval.ChromaticInterval attribute),	method), 18, 55
19, 39	make_question() (birdears.questionbase.QuestionBase
distance (birdears.interval.DiatonicInterval attribute), 20,	method), 24, 44
/1	

62 Index

	mphaisInterventQunt3tion (bird-
method), 13, 50	ears.questions.harmonicinterval.HarmonicIntervalQuestion
make_question() (birdears.questions.instrumentaldictation.	
method), 15, 52	play_resolution() (bird-
make_question() (birdears.questions.meiodicdictation.Meio method), 17, 54	odicDictation Question method), 17, 54
make_question() (birdears.questions.melodicinterval.Melodicint	
method), 18, 55	ears.questions.melodicinterval.MelodicIntervalQuestion
make_resolution() (birdears.questionbase.QuestionBase	method), 18, 55
method), 24, 44	PreQuestion (class in birdears.prequestion), 22, 42
make_resolution() (bird-	print_instrumental() (in module bird-
ears.questions.harmonicinterval.HarmonicInterva	
<u> •</u>	
method), 13, 50 make resolution() (bird-	print_question() (in module bird-
	ears.interfaces.commandline), 11, 57
ears.questions.instrumentaldictation.Instrumenta	
method), 15, 52	ears.interfaces.commandline), 12, 57
	progression_i_iv_v_i() (in module birdears.prequestion),
ears.questions.melodicdictation.MelodicDictation	nQuestion 22, 43
method), 17, 54	\circ
make_resolution() (bird-	Q
ears.questions.melodicinterval.MelodicIntervalQ method), 18, 55	uctionionBase (class in birdears.questionbase), 23, 43
MelodicDictationQuestion (class in bird-	R
ears.questions.melodicdictation), 16, 53	
MelodicIntervalQuestion (class in bird-	register_prequestion_method() (in module bird-
ears.questions.melodicinterval), 17, 54	ears.prequestion), 22, 43
cursiquestions.merodientervar), 17, 31	register_question_class() (in module bird-
N	ears.questionbase), 24, 45
• •	
named topic() (in modulo bindons posterior) 25 45	register_resolution_method() (in module bird-
nearest_tonic() (in module birdears.resolution), 25, 45	ears.resolution), 25, 45
none() (in module birdears.prequestion), 22, 43	ears.resolution), 25, 45 repeat_only() (in module birdears.resolution), 25, 45
none() (in module birdears.prequestion), 22, 43 note_and_octave (birdears.interval.ChromaticInterval at-	ears.resolution), 25, 45 repeat_only() (in module birdears.resolution), 25, 45 Resolution (class in birdears.resolution), 24, 45
none() (in module birdears.prequestion), 22, 43 note_and_octave (birdears.interval.ChromaticInterval at- tribute), 19, 39	ears.resolution), 25, 45 repeat_only() (in module birdears.resolution), 25, 45 Resolution (class in birdears.resolution), 24, 45 return_simple() (birdears.interval.IntervalBase method),
none() (in module birdears.prequestion), 22, 43 note_and_octave (birdears.interval.ChromaticInterval at- tribute), 19, 39 note_and_octave (birdears.interval.DiatonicInterval at-	ears.resolution), 25, 45 repeat_only() (in module birdears.resolution), 25, 45 Resolution (class in birdears.resolution), 24, 45
none() (in module birdears.prequestion), 22, 43 note_and_octave (birdears.interval.ChromaticInterval at- tribute), 19, 39 note_and_octave (birdears.interval.DiatonicInterval at- tribute), 20, 40	ears.resolution), 25, 45 repeat_only() (in module birdears.resolution), 25, 45 Resolution (class in birdears.resolution), 24, 45 return_simple() (birdears.interval.IntervalBase method), 21, 41
none() (in module birdears.prequestion), 22, 43 note_and_octave (birdears.interval.ChromaticInterval attribute), 19, 39 note_and_octave (birdears.interval.DiatonicInterval attribute), 20, 40 note_name (birdears.interval.ChromaticInterval at-	ears.resolution), 25, 45 repeat_only() (in module birdears.resolution), 25, 45 Resolution (class in birdears.resolution), 24, 45 return_simple() (birdears.interval.IntervalBase method),
