


Titles

Text

» [PythonInMusic](#)» [PythonInMusic](#)» [FrontPage](#)» [RecentChanges](#)» [FindPage](#)» [HelpContents](#)» [PythonInMusic](#)

## Page

» [Immutable Page](#)» [Info](#)» [Attachments](#)» More Actions: 




## User

» [Login](#)













This page is divided in four sections: Music software written in Python, Music software supporting Python, Music programming in Python, and a category of unsorted (may still fit in the above)

## Music software written in Python


### Audio Players

- »  **Bluemindo** - Bluemindo is a really simple but powerful audio player in Python/PyGTK, using Gstreamer. Bluemindo is a free (as in freedom) software, released under GPLv3, only.
- »  **cplay** - a curses front-end for various audio players
- »  **edna** - an MP3 server, edna allows you to access your MP3 collection from any networked computer. The web pages are dynamically constructed, adjusting to directory structure and the files in those directories. This is much nicer than using simple directory indexing. Rather than directly serving up an MP3, the software serves up a playlist. This gets



passed to your player (e.g. WinAmp) which turns around with an HTTP request to stream the MP3.

- »  **Listen** - Music management and playback for GNOME
- »  **MediaCore Audio/Podcast Player and CMS** - Web based CMS for music management in video, audio and podcast form. All audio, video, and podcasts added to the system are playable from any browser.
- »  **MMA** - Musical Midi Accompaniment. If you follow the above link you will find that Pymprovisator is no longer developed due to the fact that there is this similar, but more powerful GPL Python software.
- »  **Peyote** - Peyote is an audio player with friendly MC-like interface. Peyote is designed specifically for work easy with cue sheets.
- »  **Pymprovisator** - Pymprovisator is a program that emulates the program Band in a Box from PG Music. You can think in it like the electronic version of the books+CD from Jamey Aebersold. You set the basic parameters in a song: title, style, key, chords sequence,... and the program will generate a Midi file with the correct accompaniment. (dev suspended)
- »  **Pympys** - Pympys is the PYthon Music Playing System - a web based mp3/ogg jukebox. It's written in Python and utilises the PostgreSQL database.
- »  **MusicPlayer** - MusicPlayer is a high-quality music player implemented in Python, using FFmpeg and PortAudio.
- »  **Pymserv** - PyMServ is a graphical client for mserv, a music server. It is written in Python using pygtk and gconf to store prefs.
- »  **Pytone** - Pytone is a music jukebox written in Python with a curses based GUI. While providing advanced features like crossfading and multiple players, special emphasis is put on ease of use, turning PyTone into an ideal jukebox system for use at parties.
- »  **Quod Libet** - Quod Libet is a GTK+-based audio player written in Python. It lets you make playlists based on regular expressions. It lets you display and edit any tags you want in the file. And it lets you do this for all the file formats it supports -- Ogg Vorbis, FLAC, MP3, Musepack, and MOD.
- »  **TheTurcanator** - a small midi piano tutor for windows and mac. Includes CoreMIDI wrapper written in pyrex.
- »  **LinuxBand** - LinuxBand is a GUI front-end for MMA (Musical MIDI Accompaniment). Type in the chords, choose the groove and LinuxBand will play a musical accompaniment for you.






## Audio Convertors

- »  **audio-convert-mod** - audio-convert-mod is a simple audio file converter that supports many formats. At just a right-click, you can convert any amount of music files to WAV, MP3, AAC, Ogg and more. audio-convert-mod was designed with the same principles as fwbackups - keeping things

simple.

