# birdears Documentation

Release 0.2.1

**lacchus Mercurius** 

# **CONTENTS**

1	Support						
2	Featu	ures	5				
3	Insta	lling birdears	7				
	3.1	Installing the dependencies	7				
		3.1.1 Arch Linux	7				
	3.2	Installing birdears	7				
		3.2.1 In-depth installation	7				
4	Using	g birdears	9				
	4.1	What is Functional Ear Training	9				
	4.2	The method	9				
	4.3	birdears modes and basic usage	9				
		4.3.1 melodic	10				
		4.3.2 harmonic	11				
		4.3.3 dictation	12				
		4.3.4 instrumental	13				
	4.4	Loading from config/preset files	14				
		4.4.1 Pre-made presets	14				
		4.4.2 Creating new preset files	14				
	4.5	Keybindings	14				
		4.5.1 On the keybindings	14				
		4.5.2 Major (Ionian)	14				
		4.5.3 Dorian	16				
		4.5.4 Phrygian	16				
		4.5.5 Lydian	16				
		4.5.6 Mixolydian	16				
		4.5.7 Minor (Aeolian)	16				
		4.5.8 Locrian	16				
5	hirde	ears package	19				
	5.1		20				
	3.1	5.1.1 birdears.interfaces package	20				
		5.1.2 birdears.questions package	21				
	5.2	Submodules	28				
	5.3	birdears.interval module	28				
	5.4	birdears.logger module	29				
	5.5		29				
	5.6	birdears.questionbase module	30				
	5.0	on dear or que ou or	20				

	<ul><li>5.7</li><li>5.8</li><li>5.9</li></ul>	birdears.scale module	31 32 33
6	Supp	ort	35
7	Featu	ires	37
8	Insta	8	39
	8.1		39
			39
	8.2		39
		8.2.1 In-depth installation	39
9	_	,	41
	9.1	2	41
	9.2		41
	9.3	E Company of the Comp	41
			42
			43
			44 45
	9.4		43 46
	9.4	6 61	46
		1	46
	9.5	of the proper may be a series of the series	46
	,		46
			46
			48
			48
		9.5.5 Lydian	48
		9.5.6 Mixolydian	48
		9.5.7 Minor (Aeolian)	48
		9.5.8 Locrian	48
10	API	•	51
11	birde	ars package	53
	11.1	Subpackages	54
	11.2		54
	11.3		54
	11.4		55
	11.5	1 1	55 5
	11.6	1	56
	11.7 11.8		58 50
	11.8		58
	11.9	birdears.sequence module	60
12			61
	12.1 12.2		61 61
	12.2		63
	12.3		65
	12.5		67
			- 1

13	birde	ears.interfaces package										6
	13.1	Submodules	 			 				 		6
	13.2	birdears.interfaces.commandline module	 			 				 		6
Рy	thon N	Module Index										7
Ind	lex											7.

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.

Project at GitHub.

Download the PDF version of this book. Clicking here.

CONTENTS 1

2 CONTENTS

# **ONE**

# **SUPPORT**

If you need help you can get in touch via IRC or file an issue on any matter regarding birdears at Github.

Media	Channel
IRC	#birdears at irc.freenode.org/6697 -ssl
GitHub	https://github.com/iacchus/birdears
GH issues	https://github.com/iacchus/birdears/issues
ReadTheDocs	https://birdears.readthedocs.io
PyPI	https://pypi.python.org/pypi/birdears
TravisCI	https://travis-ci.org/iacchus/birdears
Coveralls	https://coveralls.io/github/iacchus/birdears

# **TWO**

# **FEATURES**

- questions
- pretty much configurable
- load from config file
- you can make your own presets
- can be used interactively (docs needed)
- can be used as a library (docs needed)

**THREE** 

## **INSTALLING BIRDEARS**

# 3.1 Installing the dependencies

### 3.1.1 Arch Linux

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

# 3.2 Installing birdears

To install, simple do this command with pip3

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

# 3.2.1 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
```

**FOUR** 

# **USING BIRDEARS**

# 4.1 What is Functional Ear Training

write me!

# 4.2 The method

We can use abc language to notate music within 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:|
```

# 4.3 birdears modes and basic usage

birdears actually has four modes:

- melodic interval question
- harmonic interval question
- melodic dictation question
- instrumental dictation question

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.
 -h, --help
                      Show this message and exit.
Commands:
 dictation
             Melodic dictation
 harmonic Harmonic interval recognition
 instrumental Instrumental melodic time-based dictation
 load 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
```

#### 4.3.1 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:
 -m, --mode <mode>
                                Mode of the question.
 -t, --tonic <tonic>
                                Tonic of the question.
  -o, --octave <octave>
                               Octave of the question.
 -d, --descending
                                 Whether the question interval is descending.
 -c, --chromatic
                                 If chosen, question has chromatic notes.
                                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 play two notes, the tonic and the interval
 melodically, ie., one after the other and you should reply which is the
```

(continues on next page)

(continued from previous page)

```
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
```

### 4.3.2 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.
 -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
 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
```

(continues on next page)

(continued from previous page)

```
-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
```

### 4.3.3 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,
  locrian
  -t <tonic> is one of: A, A#, Ab, B, Bb, C, C#, D, D#, Db, E, Eb, F, F#, G,
  -p -p prequestion_method> is one of: none, tonic_only, progression_i_iv_v_i
  -r <resolution_method> is one of: nearest_tonic, repeat_only
```

