Simple, Open, Music Recommendations with Python

Sam Thursfield

FOSDEM 2023

About me

- Systems software developer @ Codethink
- Musician and music fan
- Former teacher

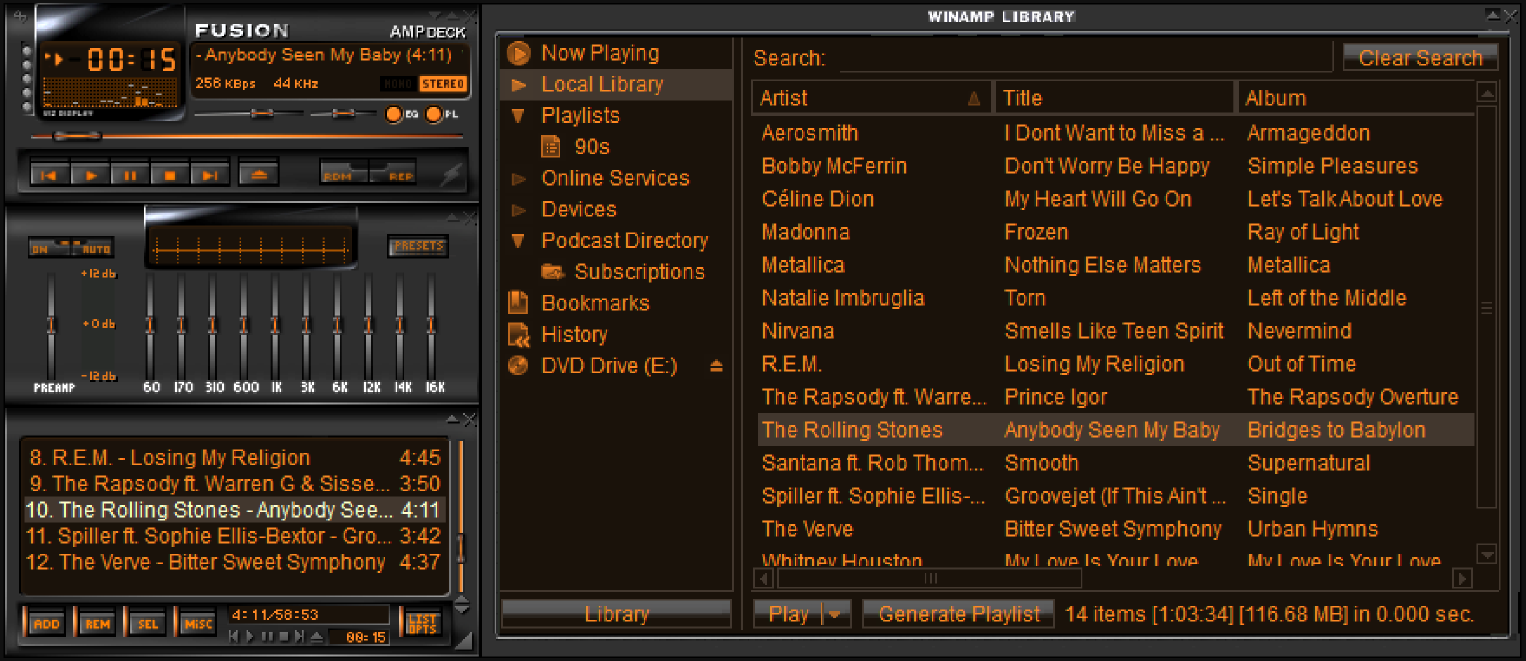
Playlists (1990s)

❌ Difficult to make

✅ Easy to share

Photo: Alamodestuff, Flickr, CC-BY-NC-ND

Playlists (2000s)



- Anybody Seen My Baby (4:11)
256 KBps 44 KHz

8. R.E.M. - Losing My Religion 4:45
9. The Rapsody ft. Warren G & Sisse... 3:50
10. The Rolling Stones - Anybody See... 4:11
11. Spiller ft. Sophie Ellis-Bextor - Gro... 3:42
12. The Verve - Bitter Sweet Symphony 4:37