- »  **SoundConverter** - SoundConverter is a simple audio file converter for the GNOME desktop, using GStreamer for conversion. It can read anything GStreamer has support for, and writes to WAV, MP3, AAC, Ogg or FLAC files.
- »  **Python Audio Tools** - Python audio tools are a collection of audio handling programs which work from the command line. These include programs for CD extraction, track conversion from one audio format to another, track renaming and retagging, track identification, CD burning from tracks, and more. Supports internationalized track filenames and metadata using Unicode. Works with high-definition, multi-channel audio as well as CD-quality. Track conversion uses multiple CPUs or CPU cores if available to greatly speed the transcoding process. Track metadata can be retrieved from FreeDB, MusicBrainz or compatible servers. Audio formats supported are: WAV, AIFF, FLAC, Apple Lossless, Shorten, WavPack, MP3, MP2, M4A, Ogg Vorbis, Ogg Speex, Ogg FLAC, & Sun AU




## Music Notation

- »  **Abjad** - Abjad is a Python API for Formalized Score Control. Abjad is designed to help composers build up complex pieces of music notation in an iterative and incremental way. You can use Abjad to create a symbolic representation of all the notes, rests, staves, nested rhythms, beams, slurs and other notational elements in any score. Because Abjad wraps the powerful LilyPond music notation package, you can use Abjad to control extremely fine-grained typographic details of all elements of any score.
- »  **Frescobaldi** - is a  **LilyPond** music score editor written in Python using PyQt4 and PyKDE4. Clicking a button runs LilyPond on the current document and displays the PDF in a preview window. There are some nice editing tools and a powerful score wizard to quickly setup a template score.
- »  **mingus** - mingus is an advanced music theory and notation package for Python. It can be used to play around with music theory, to build editors, educational tools and other applications that need to process music. It can also be used to create sheet music with LilyPond and do automated musicological analysis.
- »  **Kodou** - Kodou is a small package for algorithmic music notation which runs on top of the LilyPond compiler. It has been designed with the goal of being as minimalistic as possible, yet allowing creation of complex musical structures. It's public API consists of only two objects: a class Part (representing a musical line which itself could be mono- or polyphonic) and a main processing function kodou. Kodou's main strength however lies in it's treatment of time: rhythm is not constructed dependent on duration, but from points (notes/chords) alongside of a timeline and as such as a function of time. Setting variant durations other than the ones imposed by






Kodou is possible.

» see also 'music21' below


## Musical Analysis

- »  **music21** - a toolkit developed at MIT for computational musicology, music theory, and generative composition. Provides expandable objects and methods for most common theoretical problems. Supports music import via MusicXML, Humdrum/Kern, Musedata, ABC, and MIDI, output via MusicXML, Lilypond, and MIDI, and can easily integrate with notation editors (Finale, Sibelius, or MuseScore) and other audio and DAW software (via MIDI).
- »  **pcsets** - Pitch Class Sets are a mathematical model for analyzing and composing music.
- »  **PyOracle** - Module for Audio Oracle and Factor Oracle Musical Analysis.






## Audio Analysis

- »  **Friture** - Friture is a graphical program designed to do time-frequency analysis on audio input in real-time. It provides a set of visualization widgets to display audio data, such as a scope, a spectrum analyser, a rolling 2D spectrogram.
- »  **LibXtract** - LibXtract is a simple, portable, lightweight library of audio feature extraction functions. The purpose of the library is to provide a relatively exhaustive set of feature extraction primitives that are designed to be 'cascaded' to create a extraction hierarchies.
- »  **Yaafe** - Yet Another Audio Feature Extractor is a toolbox for audio analysis. Easy to use and efficient at extracting a large number of audio features simultaneously. WAV and MP3 files supported, or embedding in C++, Python or Matlab applications.
- »  **Aubio** - Aubio is a tool designed for the extraction of annotations from audio signals. Its features include segmenting a sound file before each of its attacks, performing pitch detection, tapping the beat and producing midi streams from live audio.
- »  **LibROSA** - A python module for audio and music analysis. It is easy to use, and implements many commonly used features for music analysis.



## Ear Training

- »  **GNU Solfege** - GNU Solfege is a computer program written to help you practice ear training. It can be useful when practicing the simple and mechanical exercises.