### 4.3.4 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.
                                Times to repeat question.
 -u, --n_repeats <times>
 -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
```

# 4.4 Loading from config/preset files

## 4.4.1 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

## 4.4.2 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.

# 4.5 Keybindings

# 4.5.1 On the keybindings

The following keyboard diagrams should give you an idea on how the keybindings work. Please note how the keys on the line from z (unison) to , (comma, octave) represent the notes that are natural to the mode, and the line above represent the chromatics.

Also, for exercises with two octaves, the **uppercased keys represent the second octave**. For example, z is *unison*, z is the *octave*, z (uppercased) is the *double octave*. The same for all the other intervals.

## 4.5.2 Major (Ionian)

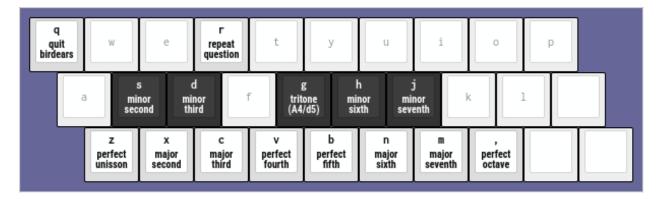


Fig. 1: Keyboard diagram for the --mode major (default).

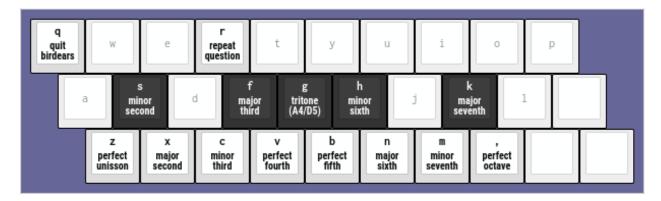


Fig. 2: Keyboard diagram for the --mode dorian.

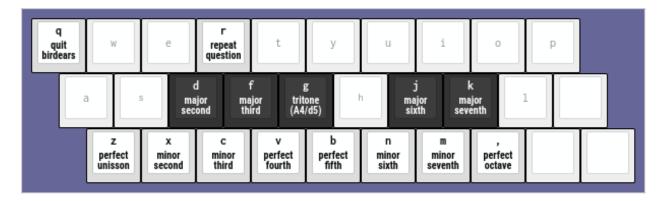


Fig. 3: Keyboard diagram for the --mode phrygian.

4.5. Keybindings

### 4.5.3 Dorian

## 4.5.4 Phrygian

# 4.5.5 Lydian

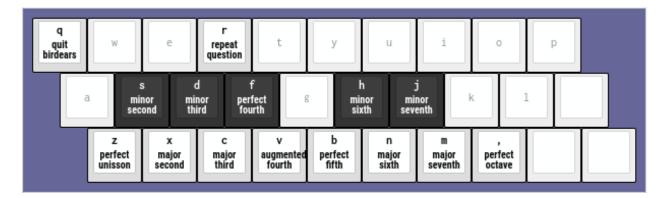


Fig. 4: Keyboard diagram for the --mode lydian.

### 4.5.6 Mixolydian

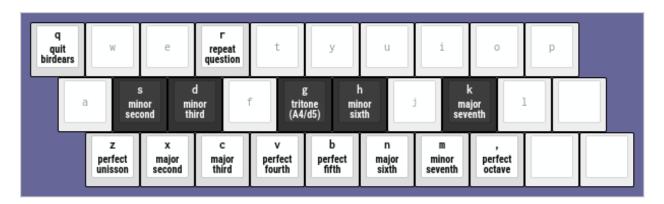


Fig. 5: Keyboard diagram for the --mode mixolydian.

# 4.5.7 Minor (Aeolian)

# 4.5.8 Locrian

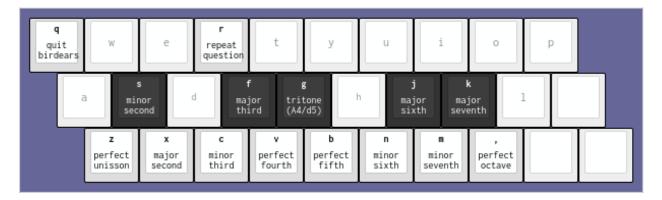


Fig. 6: Keyboard diagram for the --mode minor.

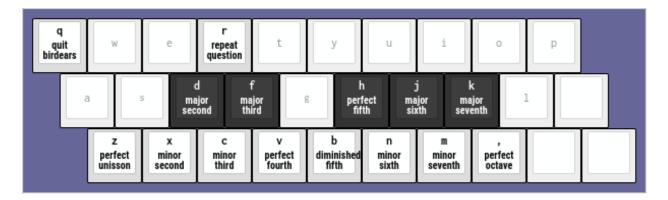


Fig. 7: Keyboard diagram for the --mode locrian.

4.5. Keybindings

**FIVE** 

### **BIRDEARS PACKAGE**

birdears provides facilities to building musical ear training exercises.

birdears.CHROMATIC\_FLAT = ('C', 'Db', 'D', 'Eb', 'E', 'F', 'Gb', 'G', 'Ab', 'A', 'Bb', 'B')

tuple - Chromatic notes names using flats.

A mapping of the chromatic note names using flats.

birdears.CHROMATIC\_SHARP = ('C', 'C#', 'D', 'D#', 'E', 'F', 'F#', 'G', 'G#', 'A', 'A#', 'B tuple - Chromatic notes names using sharps.

A mapping of the chromatic note namesu sing sharps

birdears.CHROMATIC\_TYPE = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11) tuple - A map of the chromatic scale.

A map of the semitones which compound the chromatic scale.

birdears.CIRCLE\_OF\_FIFTHS = [('C', 'G', 'D', 'A', 'E', 'B', 'Gb', 'Db', 'Ab', 'Eb', 'Bb', list of tuples - Circle of fifths.

These are the circle of fifth in both directions.

birdears.D(data, nlines=0)

birdears.DEGREE\_INDEX = {'i': [0], 'ii': [1, 2], 'iii': [3, 4], 'iv': [5, 6], 'v': [6 dict of lists - A mapping of semitones of each degree.

A mapping of semitones which index to each degree roman numeral, major/minor, perfect, augmented/diminished

birdears.DIATONIC\_MASK = {'dorian': (1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0), 'locrian': (1, dict of tuples - A map of the diatonic scale.

A mapping of the semitones which compound each of the greek modes.

birdears.INTERVALS = ((0, 'P1', 'Perfect Unison'), (1, 'm2', 'Minor Second'), (2, 'M2', 'Matuple of tuples - Data representing intervals.

A tuple of tuples representing data for the intervals with format (semitones, short name, full name).

birdears.INTERVAL\_INDEX = {1: [0], 2: [1, 2], 3: [3, 4], 4: [5, 6], 5: [6, 7], 6: [8 dict of lists – A mapping of semitones of each interval.

A mapping of semitones which index to each interval name, major/minor, perfect, augmented/diminished

birdears.KEYS = ('C', 'C#', 'Db', 'D', 'D#', 'Eb', 'E', 'F', 'F#', 'Gb', 'G', 'G#', 'Ab', tuple - Allowed keys

These are the allowed keys for exercise as comprehended by birdears.

# 5.1 Subpackages

# 5.1.1 birdears.interfaces package

#### **Submodules**

#### birdears.interfaces.commandline module

```
class birdears.interfaces.commandline.CommandLine(exercise=None, *args, **kwargs)
    Bases: object
    __init__(exercise=None, *args, **kwargs)
```

### Parameters

• **exercise** (str) – The question name.

This function implements the birdears loop for command line.

• \*\*kwargs (kwargs) - FIXME: The kwargs can contain options for specific questions.