4:11/58:53 00:15

Artist	Title	Album
Aerosmith	I Dont Want to Miss a ...	Armageddon
Bobby McFerrin	Don't Worry Be Happy	Simple Pleasures
Céline Dion	My Heart Will Go On	Let's Talk About Love
Madonna	Frozen	Ray of Light
Metallica	Nothing Else Matters	Metallica
Natalie Imbruglia	Torn	Left of the Middle
Nirvana	Smells Like Teen Spirit	Nevermind
R.E.M.	Losing My Religion	Out of Time
The Rapsody ft. Warre...	Prince Igor	The Rapsody Overture
The Rolling Stones	Anybody Seen My Baby	Bridges to Babylon
Santana ft. Rob Thom...	Smooth	Supernatural
Spiller ft. Sophie Ellis-...	Groovejet (If This Ain't ...	Single
The Verve	Bitter Sweet Symphony	Urban Hymns
Whitney Houston	My Love Is Your Love	My Love Is Your Love

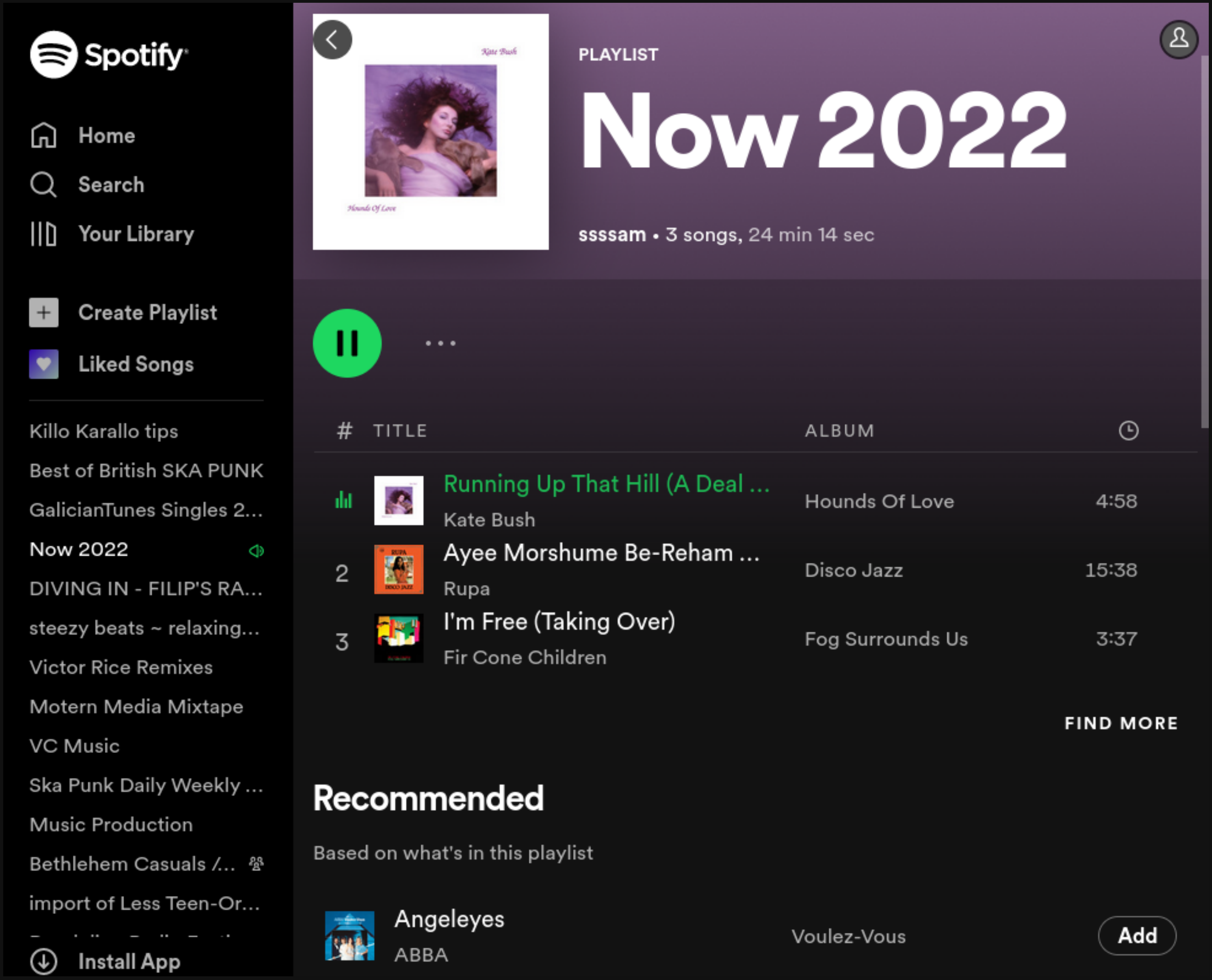
14 items [1:03:34] [116.68 MB] in 0.000 sec.