## cSound





- »  **athenaCL** - modular, polyphonic, poly-paradigm algorithmic music composition in an interactive command-line environment. The athenaCL system is an open-source, cross-platform, object-oriented composition tool written in Python; it can be scripted and embedded, includes integrated instrument libraries, post-tonal and microtonal pitch modeling tools, multiple-format graphical outputs, and musical output in Csound, MIDI, audio file, XML, and text formats.
- »  **Cabel** - Visual way to create csound instruments.
- »  **Dex Tracker** - Front end for csound that includes a tracker style score editor in a grid, text editor, cabel tested with Python 2.5.
- »  **Ounk** is a Python audio scripting environment that uses Csound as it's engine.
- »  **Cecilia** is a csound frontend that lets you create your own GUI (grapher, sliders, toggles, popup menus) using a simple syntax. Cecilia comes with a lots of original builtin modules for sound effects and synthesis. Previously written in tcl/tk, Cecilia was entirely rewritten with Python/wxPython and uses the Csound API for communicating between the interface and the audio engine. Version 4.02 beta is the current release.
- » see also 'blue' below


## Audio (Visual) Programming Frameworks

- »  **Peace Synthesizer Framework** - "Peace Synthesizer Framework" is Cross Platform Scriptable Real-Time Visualization & Sound. It has internal and external real-time scriptable visualization and sound generation and also support Nintendo system [Famicom] - like sound Emulation for 8-bits style chiptune music.
- »  **Hypersonic** - Hypersonic is for building and manipulating sound processing pipelines. It is designed for real-time control. It includes objects for oscillators, filters, file-io, soundcard and memory operations.












## Music programming in Python



### Playing & creating sound

- »  **pydub** - Pydub is a simple and easy high level interface based on ffmpeg and influenced by jquery. It manipulates audio, adding effects, id3 tags, slicing, concatenating audio tracks. Supports python 2.6, 2.7, 3.2, 3.3
- »  **audiere** - Audiore is a high-level audio API. It can play Ogg VorbisAU, MP3, FLACAS, uncompressed WAV, AIFF, MOD, S3M, XM, and ITAN files. For audio output, Audiore supports DirectSound or WinMM in Windows, OSS on Linux and Cygwin, and SGI AL on IRIX.
- »  **audiolab** - audiolab is a small Python package (now part of  **scikits**) to

import data from audio files to numpy arrays and export data from numpy arrays to audio files. It uses libsndfile from Erik Castro de Lopo for the underlying IO, which supports many different audio formats: 









<http://www.mega-nerd.com/libsndfile/>

- »  **GStreamer Python Bindings** - GStreamer is a big multimedia library, it is very simple to use it with these python bindings. Many applications rely on it (Exaile, Pitivi, Jokosher, Listen usw.). Online documentation can be found on  <http://pygstdocs.berlios.de/>
- »  **improviser** - Automatic music generation software. Experiments in musical content generation.
- »  **python-musical** - Python library for music theory, synthesis, and playback. Contains a collection of audio wave generators and filters powered by numpy. Also contains a pythonic music theory library for handling notes, chords, scales. Can load, save, and playback audio.
- »  **LoopJam** - Instant 1 click remixing of sample loops, able to boost your creativity and multiply your sample loop library. Remix audio loops on a slice level, apply up to 9 FX to individual slices or create countless versions using LJ's auto-remix feature (jam) which re-arranges the audio loop forming musical patterns.
- »  **Loris** - Loris is an Open Source C++ class library implementing analysis, manipulation, and synthesis of digitized sounds using the Reassigned Bandwidth-Enhanced Additive Sound Model. Loris supports modified resynthesis and manipulations of the model data, such as time- and frequency-scale modification and sound morphing. Loris includes support and wrapper code for building extension modules for various scripting languages (Python, Tcl, Perl).
- »  **MusicKit** - The MusicKit is an object-oriented software system for building music, sound, signal processing, and MIDI applications. It has been used in such diverse commercial applications as music sequencers, computer games, and document processors. Professors and students in academia have used the MusicKit in a host of areas, including music performance, scientific experiments, computer-aided instruction, and physical modeling. PyObjC is required to use this library in Python.
- »  **pyao** - pyao provides Python bindings for  **libao**, a cross-platform audio output library. It supports audio output on Linux (OSS, ALSA, PulseAudio, esd), MacOS X, Windows, \*BSD and some more. There are ready-to-use packages in  **Debian**/ **Ubuntu**, and Audio output is as easy as:  





```
as: import ao; pcm = ao.AudioDevice("pulse"); pcm.play(data)
```
- »  **pyAudio** - **PyAudio** provides Python bindings for PortAudio, the cross-platform audio I/O library. Using **PyAudio**, you can easily use Python to play and record audio on a variety of platforms. Seems to be a successor of fastaudio, a once popular binding for PortAudio
- »  **pyFluidSynth** - Python bindings for FluidSynth, a MIDI synthesizer that