```
process_key (user_input)
```

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()

user\_durations=None,

tion method='none',

tion\_method='nearest\_tonic',

preques-

resolu-

\*args, \*\*kwargs)

### 5.1.2 birdears.guestions package

#### **Submodules**

### birdears.questions.harmonicinterval module

```
tonic='C',
octave=4,
descend-
ing=False,
chro-
matic=False,
n_octaves=1,
valid_intervals=(0,
1, 2, 3, 4,
5, 6, 7, 8,
9, 10, 11),
```

class birdears.questions.harmonicinterval.HarmonicIntervalQuestion(mode='major',

Bases: birdears.questionbase.QuestionBase

Implements a Harmonic Interval test.

```
__init__ (mode='major', tonic='C', octave=4, descending=False, chromatic=False, n_octaves=1, valid_intervals=(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11), 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

5.1. Subpackages 21

seconds; delay is the time to wait before playing the next note, and pos\_delay is the time to wait after all the notes of the respective sequence have been played. If any of the user durations is 'n, the default duration for the type of question will be used instead. Example:

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

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

```
check_question (user_input_char)
```

Checks whether the given answer is correct.

```
{\tt make\_pre\_question}\ (\textit{method}\ )
```

make\_question()

This method should be overwritten by the question subclasses.

```
make_resolution (method)
```

This method should be overwritten by the question subclasses.

```
name = 'harmonic'
```

play\_question (callback=None, end\_callback=None, \*args, \*\*kwargs)

This method should be overwritten by the question subclasses.

play\_resolution (callback=None, end\_callback=None, \*args, \*\*kwargs)

### birdears.questions.instrumentaldictation module

 $\textbf{class} \ \, \texttt{birdears.questions.instrumental dictation.Instrumental Dictation Question} \ \, (\textit{mode='major'}, \\ \, \text{the description of the d$ 

 $wait\_time=11$ ,  $n\_repeats=1$ ,  $max_intervals=3$ ,  $n\_notes=4$ , tonic='C', octave=4, descending=False, chromatic=False,  $n \ octaves=1$ , valid intervals=(0, 1, 2, 3. 4, 5, 6, 7, 8, 9. 10, 11), 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='C', octave=4, descending=False, chromatic=False, n_octaves=1, valid_intervals=(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11), 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.

5.1. Subpackages 23

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

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

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

### check\_question()

Checks whether the given answer is correct.

This currently doesn't applies to instrumental dictation questions.

```
make_pre_question (method)
```

#### make\_question()

This method should be overwritten by the question subclasses.

### make\_resolution (method)

This method should be overwritten by the question subclasses.

```
name = 'instrumental'
```

play\_question (callback=None, end\_callback=None, \*args, \*\*kwargs)

This method should be overwritten by the question subclasses.

#### birdears.guestions.melodicdictation module

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

 $max_intervals=3$ ,  $n\_notes=4$ , tonic='C', octave=4, descending=False, chromatic=False,  $n \ octaves=1$ ,  $valid\ intervals=(0,$ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11), user durations=None, prequestion\_method='progression\_i\_iv\_v\_ resolution\_method='repeat\_only', \*args, \*\*kwargs)

Bases: birdears.questionbase.QuestionBase

Implements a melodic dictation test.

```
__init__ (mode='major', max_intervals=3, n_notes=4, tonic='C', octave=4, descending=False, chromatic=False, n_octaves=1, valid_intervals=(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11), 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'.
- 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 dura-

5.1. Subpackages 25

tion (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.

This method should be overwritten by the question subclasses.

#### make\_resolution (method)

This method should be overwritten by the question subclasses.

```
name = 'dictation'
```

play\_question (callback=None, end\_callback=None, \*args, \*\*kwargs)
This method should be overwritten by the question subclasses.

play\_resolution (callback=None, end\_callback=None, \*args, \*\*kwargs)

#### birdears.questions.melodicinterval module

```
class birdears.questions.melodicinterval.MelodicIntervalQuestion(mode='major',
```

```
tonic='C',
octave=4,
descend-
ing=False,
chro-
matic=False,
n\_octaves=1,
valid intervals=(0,
1, 2, 3, 4,
5, 6, 7, 8,
9, 10, 11),
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='C', octave=4, descending=False, chromatic=False, n\_octaves=1, valid\_intervals=(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11), user\_durations=None, prequestion\_method='tonic\_only', resolution\_method='nearest\_tonic', \*args, \*\*kwargs)

Inits the class.

#### **Parameters**

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

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

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

```
check_question (user_input_char)
```

Checks whether the given answer is correct.

```
make_pre_question (method)
make_question ()
```

This method should be overwritten by the question subclasses.

```
make_resolution (method)
```

This method should be overwritten by the question subclasses.

```
name = 'melodic'
play_question(callback=None, end_callback=None, *args, **kwargs)
    This method should be overwritten by the question subclasses.
play resolution(callback=None, end_callback=None, *args, **kwargs)
```

5.1. Subpackages 27

## 5.2 Submodules

## 5.3 birdears.interval module

```
class birdears.interval.Interval(pitch_a, pitch_b)
Bases: dict.
```

This class represents the interval between two pitches..

#### 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___(pitch_a, pitch_b)
```

Measures the musical interval from pitch\_a to pitch\_b.

#### **Parameters**

- pitch\_a (str) First Pitch object to be measured.
- pitch\_b (str) Second Pitch object to be measured.

birdears.interval.get\_interval\_by\_semitones(semitones)

# 5.4 birdears.logger module

This submodule exports *logger* to log events.

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

```
Numeric value
CRITICAL
           50
           40
ERROR
WARNING
           30
INFO
           2.0
DEBUG
           10
NOTSET
Level
           When it's used
DEBUG
            Detailed information, typically of interest only when
                diagnosing problems.
INFO
           Confirmation that things are working as expected.
WARNING
           An indication that something unexpected happened, or indicative
                of some problem in the near future (e.g. 'disk space low').
                The software is still working as expected.
ERROR
           Due to a more serious problem, the software has not been able
                to perform some function.
CRITICAL
            A serious error, indicating that the program itself may be
                unable to continue running.
```

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

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

# 5.5 birdears.prequestion module

This module implements pre-questions' progressions.

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

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

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

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

#### **Parameters**

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

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

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

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

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

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

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

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

Decorator for prequestion method functions.

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

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

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

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

# 5.6 birdears.questionbase module