✅ Easy to make

❌ Difficult to share

Photo: Ariszlo, DeviantArt, CC-BY-NC-SA

Playlists (2010s)



Killo Karallo tips
Best of British SKA PUNK
GalicianTunes Singles 2...
Now 2022
DIVING IN - FILIP'S RA...
steezy beats ~ relaxing...
Victor Rice Remixes
Motern Media Mixtape
VC Music
Ska Punk Daily Weekly ...
Music Production
Bethlehem Casuals /...
import of Less Teen-Or...

Now 2022

ssssam • 3 songs, 24 min 14 sec

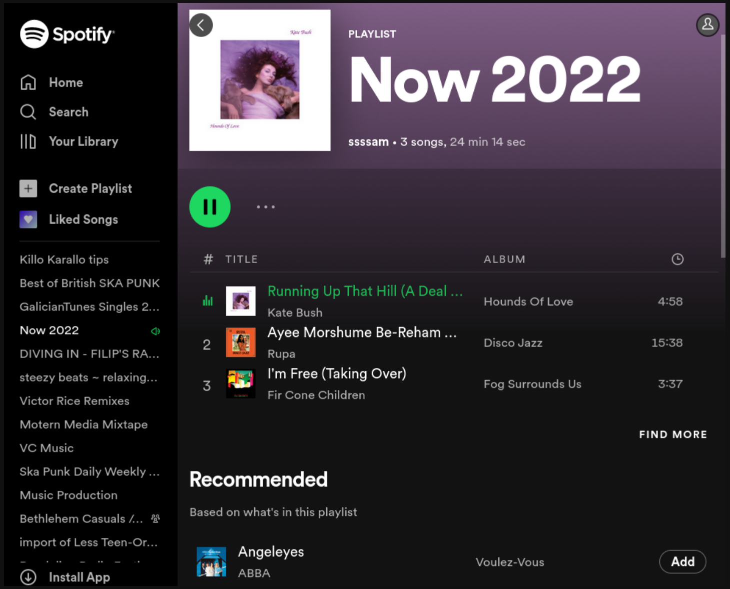
#	TITLE	ALBUM	
	Running Up That Hill (A Deal ... — Kate Bush	Hounds Of Love	4:58
2	Ayee Morshume Be-Reham ... — Rupa	Disco Jazz	15:38
3	I'm Free (Taking Over) — Fir Cone Children	Fog Surrounds Us	3:37

Recommended

Based on what's in this playlist

Angeleyes — ABBA	Voulez-Vous	Add

Playlists (2010s)



Spotify

Home

Search

Your Library

Create Playlist

Liked Songs

Killo Karallo tips

Best of British SKA PUNK

GalicianTunes Singles 2...

Now 2022

DIVING IN - FILIP'S RA...

steezy beats ~ relaxing...

Victor Rice Remixes

Motern Media Mixtape

VC Music

Ska Punk Daily Weekly ...

Music Production

Bethlehem Casuals /...

import of Less Teen-Or...