uses SoundFont instruments. This module contains Python bindings for FluidSynth. FluidSynth is a software synthesizer for generating music. It works like a MIDI synthesizer. You load patches, set parameters, then send NOTEON and NOTEOFF events to play notes. Instruments are defined in SoundFonts, generally files with the extension SF2. FluidSynth can either be used to play audio itself, or you can call a function that returns chunks of audio data and output the data to the soundcard yourself.

- »  **Pygame** - Pygame is a set of Python modules designed for writing games. It is written on top of the excellent SDL library. This allows you to create fully featured games and multimedia programs in the Python language. Pygame is highly portable and runs on nearly every platform and operating system. .ogg .wav .midi .mod .xm .mp3. Sound output. midi input and output. Load sounds into numeric and numpy arrays.
- »  **PyMedia** - (Not updated since 2006) **PyMedia** is a Python module for the multimedia purposes. It provides rich and simple interface for the digital media manipulation( wav, mp3, ogg, avi, divx, dvd, cdda etc ). It includes parsing, demultiplexing, multiplexing, coding and decoding. It can be compiled for Windows, Linux and cygwin.
- »  **pyo** - pyo is a Python module containing classes for a wide variety of audio signal processing types. With pyo, user will be able to include signal processing chains directly in Python scripts or projects, and to manipulate them in real time through the interpreter. Tools in pyo module offer primitives, like mathematical operations on audio signal, basic signal processing (filters, delays, synthesis generators, etc.), but also complex algorithms to create sound granulation and others creative audio manipulations. pyo supports OSC protocol (Open Sound Control), to ease communications between softwares, and MIDI protocol, for generating sound events and controlling process parameters. pyo allows creation of sophisticated signal processing chains with all the benefits of a mature, and wildly used, general programming language.
- »  **Zyne** - Zyne is a Python modular synthesizer using pyo as its audio engine. Zyne comes with more than 10 builtin modules implementing different kind of synthesis engines and provides a simple API to create your own custom modules.
- »  **Soundgrain** - Soundgrain is a graphical interface where users can draw and edit trajectories to control granular sound synthesis modules. Soundgrain is written with Python and [WxPython](#) and uses pyo as its audio engine.
- »  **Pyper** - (Not updated since early 2005) Pyper is a musical development environment. It allows you to write Python scripts that generates music in real-time. Pyper uses QuickTime Musical Instruments for synthesis.
- »  **pySonic** - (Not updated since 2005) pySonic is a Python wrapper around the high performance, cross platform, but closed source,  [FMOD sound library](#). You get all the benefits of the FMOD library, but in a Pythonic, object

oriented package.