```
class birdears.questionbase.QuestionBase (mode='major', tonic='C', octave=4, descend-
ing=False, chromatic=False, n_octaves=1,
valid_intervals=(0, 1, 2, 3, 4, 5, 6, 7, 8,
9, 10, 11), user_durations=None, preques-
tion_method=None, resolution_method=None,
default durations=None, *args, **kwargs)
```

Bases: object

Base Class to be subclassed for Question classes.

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

\_\_init\_\_ (mode='major', tonic='C', octave=4, descending=False, chromatic=False, n\_octaves=1, valid\_intervals=(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11), user\_durations=None, prequestion\_method=None, resolution\_method=None, default\_durations=None, \*args, \*\*kwargs)
Inits the class.

#### **Parameters**

- mode (str) A string represnting the mode of the question. Eg., 'major' or 'minor'
- **tonic** (*str*) A string representing the tonic of the question, eg.: 'C'; if omitted, it will be selected randomly.

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

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

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

### check\_question()

This method should be overwritten by the question subclasses.

### 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.get_valid_pitches(scale, valid_intervals=(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11))
```

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

Decorator for question classes.

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

## 5.7 birdears.resolution module

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

This class implements methods for different types of question resolutions.

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

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

Calls the resolution method and pass arguments to it.

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

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

Inits the resolution class.

#### **Parameters**

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

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

Resolution method that resolve the intervals to their nearest tonics.

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

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

Decorator for resolution method functions.

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

```
birdears.resolution.repeat_only (question)
```

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

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

## 5.8 birdears.scale module

Bases: birdears.scale.ScaleBase

Builds a musical chromatic scale.

#### scale

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

\_\_init\_\_ (tonic='C', octave=4, n\_octaves=1, descending=False, dont\_repeat\_tonic=False)

Returns a chromatic scale from tonic.

#### **Parameters**

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

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

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

#### **Parameters**

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

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

Bases: birdears.scale.ScaleBase

Builds a musical diatonic scale.

#### scale

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

```
__init__(tonic='C', mode='major', octave=4, n_octaves=1, descending=False, dont_repeat_tonic=False)
```

Returns a diatonic scale from tonic and mode.

#### **Parameters**

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

```
get triad(index=0, degree=None)
```

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

#### **Parameters**

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

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

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

# 5.9 birdears.sequence module

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

Register a Sequence of notes and/or chords.

#### elements

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

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

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

### make\_chord\_progression (tonic\_pitch, 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, \*args, \*\*kwargs)

# CHAPTER

# SIX

# **SUPPORT**

If you need help you can get in touch via IRC or file an issue on any matter regarding birdears at Github.

Media	Channel
IRC	#birdears at irc.freenode.org/6697 -ssl
GitHub	https://github.com/iacchus/birdears
GH issues	https://github.com/iacchus/birdears/issues
ReadTheDocs	https://birdears.readthedocs.io
PyPI	https://pypi.python.org/pypi/birdears
TravisCI	https://travis-ci.org/iacchus/birdears
Coveralls	https://coveralls.io/github/iacchus/birdears

## **CHAPTER**

# **SEVEN**

# **FEATURES**

- questions
- pretty much configurable
- load from config file
- you can make your own presets
- can be used interactively (docs needed)
- can be used as a library (docs needed)

38 Chapter 7. Features

**CHAPTER** 

**EIGHT** 

## **INSTALLING BIRDEARS**

# 8.1 Installing the dependencies

## 8.1.1 Arch Linux

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

# 8.2 Installing birdears

To install, simple do this command with pip3

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

# 8.2.1 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** 

NINE

# **USING BIRDEARS**

# 9.1 What is Functional Ear Training

write me!

## 9.2 The method

We can use abc language to notate music within 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:|
```

# 9.3 birdears modes and basic usage

birdears actually has four modes:

- · melodic interval question
- harmonic interval question
- melodic dictation question
- instrumental dictation question

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.
 -h, --help
                      Show this message and exit.
Commands:
 dictation
             Melodic dictation
 harmonic Harmonic interval recognition
 instrumental Instrumental melodic time-based dictation
 load 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
```

#### 9.3.1 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:
 -m, --mode <mode>
                                Mode of the question.
 -t, --tonic <tonic>
                                Tonic of the question.
  -o, --octave <octave>
                               Octave of the question.
 -d, --descending
                                 Whether the question interval is descending.
 -c, --chromatic
                                 If chosen, question has chromatic notes.
                                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 play two notes, the tonic and the interval
 melodically, ie., one after the other and you should reply which is the
```

(continues on next page)

(continued from previous page)

```
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
```

### 9.3.2 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.
 -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
 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
```

(continues on next page)

(continued from previous page)

```
-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
```

### 9.3.3 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,
  locrian
  -t <tonic> is one of: A, A#, Ab, B, Bb, C, C#, D, D#, Db, E, Eb, F, F#, G,
  -p -p prequestion_method> is one of: none, tonic_only, progression_i_iv_v_i
  -r <resolution_method> is one of: nearest_tonic, repeat_only
```

### 9.3.4 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
```

# 9.4 Loading from config/preset files

# 9.4.1 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

## 9.4.2 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.

# 9.5 Keybindings

## 9.5.1 On the keybindings

The following keyboard diagrams should give you an idea on how the keybindings work. Please note how the keys on the line from z (unison) to , (comma, octave) represent the notes that are natural to the mode, and the line above represent the chromatics.

Also, for exercises with two octaves, the **uppercased keys represent the second octave**. For example, z is *unison*, z is the *octave*, z (uppercased) is the *double octave*. The same for all the other intervals.

## 9.5.2 Major (Ionian)

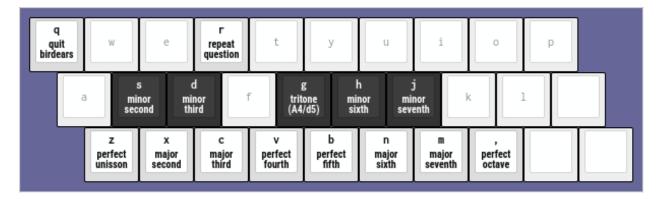


Fig. 1: Keyboard diagram for the --mode major (default).

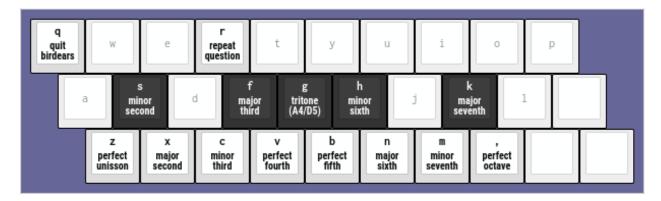


Fig. 2: Keyboard diagram for the --mode dorian.