none() (in module birdears.prequestion), 22, 43 note_and_octave (birdears.interval.ChromaticInterval attribute), 19, 39 note_and_octave (birdears.interval.DiatonicInterval attribute), 20, 40 note_name (birdears.interval.ChromaticInterval attribute), 19, 39	ears.resolution), 25, 45 repeat_only() (in module birdears.resolution), 25, 45 Resolution (class in birdears.resolution), 24, 45 return_simple() (birdears.interval.IntervalBase method),
none() (in module birdears.prequestion), 22, 43 note_and_octave (birdears.interval.ChromaticInterval attribute), 19, 39 note_and_octave (birdears.interval.DiatonicInterval attribute), 20, 40 note_name (birdears.interval.ChromaticInterval attribute), 19, 39 note_name (birdears.interval.DiatonicInterval attribute),	ears.resolution), 25, 45 repeat_only() (in module birdears.resolution), 25, 45 Resolution (class in birdears.resolution), 24, 45 return_simple() (birdears.interval.IntervalBase method), 21, 41 S
none() (in module birdears.prequestion), 22, 43 note_and_octave (birdears.interval.ChromaticInterval attribute), 19, 39 note_and_octave (birdears.interval.DiatonicInterval attribute), 20, 40 note_name (birdears.interval.ChromaticInterval attribute), 19, 39	ears.resolution), 25, 45 repeat_only() (in module birdears.resolution), 25, 45 Resolution (class in birdears.resolution), 24, 45 return_simple() (birdears.interval.IntervalBase method), 21, 41 S scale (birdears.scale.ChromaticScale attribute), 25, 46
none() (in module birdears.prequestion), 22, 43 note_and_octave (birdears.interval.ChromaticInterval attribute), 19, 39 note_and_octave (birdears.interval.DiatonicInterval attribute), 20, 40 note_name (birdears.interval.ChromaticInterval attribute), 19, 39 note_name (birdears.interval.DiatonicInterval attribute), 20, 40	ears.resolution), 25, 45 repeat_only() (in module birdears.resolution), 25, 45 Resolution (class in birdears.resolution), 24, 45 return_simple() (birdears.interval.IntervalBase method),
none() (in module birdears.prequestion), 22, 43 note_and_octave (birdears.interval.ChromaticInterval attribute), 19, 39 note_and_octave (birdears.interval.DiatonicInterval attribute), 20, 40 note_name (birdears.interval.ChromaticInterval attribute), 19, 39 note_name (birdears.interval.DiatonicInterval attribute), 20, 40 P	ears.resolution), 25, 45 repeat_only() (in module birdears.resolution), 25, 45 Resolution (class in birdears.resolution), 24, 45 return_simple() (birdears.interval.IntervalBase method),
none() (in module birdears.prequestion), 22, 43 note_and_octave (birdears.interval.ChromaticInterval attribute), 19, 39 note_and_octave (birdears.interval.DiatonicInterval attribute), 20, 40 note_name (birdears.interval.ChromaticInterval attribute), 19, 39 note_name (birdears.interval.DiatonicInterval attribute), 20, 40 P play() (birdears.sequence.Sequence method), 27, 48	ears.resolution), 25, 45 repeat_only() (in module birdears.resolution), 25, 45 Resolution (class in birdears.resolution), 24, 45 return_simple() (birdears.interval.IntervalBase method),
none() (in module birdears.prequestion), 22, 43 note_and_octave (birdears.interval.ChromaticInterval attribute), 19, 39 note_and_octave (birdears.interval.DiatonicInterval attribute), 20, 40 note_name (birdears.interval.ChromaticInterval attribute), 19, 39 note_name (birdears.interval.DiatonicInterval attribute), 20, 40 P play() (birdears.sequence.Sequence method), 27, 48 play_element() (birdears.sequence.Sequence method),	ears.resolution), 25, 45 repeat_only() (in module birdears.resolution), 25, 45 Resolution (class in birdears.resolution), 24, 45 return_simple() (birdears.interval.IntervalBase method),