- »  **PySndObj** - The Sound Object Library is an object-oriented audio processing library. It provides objects for synthesis and processing of sound that can be used to build applications for computer-generated music. The core code, including soundfile and text input/output, is fully portable across several platforms. Platform-specific code includes realtime audio IO and MIDI input support for Linux (OSS,ALSA and Jack), Windows (MME and ASIO), MacOS X (CoreAudio, but no MIDI at moment), Silicon Graphics (Irix) machines and any Open Sound System-supported UNIX. The SndObj library also exists as Python module, aka [PySndObj](#). The programming principles for Python SndObj programming are similar to the ones used in C++. It is also possible to use the Python interpreter for on-the-fly synthesis programming.
- »  **PySynth** - A simple music synthesizer.
- »  **Snack** - (last update: December 2005) The Snack Sound Toolkit is designed to be used with a scripting language such as Tcl/Tk or Python. Using Snack you can create powerful multi-platform audio applications with just a few lines of code. Snack has commands for basic sound handling, such as playback, recording, file and socket I/O. Snack also provides primitives for sound visualization, e.g. waveforms and spectrograms. It was developed mainly to handle digital recordings of speech (being developed at the KTH music&speech department), but is just as useful for general audio. Snack has also successfully been applied to other one-dimensional signals. The combination of Snack and a scripting language makes it possible to create sound tools and applications with a minimum of effort. This is due to the rapid development nature of scripting languages. As a bonus you get an application that is cross-platform from start. It is also easy to integrate Snack based applications with existing sound analysis software.
- »  **AudioLazy** - Real-Time Expressive Digital Signal Processing (DSP) Package for Python, using any Python iterable as a [-1;1] range audio source. Has time-variant linear filters as well as LTI filters using Z-Transform equations like  $1 - z^{-1}$ , as well as analysis (ZCR / zero crossing rate, LPC / Linear Predictive Coding, AMDF, etc.), synthesis (table lookup, ADSR, etc.), ear modeling (Patterson-Holdsworth with gammatone filters and ERB models), and multiple implementation of common filters (lowpass, highpass, comb, resonator), among several other resources (e.g. Lagrange polynomial interpolation, simple converters among MIDI pitch / frequency / string). Works mainly with `Stream` instances for its signal outputs, a generator-like (lazy) iterable with elementwise/broadcast-style operators similar to the Numpy array operators. Integrated with Matplotlib for LTI filter plotting, although it doesn't require Matplotlib nor Numpy for computation, DSP or I/O. Emphasizes sample-based processing while keeping block-based processing easy to be done, this package can also be



seen as a highly enhanced itertools. Pure Python, multiplatform, compatible with Python 2.7 and 3.2+, uses [PyAudio](#) for audio I/O (if needed). Can be used together with Scipy, SymPy, music21 and several other packages, none required for DSP computation based on Python iterables.

- » [sounddevice](#) - This module provides bindings for the [PortAudio](#) library (using [CFFI](#)) and a few convenience functions to play and record [NumPy](#) arrays containing audio signals.

## Community

- » [PythonSound](#) - The Python Sound Project aims to develop a productive community around Python, Csound and other synthesis engines as tools for algorithmic and computer assisted composition of electroacoustic music.









## Csound

- » [CSound](#) / [CsoundAC](#) - Csound is a sound and music synthesis system, providing facilities for composition and performance over a wide range of platforms and for any style of music. The Csound orchestra language features over 1200 unit generators (called "opcodes") covering nearly every sound synthesis method and that the user can combine into "instruments" of unlimited complexity and flexibility. Csound 5 allows Python code to be called from or directly embedded into Csound orchestras. Additionally, the csnd Python extension module wraps the Csound API so that Csound can be embedded into Python applications. CsoundAC (for "Csound Algorithmic Composition") is a GUI front end to Csound with Python scripting and a Python module providing tools for the algorithmic generation or manipulation of Csound scores. [Csound on Sourceforge](#); [Csound-Python](#) and [Csound](#) (some brief tutorials on the OLPC Wiki)
- » [Csound Routines](#) - set of routines to manipulate and convert csound files
- » [PMask](#) - Python implementation of CMask, a stochastic event generator for Csound.


## MP3 stuff and Metadata editors











- » [eyed3](#) - eyeD3 is a Python module and program for processing ID3 tags. Information about mp3 files (i.e bit rate, sample frequency, play time, etc.) is also provided. The formats supported are ID3 v1.0/v1.1 and v2.3/v2.4.
- » [mutagen](#) - Mutagen is a Python module to handle audio metadata. It supports ASF, FLAC, M4A, Monkey's Audio, MP3, Musepack, Ogg FLAC, Ogg Speex, Ogg Theora, Ogg Vorbis, True Audio, WavPack and OptimFROG audio files. All versions of ID3v2 are supported, and all standard ID3v2.4
















frames are parsed. It can read Xing headers to accurately calculate the bitrate and length of MP3s. ID3 and APEv2 tags can be edited regardless of audio format. It can also manipulate Ogg streams on an individual packet/page level.