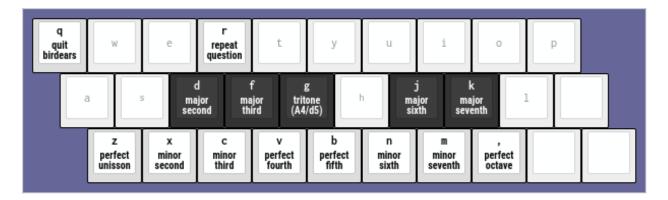


Fig. 3: Keyboard diagram for the --mode phrygian.

9.5. Keybindings 47

## 9.5.3 Dorian

## 9.5.4 Phrygian

## 9.5.5 Lydian

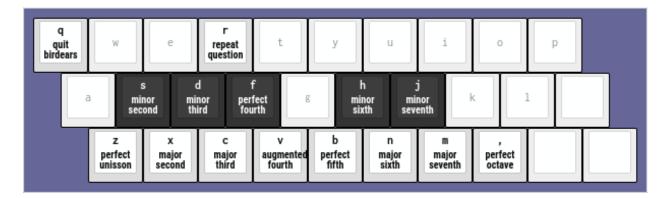


Fig. 4: Keyboard diagram for the --mode lydian.

## 9.5.6 Mixolydian

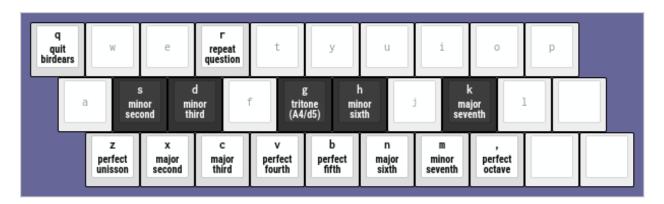


Fig. 5: Keyboard diagram for the --mode mixolydian.

## 9.5.7 Minor (Aeolian)

## 9.5.8 Locrian

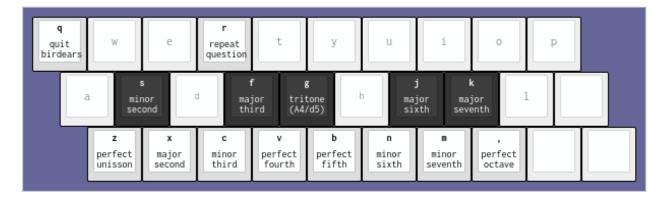


Fig. 6: Keyboard diagram for the --mode minor.

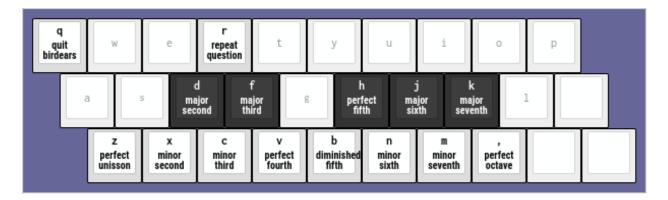


Fig. 7: Keyboard diagram for the --mode locrian.

9.5. Keybindings 49

CHA	PTER
	TEN

API

52 Chapter 10. API

**CHAPTER** 

### **ELEVEN**

## **BIRDEARS PACKAGE**

birdears provides facilities to building musical ear training exercises.

birdears.CHROMATIC\_FLAT = ('C', 'Db', 'D', 'Eb', 'E', 'F', 'Gb', 'G', 'Ab', 'A', 'Bb', 'B')

tuple - Chromatic notes names using flats.

A mapping of the chromatic note names using flats.

birdears.CHROMATIC\_SHARP = ('C', 'C#', 'D', 'D#', 'E', 'F', 'F#', 'G', 'G#', 'A', 'A#', 'B tuple - Chromatic notes names using sharps.

A mapping of the chromatic note namesu sing sharps

birdears.CHROMATIC\_TYPE = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11) tuple - A map of the chromatic scale.

A map of the semitones which compound the chromatic scale.

birdears.CIRCLE\_OF\_FIFTHS = [('C', 'G', 'D', 'A', 'E', 'B', 'Gb', 'Db', 'Ab', 'Eb', 'Bb', list of tuples - Circle of fifths.

These are the circle of fifth in both directions.

birdears.D(data, nlines=0)

birdears.DEGREE\_INDEX = {'i': [0], 'ii': [1, 2], 'iii': [3, 4], 'iv': [5, 6], 'v': [6 dict of lists - A mapping of semitones of each degree.

A mapping of semitones which index to each degree roman numeral, major/minor, perfect, augmented/diminished

birdears.DIATONIC\_MASK = {'dorian': (1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0), 'locrian': (1, dict of tuples - A map of the diatonic scale.

A mapping of the semitones which compound each of the greek modes.

birdears.INTERVALS = ((0, 'P1', 'Perfect Unison'), (1, 'm2', 'Minor Second'), (2, 'M2', 'Matuple of tuples - Data representing intervals.

A tuple of tuples representing data for the intervals with format (semitones, short name, full name).

birdears.INTERVAL\_INDEX = {1: [0], 2: [1, 2], 3: [3, 4], 4: [5, 6], 5: [6, 7], 6: [8 dict of lists – A mapping of semitones of each interval.

A mapping of semitones which index to each interval name, major/minor, perfect, augmented/diminished

birdears.KEYS = ('C', 'C#', 'Db', 'D', 'D#', 'Eb', 'E', 'F', 'F#', 'Gb', 'G', 'G#', 'Ab', tuple - Allowed keys

These are the allowed keys for exercise as comprehended by birdears.

# 11.1 Subpackages

## 11.2 Submodules

## 11.3 birdears.interval module

```
class birdears.interval.Interval(pitch_a, pitch_b)
    Bases: dict
```

This class represents the interval between two pitches..