none() (in module birdears.prequestion), 22, 43 note_and_octave (birdears.interval.ChromaticInterval attribute), 19, 39 note_and_octave (birdears.interval.DiatonicInterval attribute), 20, 40 note_name (birdears.interval.ChromaticInterval attribute), 19, 39 note_name (birdears.interval.DiatonicInterval attribute), 20, 40 P play() (birdears.sequence.Sequence method), 27, 48 play_element() (birdears.sequence.Sequence method), 27, 48	ears.resolution), 25, 45 repeat_only() (in module birdears.resolution), 25, 45 Resolution (class in birdears.resolution), 24, 45 return_simple() (birdears.interval.IntervalBase method),
none() (in module birdears.prequestion), 22, 43 note_and_octave (birdears.interval.ChromaticInterval attribute), 19, 39 note_and_octave (birdears.interval.DiatonicInterval attribute), 20, 40 note_name (birdears.interval.ChromaticInterval attribute), 19, 39 note_name (birdears.interval.DiatonicInterval attribute), 20, 40 P play() (birdears.sequence.Sequence method), 27, 48 play_element() (birdears.sequence.Sequence method),	ears.resolution), 25, 45 repeat_only() (in module birdears.resolution), 25, 45 Resolution (class in birdears.resolution), 24, 45 return_simple() (birdears.interval.IntervalBase method),
none() (in module birdears.prequestion), 22, 43 note_and_octave (birdears.interval.ChromaticInterval attribute), 19, 39 note_and_octave (birdears.interval.DiatonicInterval attribute), 20, 40 note_name (birdears.interval.ChromaticInterval attribute), 19, 39 note_name (birdears.interval.DiatonicInterval attribute), 20, 40 P play() (birdears.sequence.Sequence method), 27, 48 play_element() (birdears.sequence.Sequence method), 27, 48 play_question() (birdears.questionbase.QuestionBase method), 24, 44	ears.resolution), 25, 45 repeat_only() (in module birdears.resolution), 25, 45 Resolution (class in birdears.resolution), 24, 45 return_simple() (birdears.interval.IntervalBase method),
none() (in module birdears.prequestion), 22, 43 note_and_octave (birdears.interval.ChromaticInterval attribute), 19, 39 note_and_octave (birdears.interval.DiatonicInterval attribute), 20, 40 note_name (birdears.interval.ChromaticInterval attribute), 19, 39 note_name (birdears.interval.DiatonicInterval attribute), 20, 40 P play() (birdears.sequence.Sequence method), 27, 48 play_element() (birdears.sequence.Sequence method), 27, 48 play_question() (birdears.questionbase.QuestionBase method), 24, 44 play_question() (birdears.questions.harmonicinterval.Harmonicint	ears.resolution), 25, 45 repeat_only() (in module birdears.resolution), 25, 45 Resolution (class in birdears.resolution), 24, 45 return_simple() (birdears.interval.IntervalBase method),
none() (in module birdears.prequestion), 22, 43 note_and_octave (birdears.interval.ChromaticInterval attribute), 19, 39 note_and_octave (birdears.interval.DiatonicInterval attribute), 20, 40 note_name (birdears.interval.ChromaticInterval attribute), 19, 39 note_name (birdears.interval.DiatonicInterval attribute), 20, 40 P play() (birdears.sequence.Sequence method), 27, 48 play_element() (birdears.sequence.Sequence method), 27, 48 play_question() (birdears.questionbase.QuestionBase method), 24, 44 play_question() (birdears.questions.harmonicinterval.Harmonicint	ears.resolution), 25, 45 repeat_only() (in module birdears.resolution), 25, 45 Resolution (class in birdears.resolution), 24, 45 return_simple() (birdears.interval.IntervalBase method),