- »  **ID3.py** - This module allows one to read and manipulate so-called ID3 informational tags on MP3 files through an object-oriented Python interface.
- »  **id3reader.py** - Id3reader.py is a Python module that reads ID3 metadata tags in MP3 files. It can read ID3v1, ID3v2.2, ID3v2.3, or ID3v2.4 tags. It does not write tags at all.
- »  **mpgedit** - mpgedit is an MPEG 1 layer 1/2/3 (mp3), MPEG 2, and MPEG 2.5 audio file editor that is capable of processing both Constant Bit Rate (CBR) and Variable Bit Rate (VBR) encoded files. mpgedit can cut an input MPEG file into one or more output files, as well as join one or more input MPEG files into a single output file. Since no file decoding / encoding occurs during editing, there is no audio quality loss when editing with mpgedit. A Python development toolkit enables Python developers to utilize the core mpgedit API, providing access to mp3 file playback, editing and indexing functionality.
- »  **m3ute2** - m3ute2 is program for copying, moving, and otherwise organizing M3U playlists and directories. m3ute2 can also generate detailed reports about lists of files.
- »  **mmpython** - MMPython is a Media Meta Data retrieval framework. It retrieves metadata from mp3, ogg, avi, jpg, tiff and other file formats. Among others it thereby parses ID3v2, ID3v1, EXIF, IPTC and Vorbis data into an object oriented struture.
- »  **KaaMetadata** Sucessor of MMPython.
- »  **PyID3** - pyid3 is a pure Python library for reading and writing id3 tags (version 1.0, 1.1, 2.3, 2.4, readonly support for 2.2). What makes this better than all the others? Testing! This library has been tested against some 200+ MB of just tags.
- »  **beets** - music tag correction and cataloging tool. Consists of both a command-line interface for music manipulation and a library for building related tools. Can automatically correct tags using the MusicBrainz database.
- » see also: PySonic for programmable MP3 playback






## MIDI Mania




- »  **pygame.midi** - is a portmidi wrapper orginally based on the pyportmidi wrapper. Also pygame.music can play midi files. Can get input from midi devices and can output to midi devices. For osx, linux and windows. New with pygame 1.9.0. *python -m pygame.examples.midi --output*

- »  **pyMIDI** - Provides object oriented programmatic manipulation of MIDI streams. Using this framework, you can read MIDI files from disk, build new MIDI streams, process, or filter preexisting streams, and write your changes back to disk. If you install this package on a Linux platform with alsalib, you can take advantage of the ALSA kernel sequencer, which provides low latency scheduling and receiving of MIDI events. SWIG is required to compile the ALSA extension sequencer extension. Although OS-X and Windows provide similar sequencer facilities, the current version of the API does not yet support them. Some bugs are remaining in this package (for example when trying to delete a track), it has not been updated since 2006. This package is by Giles Hall. A sourceforge download.
- »  **midi.py** - (DEAD LINK) - Python MIDI classes: meaningful data structures that represent MIDI events and other objects. You can read MIDI files to create such objects, or generate a collection of objects and use them to write a MIDI file.
- »  **MIDI.py** - This module offers functions: concatenate\_scores(), grep(), merge\_scores(), mix\_scores(), midi2opus(), midi2score(), opus2midi(), opus2score(), play\_score(), score2midi(), score2opus(), score2stats(), score\_type(), segment(), timeshift() and to\_millisecs(). Uses Python3. There is a call-compatible Lua module.
- »  **PMIDI** - The PMIDI library allows the generation of short MIDI sequences in Python code. The interface allows a programmer to specify songs, instruments, measures, and notes. Playback is handled by the Windows MIDI stream API so proper playback timing is handled by the OS rather than by client code. The library is especially useful for generating earcons.
- »  **portmidizero** - portmidizero is a simple ctypes wrapper for PortMidi in pure Python.
- »  **PyChoReLib** - Python Chord Recognition Library. This is a library that implements the transformation from a list of notenames to a chord name. The system can be taught new chords by example: tell it that ['c', 'e', 'g'] is called a 'C' chord, and using its built-in music knowledge it immediately recognizes all major triads in all keys and all inversions/permutations. Comes with a real-time midi-input demo program (needs [PyPortMidi](#)).
- »  **PyMIDI** - The MIDI module provides MIDI input parsers for Python. Package not updated since 2000.
- »  **PyPortMidi** - [PyPortMidi](#) is a Python wrapper for PortMidi. PortMidi is a cross-platform C library for realtime MIDI control. Using [PyPortMidi](#), you can send and receive MIDI data in realtime from Python. Besides using [PyPortMidi](#) to communicate to synthesizers and the like, it is possible to use [PyPortMidi](#) as a way to send MIDI messages between software packages on the same computer. [PyPortMidi](#) is now maintained at  <http://bitbucket.org/aalex/pyportmidi/>
- »  **PythonMIDI** - The Python Midi package is a collection of classes handling Midi in and output in the Python programming language.