#### 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___(pitch_a, pitch_b)
```

Measures the musical interval from pitch\_a to pitch\_b.

#### **Parameters**

- pitch\_a (str) First Pitch object to be measured.
- pitch\_b (str) Second Pitch object to be measured.

birdears.interval.get\_interval\_by\_semitones (semitones)

# 11.4 birdears.logger module

This submodule exports *logger* to log events.

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

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

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

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

# 11.5 birdears.prequestion module

This module implements pre-questions' progressions.

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

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

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

```
class birdears.prequestion.PreQuestion (method, question)
    Bases: object
    __call__(*args, **kwargs)
    Calls the resolution method and pass arguments to it.
```

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

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

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

#### **Parameters**

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

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

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

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

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

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

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

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

Decorator for prequestion method functions.

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

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

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

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

# 11.6 birdears.questionbase module

```
class birdears.questionbase.QuestionBase (mode="major", tonic="C", octave=4, descending=False, chromatic=False, n\_octaves=1, valid\_intervals=(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11), user\_durations=None, prequestion\_method=None, resolution\_method=None, default\_durations=None, *args, **kwargs)
```

Bases: object

Base Class to be subclassed for Question classes.

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

\_\_init\_\_ (mode='major', tonic='C', octave=4, descending=False, chromatic=False, n\_octaves=1, valid\_intervals=(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11), user\_durations=None, prequestion\_method=None, resolution\_method=None, default\_durations=None, \*args, \*\*kwargs)
Inits the class.

### **Parameters**

• mode (str) – A string represnting the mode of the question. Eg., 'major' or 'minor'

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

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

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

#### check\_question()

This method should be overwritten by the question subclasses.

### 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.get_valid_pitches (scale, valid_intervals=(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11))
```

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

Decorator for question classes.

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

## 11.7 birdears.resolution module

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

This class implements methods for different types of question resolutions.

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

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

Calls the resolution method and pass arguments to it.

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

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

Inits the resolution class.

#### **Parameters**

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

```
birdears.resolution.nearest tonic (question)
```

Resolution method that resolve the intervals to their nearest tonics.

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

```
birdears.resolution.register resolution method(f, *args, **kwargs)
```

Decorator for resolution method functions.

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

```
birdears.resolution.repeat_only (question)
```

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

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

## 11.8 birdears.scale module

 $Bases: \ \textit{birdears.scale.ScaleBase}$ 

Builds a musical chromatic scale.

#### scale

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

\_\_init\_\_ (tonic='C', octave=4, n\_octaves=1, descending=False, dont\_repeat\_tonic=False)
Returns a chromatic scale from tonic.

#### **Parameters**

- tonic (str) The note which the scale will be built upon.
- **octave** (*int*) The scientific octave the scale will be built upon.

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

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

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

#### **Parameters**

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

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

Bases: birdears.scale.ScaleBase

Builds a musical diatonic scale.

#### scale

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

\_\_init\_\_(tonic='C', mode='major', octave=4, n\_octaves=1, descending=False, dont\_repeat\_tonic=False)

Returns a diatonic scale from tonic and mode.

#### **Parameters**

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

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

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

#### **Parameters**

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

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

class birdears.scale.ScaleBase

Bases: list

# 11.9 birdears.sequence module

class birdears.sequence (elements=[], duration=2, delay=1.5, pos\_delay=1)
 Bases: list

Register a Sequence of notes and/or chords.

#### elements

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

async\_play (callback, end\_callback, args, kwargs)

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

make\_chord\_progression (tonic\_pitch, 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, \*args, \*\*kwargs)

**CHAPTER** 

## **TWELVE**

## BIRDEARS.QUESTIONS PACKAGE

## 12.1 Submodules

# 12.2 birdears.questions.harmonicinterval module

```
class birdears.questions.harmonicinterval.HarmonicIntervalQuestion(mode='major',
                                                                                        tonic='C',
                                                                                        octave=4,
                                                                                        descend-
                                                                                        ing=False,
                                                                                        chro-
                                                                                        matic=False,
                                                                                        n_octaves=1,
                                                                                        valid\_intervals = (0,
                                                                                        1, 2, 3, 4,
                                                                                        5, 6, 7, 8,
                                                                                        9, 10, 11),
                                                                                        user_durations=None,
                                                                                        preques-
                                                                                        tion_method='none',
                                                                                        resolu-
                                                                                        tion_method='nearest_tonic',
                                                                                        *args,
                                                                                        **kwargs)
```

Bases: birdears.questionbase.QuestionBase

Implements a Harmonic Interval test.

```
__init__ (mode='major', tonic='C', octave=4, descending=False, chromatic=False, n_octaves=1, valid_intervals=(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11), user_durations=None, prequestion_method='none', resolution_method='nearest_tonic', *args, **kwargs')
Inits the class.
```

#### **Parameters**

- mode(str) A string representing the mode of the question. Eg., 'major' or 'minor'
- **tonic** (*str*) A string representing the tonic of the question, eg.: 'C'; if omitted, it will be selected randomly.
- **octave** (*int*) A scienfic octave notation, for example, 4 for 'C4'; if not present, it will be randomly chosen.

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

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

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

```
check_question(user_input_char)
```

Checks whether the given answer is correct.

```
{\tt make\_pre\_question}\ (method)
```

make\_question()

This method should be overwritten by the question subclasses.

make resolution (method)

This method should be overwritten by the question subclasses.

name = 'harmonic'

play\_question(callback=None, end\_callback=None, \*args, \*\*kwargs)

This method should be overwritten by the question subclasses.

play\_resolution (callback=None, end\_callback=None, \*args, \*\*kwargs)

# 12.3 birdears.questions.instrumentaldictation module

 $\textbf{class} \ \, \texttt{birdears.questions.instrumental} \\ \textbf{dictation.InstrumentalDictationQuestion} \, (\textit{mode='major'}, \\ \textbf{or instrumentalDictationQuestion}) \, (\textit{mode='major'}, \\ \textbf{or instrumentalDictation}) \, (\textit{$ 

```
wait\_time=11,
n repeats=1,
max_intervals=3,
n \ notes=4,
tonic='C',
oc-
tave=4,
de-
scend-
ing=False,
chro-
matic=False,
n_octaves=1,
valid\_intervals=(0,
1,
2,
3,
4,
5,
6,
7,
8,
9.
10,
11).
user_durations=No.
pre-
ques-
tion_method='prog
res-
0-
lu-
tion_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='C', octave=4, descending=False, chromatic=False, n_octaves=1, valid_intervals=(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11), user_durations=None, prequestion_method='progression_i_iv_v_i', resolution_method='repeat_only', *args, **kwargs')
Inits the class.
```

### **Parameters**

- mode (str) A string representing the mode of the question. Eg., 'major' or 'minor'.
- wait\_time (float) Wait time in seconds for the next question or repeat.
- n\_repeats (int) Number of times the same dictation will be repeated before the end of the exercise.

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

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

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

### check\_question()

Checks whether the given answer is correct.

This currently doesn't applies to instrumental dictation questions.

```
make_pre_question (method)
```

#### make\_question()

This method should be overwritten by the question subclasses.