none() (in module birdears.prequestion), 22, 43 note_and_octave (birdears.interval.ChromaticInterval attribute), 19, 39 note_and_octave (birdears.interval.DiatonicInterval attribute), 20, 40 note_name (birdears.interval.ChromaticInterval attribute), 19, 39 note_name (birdears.interval.DiatonicInterval attribute), 20, 40 P play() (birdears.sequence.Sequence method), 27, 48 play_element() (birdears.sequence.Sequence method), 27, 48 play_question() (birdears.questionbase.QuestionBase method), 24, 44 play_question() (birdears.questions.harmonicinterval.Harm method), 13, 50 play_question() (birdears.questions.instrumentaldictation.In	ears.resolution), 25, 45 repeat_only() (in module birdears.resolution), 25, 45 Resolution (class in birdears.resolution), 24, 45 return_simple() (birdears.interval.IntervalBase method),
none() (in module birdears.prequestion), 22, 43 note_and_octave (birdears.interval.ChromaticInterval attribute), 19, 39 note_and_octave (birdears.interval.DiatonicInterval attribute), 20, 40 note_name (birdears.interval.ChromaticInterval attribute), 19, 39 note_name (birdears.interval.DiatonicInterval attribute), 20, 40 P play() (birdears.sequence.Sequence method), 27, 48 play_element() (birdears.sequence.Sequence method), 27, 48 play_question() (birdears.questionbase.QuestionBase method), 24, 44 play_question() (birdears.questions.harmonicinterval.Harmonthod), 13, 50 play_question() (birdears.questions.instrumentaldictation.Interval), 15, 52 play_question() (birdears.questions.melodicdictation.Melodicalication.Melodicalication() (birdears.questions.melodicdictation.Melodicalication() (birdears.questions.melodicdictation() (birdears.questions.melodicdictation() (birdears.questions.melodicdictation() (birdears.questions.melodicdictation() (birdears.questions.melodicdictation() (birdears.questions.melodicdictation() (birdears.questions.melodicdictation() (birdears.questions.melodicdictation() (birdears.questions.melodicdictation() (birdears.questions) (birdears.questi	ears.resolution), 25, 45 repeat_only() (in module birdears.resolution), 25, 45 Resolution (class in birdears.resolution), 24, 45 return_simple() (birdears.interval.IntervalBase method),
none() (in module birdears.prequestion), 22, 43 note_and_octave (birdears.interval.ChromaticInterval attribute), 19, 39 note_and_octave (birdears.interval.DiatonicInterval attribute), 20, 40 note_name (birdears.interval.ChromaticInterval attribute), 19, 39 note_name (birdears.interval.DiatonicInterval attribute), 20, 40 P play() (birdears.sequence.Sequence method), 27, 48 play_element() (birdears.sequence.Sequence method), 27, 48 play_question() (birdears.questionbase.QuestionBase method), 24, 44 play_question() (birdears.questions.harmonicinterval.Harm method), 13, 50 play_question() (birdears.questions.instrumentaldictation.In	ears.resolution), 25, 45 repeat_only() (in module birdears.resolution), 25, 45 Resolution (class in birdears.resolution), 24, 45 return_simple() (birdears.interval.IntervalBase method), 21, 41 S scale (birdears.scale.ChromaticScale attribute), 25, 46 scale (birdears.scale.DiatonicScale attribute), 26, 46 ScaleBase (class in birdears.scale), 26, 47 semitones (birdears.interval.ChromaticInterval attribute), 19, 39 semitones (birdears.interval.DiatonicInterval attribute), 20, 41 Sequence (class in birdears.sequence), 26, 47 T conicIntervalQuestion conic_octave (birdears.interval.ChromaticInterval attribute), 19, 39 strumentalDictationQuestion conic_octave (birdears.interval.DiatonicInterval attribute), 20, 40 dicDictationQuestion conic_only() (in module birdears.prequestion), 23, 43

Index 63