- »  **PySeq** - Python bindings for ALSA using ctypes
- »  **milk** - Superceding the older  **Nam**, milk provides Python with classes representing key MIDI sequencer components: MIDI I/O, EventLists, Plugins and a realtime Flow class. The components can be freely interconnected in a fashion very similar to physical MIDI cabling, however the milk event system is not limited to MIDI events alone; you can define your own extensions should the need arise. Website says it is unpolished and unfinished.
- »  **pyrtmidi** - rtmidi provides realtime MIDI input/output across Linux (ALSA), Macintosh OS X, SGI, and Windows (Multimedia Library) operating systems. It is very fast, has a clean and pythonic interface, and supports virtual ports, according to author Patrick Kidd. In fact it is a wrapper for Gary Scavone's rtmidi from  [here](#), rather than the address on this website:
- »  **rtmidi-python** - Another  **RtMidi** wrapper.
- »  **winmidi.pyd** - A demo? of a Python extension interfacing to the native windows midi libs that developed from  [earlier](#) attempts.
- »  **win32midi** - Some Python samples to demonstrate how to output MIDI stream on MS windows platform. Unlike previous links, these samples playback MIDI by directly calling the Win32 MIDI APIs without an intermediate portable library. It provides a simple player class for playing with MIDI sound using the synthesizer on the soundcard/onboard soundchip. A sample script is provided for testing it out. As it is still a work in progress, bugs are expected.
- »  **midiutil** - A pure Python library for generating Midi files
- »  **Pyknon** - Pyknon is a simple music library for Python hackers. With Pyknon you can generate Midi files quickly and reason about musical proprieties.
- »  **Desfonema Sequencer** - A tracker minded MIDI sequencer for Linux (ALSA) written in Python/PyGTK
- »  **python-music-gen** - Simple library to generate midi patterns from numbers. Useful for building generative music tools.
- »  **fluidsynth-gui** - Graphical User Interface for [FluidSynth](#), and an alternative to Qsynth.

## Other protocols

- »  **OSC.py** - Python classes for  [OpenSoundControl](#) library client functionality. The OSC homepage is at  <http://opensoundcontrol.org>
- »  **Twisted-osc** - OSC Library for Twisted, an event-driven Python framework. It could really be ported to a non-Twisted framework as well, but is currently in the process of possibly become an official part of Twisted.
- »  **aiosc** - Minimalistic OSC communication module using asyncio.

- »  **pyalsaaudio** - This package contains wrappers for accessing the ALSA API (The Advanced Linux Sound Architecture (ALSA) provides audio and MIDI functionality to the Linux operating system) from Python. It is fairly complete for PCM devices and Mixer access.
- »  **pkaudio** - pkaudio is a collection of Python-based modules for midi input, osc communication with supercollider, and pyqt functionality.
- »  **PyJack** - This is a Python C extension module which provides an interface to the Jack Audio Server. It is possible to access the Jack graph to perform port connections/disconnections, monitor graph change events, and to perform realtime audio capture and playback using Numeric Python arrays. This is released under the GPL.