### make\_resolution (method)

This method should be overwritten by the question subclasses.

```
name = 'instrumental'
```

```
play_question (callback=None, end_callback=None, *args, **kwargs)
```

This method should be overwritten by the question subclasses.

# 12.4 birdears.questions.melodicdictation module

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

```
max\_intervals=3,
n \ notes=4,
tonic='C',
octave=4,
descend-
ing=False,
chro-
matic = False,
n \ octaves=1,
valid intervals=(0,
1, 2, 3, 4,
5, 6, 7, 8,
9, 10, 11),
user_durations=None,
preques-
tion_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='C', octave=4, descending=False, chromatic=False, n_octaves=1, valid_intervals=(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11), 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'.
- max\_intervals (int) The maximum number of random intervals the question will have.
- n\_notes (int) The number of notes the melodic dictation will have.
- tonic (str) A string representing the tonic of the question, eg.: 'C'; if omitted, it will be selected randomly.
- **octave** (*int*) A scienfic octave notation, for example, 4 for 'C4'; if not present, it will be randomly chosen.
- **descending** (bool) Is the question direction in descending, ie., intervals have lower pitch than the tonic.
- **chromatic** (bool) If the question can have (True) or not (False) chromatic intervals, ie., intervals not in the diatonic scale of tonic/mode.
- n\_octaves (int) Maximum number of octaves of the question.
- **valid\_intervals** (*list*) A list with intervals (int) valid for random choice, 1 is 1st, 2 is second etc. Eg. [1, 4, 5] to allow only tonics, fourths and fifths.

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

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

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

```
check_question(user_input_keys)
```

Checks whether the given answer is correct.

```
make_pre_question (method)
```

make question()

This method should be overwritten by the question subclasses.

```
make_resolution (method)
```

This method should be overwritten by the question subclasses.

```
name = 'dictation'
```

play\_question (callback=None, end\_callback=None, \*args, \*\*kwargs)

This method should be overwritten by the question subclasses.

 $\verb"play_resolution" (callback=None, end_callback=None, *args, **kwargs)$ 

# 12.5 birdears.questions.melodicinterval module

octave=4, descending=False, chromatic=False,  $n \ octaves=1$ , valid intervals=(0,1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11), user\_durations=None, prequestion\_method='tonic\_only', resolution\_method='nearest\_tonic', \*args, \*\*kwargs)

Bases: birdears.questionbase.QuestionBase

Implements a Melodic Interval test.

\_\_init\_\_ (mode='major', tonic='C', octave=4, descending=False, chromatic=False, n\_octaves=1, valid\_intervals=(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11), user\_durations=None, prequestion\_method='tonic\_only', resolution\_method='nearest\_tonic', \*args, \*\*kwargs)
Inits the class.

#### **Parameters**

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

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

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

```
check_question (user_input_char)
    Checks whether the given answer is correct.

make_pre_question (method)

make_question()
    This method should be overwritten by the question subclasses.

make_resolution (method)
    This method should be overwritten by the question subclasses.

name = 'melodic'

play_question (callback=None, end_callback=None, *args, **kwargs)
    This method should be overwritten by the question subclasses.
```

play\_resolution (callback=None, end\_callback=None, \*args, \*\*kwargs)

**CHAPTER** 

## **THIRTEEN**

## **BIRDEARS.INTERFACES PACKAGE**

## 13.1 Submodules

## 13.2 birdears.interfaces.commandline module

```
__init__(exercise=None, *args, **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.

```
process_key (user_input)
```

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

This function returns input text centered according to terminal columns.

### **Parameters**

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

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

Makes a string representing intervals entered by the user.

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

#### **Parameters**

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

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

Prints the formatted response for 'instrumental' exercise.

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

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

Prints the question to the user.

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

birdears.interfaces.commandline.print\_response(response)
Prints the formatted response.

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

# **PYTHON MODULE INDEX**

## b

```
birdears,53
birdears.interfaces,69
birdears.interfaces.commandline,69
birdears.interval,54
birdears.logger,55
birdears.prequestion,55
birdears.questionbase,56
birdears.questions,61
birdears.questions.harmonicinterval,61
birdears.questions.instrumentaldictation,63
birdears.questions.melodicdictation,65
birdears.questions.melodicinterval,67
birdears.resolution,58
birdears.scale,58
birdears.sequence,60
```

72 Python Module Index

# **INDEX**

Symbols	birdears.questions.melodicinterval (module), 26, 67
call() (birdears.prequestion.PreQuestion method),	birdears.resolution (module), 31, 58
29, 55	birdears.scale (module), 32, 58
call() (birdears.resolution.Resolution method), 32,	birdears.sequence (module), 33, 60
50	•
init() (birdears.interfaces.commandline.CommandLine	$_{\mathfrak{g}}G$
method), 20, 69	center_text() (in module bird-
init() (birdears.interval.Interval method), 28, 54	ears.interfaces.commandline), 20, 69
init() (birdears.prequestion.PreQuestion method),	check_question() (birdears.questionbase.QuestionBase
29, 56	method), 31, 57
init() (birdears.questionbase.QuestionBase method),	check_question() (bird-
30, 56	ears.questions.harmonicinterval.HarmonicIntervalQuestion
init() (birdears.questions.harmonicinterval.HarmonicIn	ntervalQuestiethod), 22, 62
method), 21, 61	check_question() (bird-
init() (birdears.questions.instrumentaldictation.Instrum	nentalDictaciansQuestions.instrumentaldictation.InstrumentalDictationQuestion
method), 23, 63	method), 24, 64
init() (birdears.questions.melodicdictation.MelodicDic	etablealQquestition() (bird-
method), 25, 65	ears.questions.melodicdictation.MelodicDictationQuestion
init() (birdears.questions.melodicinterval.MelodicInter	valQuestiomethod), 26, 66
method), 26, 67	check_question() (bird-
init() (birdears.resolution.Resolution method), 32,	ears. questions. melodic interval. Melodic Interval Question
58	method), 27, 68
init() (birdears.scale.ChromaticScale method), 32,	CHROMATIC_FLAT (in module birdears), 19, 53
58	chromatic_offset (birdears.interval.Interval attribute), 28,
init() (birdears.scale.DiatonicScale method), 33, 59	54
Λ	CHROMATIC_SHARP (in module birdears), 19, 53
A	CHROMATIC_TYPE (in module birdears), 19, 53
async_play() (birdears.sequence.Sequence method), 34,	ChromaticScale (class in birdears.scale), 32, 58
60	CIRCLE_OF_FIFTHS (in module birdears), 19, 53
В	CommandLine (class in bird-
В	ears.interfaces.commandline), 20, 69
birdears (module), 19, 53	D
birdears.interfaces (module), 20, 69	_
birdears.interfaces.commandline (module), 20, 69	D() (in module birdears), 19, 53
birdears.interval (module), 28, 54	data (birdears.interval.Interval attribute), 28, 54
birdears.logger (module), 29, 55	DEGREE_INDEX (in module birdears), 19, 53
birdears.prequestion (module), 29, 55	diatonic_index (birdears.interval.Interval attribute), 28,
birdears.questionbase (module), 30, 56	54
birdears.questions (module), 21, 61	DIATONIC_MASK (in module birdears), 19, 53
birdears.questions.harmonicinterval (module), 21, 61	DiatonicScale (class in birdears.scale), 33, 59
birdears.questions.instrumentaldictation (module), 23, 63	distance (birdears.interval.Interval attribute), 28, 54
birdears.questions.melodicdictation (module), 25, 65	