## MAX/MSP & PureData








- »  **mxdublin** - mxdublin is an object oriented framework to generate events in pd and  **max**. pd, short for  **Pure Data**, a graphical Computer Music System written by  **Miller S. Puckette**. mxdublin is a real time Python user environment working within pd/max. It is designed to put logic into a sequence of events. Python has been chosen has the interface language to build and run sequencing objects. Has a prerequisites, the users needs to know a minimal of Python and pd/max.
- »  **net.loadbang.jython** is a package which supports the Python scripting/programming language within MXJ for Max/MSP. We use the Jython interpreter, which allows Python and Java to interact, and gives Python access to the standard Java libraries (as well as any other Java code available to MXJ).
- »  **OpenExposition** - OpenExposition is a library aimed at automatic generation of user interfaces. The programmer only needs to specify what parts of the application need to be exposed to the user, and OpenExposition does the rest. At present, OpenExposition allows access to variables (either directly or through a pair of set/get methods), and class methods. It can construct the user interface graphically (using either the multi-platform FLTK library or Cocoa on Mac OS X), programatically (through Python), aurally (using the speech synthesis and recognition capabilities on Mac OS X), and by building MAX/MSP externals that can then be used in MAX/MSP.
- »  **Py/pyext** - Python script objects is an object library providing a full integration of the Python scripting language into the PD (and in the future Max/MSP) real-time system. With the py object you can load Python modules and execute the functions therein. With pyext you can use Python classes to represent full-featured pd/Max message objects. Multithreading (detached methods) is supported for both objects. You can send messages to named objects or receive (with pyext) with Python methods.
- »  **Purity** is a Python library for Pure Data dynamic patching. The idea is to



be able to harness the power of Pure Data for audio programming without having to use its graphical interface. Python's clear and intuitive syntax can be used with profit in order to create intricate patches with advanced string handling, graphical user interfaces and asynchronous network operations. Purity uses Twisted, an event-driven Python framework.

## Music software supporting Python

### Multitrack Studios

- »  **REAPER** - "Audio Production Without Limits": REAPER is a professional digital audio workstation (DAW) for Windows, OS X and WINE. It comes with an uncrippled evaluation licence and supports advanced audio and MIDI recording, arranging and mixing. The support of several plugin formats (like VST, DX and AU) as well as the extremely flexible routing capabilities make it a powerful production suite. Since version 3.12 REAPER is scriptable with Python, allowing access to internal actions and parts of the API.
- »  **Ableton Live** - Award-winning commercial music creation, production and performance platform for Mac OS and Windows. Live is far and away one of the most interesting and groundbreaking audio recording and sequencing tools to come along in the past five years. Live uses Python internally and an experimental API has been exposed at  [this site](#), and there is a discussion group  [here](#).
- »  **blue** - blue is a Java program for use with Csound. It's interface is much like a digital multitrack, but differs in that there timelines within timelines (polyObjects). This allows for a compositional organization in time that seems to me to be very intuitive, informative, and flexible. soundObjects are the building blocks within blue's score timeline. soundObjects can be lists of notes, algorithmic generators, Python script code, csound instrument definitions, and whatever plugins that are developed for blue. these soundObjects may be text based, but they can be completely GUI based as well.
- »  **Jokosher** - Jokosher is a simple yet powerful multi-track studio. With it you can create and record music, podcasts and more, all from an integrated simple environment. Jokosher is written in Python and uses the GNOME platform and the GTK widget set. The audio engine is powered by GStreamer, and we use Cairo for some of the graphics.
- »  **Jeskola Buzz Modular** - Buzz is a modular audio host that saw the beginning of it's development in 1997 leading the way in open plugin-format hosting (pre-VST) and a unique spin on modular routing using modules called Machines in the form of Generators, Effects, Control machines. Buzz's implementation of Python comes through the use of the Control plugin called PyBuzz, a fully customizable and assignable meta-editor created by Leonard Ritter (creator of the Lunar audio library and the



Linux modular host, Aldrin). A discussion group can be found [!\[\]\(919a2cb85b99741a73c0c31a427236a8\_img.jpg\) here](#)

- » [!\[\]\(38441ceaa711016e0bf2ad46ad394ff4\_img.jpg\) \*\*PyDAW\*\*](#) - PyDAW is a powerful pattern-based DAW and plugin suite for producing electronic music. The UI is written entirely in Python/PyQt, and the audio engine in C.

PythonInMusic (last edited 2019-02-17 13:20:10 by [AmirTeymuri](#))

- » [MoinMoin Powered](#)
- » [Python Powered](#)
- » [GPL licensed](#)
- » [Valid HTML 4.01](#)

[Unable to edit the page? See the FrontPage for instructions.](#)