E	make_question() (birdears.questions.instrumentaldictation.InstrumentalDict
elements (birdears.sequence.Sequence attribute), 33, 60	method), 24, 64 make_question() (birdears.questions.melodicdictation.MelodicDictationQue
G	method), 26, 66
get_interval_by_semitones() (in module bird-	make_question() (birdears.questions.melodicinterval.MelodicIntervalQuestion), 27, 68
ears.interval), 28, 55 get_triad() (birdears.scale.ChromaticScale method), 32, 59	make_resolution() (birdears.questionbase.QuestionBase method), 31, 57
get_triad() (birdears.scale.DiatonicScale method), 33, 59	make_resolution() (bird-
get_unad() (birdears.scale.biatomeseare method), 33, 39 get_valid_pitches() (in module birdears.questionbase), 31, 57	ears.questions.harmonicinterval.HarmonicIntervalQuestion method), 22, 62
	make_resolution() (bird-
H	ears.questions.instrumentaldictation.InstrumentalDictationQuestic
HarmonicIntervalQuestion (class in bird-	method), 24, 64
ears.questions.harmonicinterval), 21, 61	make_resolution() (bird- ears.questions.melodicdictation.MelodicDictationQuestion
1	method), 26, 66
	make_resolution() (bird-
InstrumentalDictationQuestion (class in bird-	ears.questions.melodicinterval.MelodicIntervalQuestion
ears.questions.instrumentaldictation), 23,	method), 27, 68
63	MelodicDictationQuestion (class in bird-
Interval (class in birdears.interval), 28, 54	ears.questions.melodicdictation), 25, 65
INTERVAL_INDEX (in module birdears), 19, 53 INTERVALS (in module birdears), 19, 53	MelodicIntervalQuestion (class in bird-
is_chromatic (birdears.interval.Interval attribute), 28, 54	ears.questions.melodicinterval), 26, 67
is_descending (birdears.interval.Interval attribute), 28, 54	N
K	name (birdears.questions.harmonicinterval.HarmonicIntervalQuestion attribute), 22, 62
KEYS (in module birdears), 19, 53	name (birdears.questions.instrumentaldictation.InstrumentalDictationQuest
1	attribute), 24, 64
L	name (birdears.questions.melodicdictation.MelodicDictationQuestion
log_event() (in module birdears.logger), 29, 55	attribute), 26, 66
M	name (birdears.questions.melodicinterval.MelodicIntervalQuestion attribute), 27, 68
make_chord_progression() (birdears.sequence.Sequence	nearest_tonic() (in module birdears.resolution), 32, 58
method), 34, 60	none() (in module birdears.prequestion), 30, 56
make_input_str() (in module bird- ears.interfaces.commandline), 20, 69	note_and_octave (birdears.interval.Interval attribute), 28,
	note_name (birdears.interval.Interval attribute), 28, 54
ears.questions.harmonicinterval.HarmonicInterva	
method), 22, 62	P
	play() (birdears.sequence.Sequence method), 34, 60
ears.questions.instrumentaldictation.Instrumental	Ipiatatique diestion (birdears.questionbase.QuestionBase
method), 24, 64	method), 31, 57
	$\underline{play\_question()} \ (birdears.questions.harmonic interval. Harmonic Interval Questions) \\$
ears.questions.melodicdictation.MelodicDictation	
method), 26, 66 make pre question() (bird-	play_question() (birdears.questions.instrumentaldictation.InstrumentalDicta
	method), 24, 64  uptaion() (birdears.questions.melodicdictation.MelodicDictationQuestion()
method), 27, 68	method), 26, 66
	play_question() (birdears.questions.melodicinterval.MelodicIntervalQuestic
method), 31, 57	method), 27, 68
make_question() (birdears.questions.harmonicinterval.Harr	
method), 22, 62	ears.questions.harmonicinterval.HarmonicIntervalQuestion

74 Index

```
method), 22, 62
play_resolution()
                                                   (bird-
         ears.questions.melodicdictation.MelodicDictationQuestion
         method), 26, 66
                                                   (bird-
play_resolution()
         ears.questions.melodicinterval.MelodicIntervalQuestion
         method), 27, 68
PreQuestion (class in birdears.prequestion), 29, 55
print instrumental()
                           (in
                                     module
                                                    bird-
         ears.interfaces.commandline), 20, 69
print_question()
                        (in
                                    module
                                                    bird-
         ears.interfaces.commandline), 20, 69
                                    module
                                                    bird-
print_response()
                         (in
         ears.interfaces.commandline), 20, 70
process_key() (birdears.interfaces.commandline.CommandLine
          method), 20, 69
progression_i_iv_v_i() (in module birdears.prequestion),
Q
QuestionBase (class in birdears.questionbase), 30, 56
R
register_prequestion_method()
                                  (in
                                         module
                                                    bird-
         ears.prequestion), 30, 56
register_question_class()
                                       module
                                                    bird-
         ears.questionbase), 31, 57
register_resolution_method()
                                 (in
                                        module
                                                    bird-
         ears.resolution), 32, 58
repeat_only() (in module birdears.resolution), 32, 58
Resolution (class in birdears.resolution), 31, 58
S
scale (birdears.scale.ChromaticScale attribute), 32, 58
scale (birdears.scale.DiatonicScale attribute), 33, 59
ScaleBase (class in birdears.scale), 33, 59
semitones (birdears.interval.Interval attribute), 28, 54
Sequence (class in birdears.sequence), 33, 60
Т
tonic_octave (birdears.interval.Interval attribute), 28, 54
tonic_only() (in module birdears.prequestion), 30, 56
```

Index 75