Install App

PLAYLIST

Now 2022

ssssam • 3 songs, 24 min 14 sec

#	TITLE	ALBUM	🕒
	Running Up That Hill (A Deal ... Kate Bush	Hounds Of Love	4:58
2	Ayee Morshume Be-Reham ... Rupa	Disco Jazz	15:38
3	I'm Free (Taking Over) Fir Cone Children	Fog Surrounds Us	3:37

FIND MORE

Recommended

Based on what's in this playlist

Angeleyes ABBA	Voulez-Vous	Add

✅ Easy to make

✅ Easy to share

🤖 Can generate the playlist for you

Spotify philosophy

- Grow as big as possible ("blitzscaling")
- Pay artists as little as possible
- Optimize for passive listener engagement
- Apply user surveillance and machine-learning to every problem
- All hail the Algorithm

What would the opposite look like?

- Not for profit / DIY
- Encourage building a local music collection
- Link to artist-controlled websites
- Work with open data

Let's get experimenting!

What can we learn from ... **Dynamicland** ?

What can we learn from **Git**?

Git's core ideas were implemented in a month

1. Well-defined data model: blobs, trees, commits, refs.
2. Multi-call binary: small programs that work together
3. "Porcelain" and "Plumbing" layers

Git's design allows...

- a "polyglot" codebase
- easy extensions
- popular websites built around it

Calliope: the same principle for playlists.

- Data model: everything is a playlist
- Multi-call binary `cpe` (also has a Python API)
- Build recommendation pipelines as shell pipelines
- Optimized for ease of maintenance over ease of use.

```
pip3 install calliope-music
```

Core data model

Playlist item

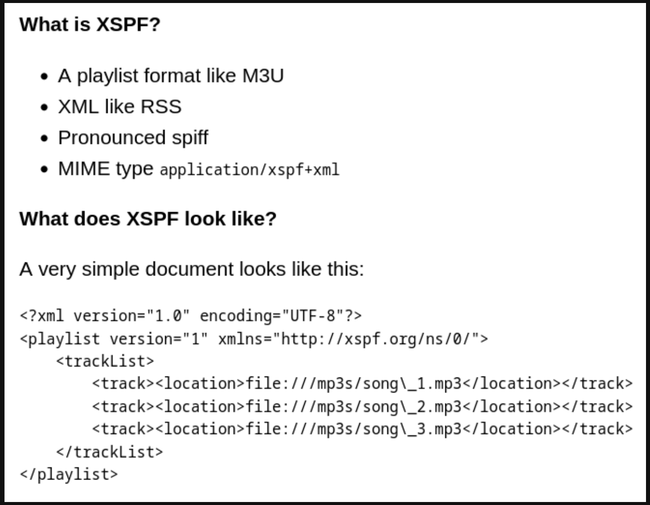
```
{ "creator": "Artist 1", "title": "Great Song" }
```

Playlist

```
{ "creator": "Artist 1", "title": "Great Song" }
{ "creator": "Artist 2", "title": "Banging Tune" }
{ "creator": "Artist 3", "title": "Unpleasant Noise" }
```

This is JSON Lines data so it can be processed one line at a time.

...based on XPSF



What is XSPF?

- A playlist format like M3U
- XML like RSS
- Pronounced spiff
- MIME type `application/xspf+xml`

What does XSPF look like?

A very simple document looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<playlist version="1" xmlns="http://xspf.org/ns/0/">
    <trackList>
        <track><location>file:///mp3s/song\_1.mp3</location></track>
        <track><location>file:///mp3s/song\_2.mp3</location></track>
        <track><location>file:///mp3s/song\_3.mp3</location></track>
    </trackList>
</playlist>
```

Calliope's playlist format is documented here.

Demo: playlist manipulation

```
{ "creator": "Artist 1", "title": "Great Song" }
{ "creator": "Artist 2", "title": "Banging Tune" }
{ "creator": "Artist 3", "title": "Unpleasant Noise" }
```

- Shuffle: `cpe shuffle`
- Export: `cpe export`
- Line-based shell processing
- Data-oriented shell processing

What's next?

Content resolution

XSPF is an intermediate format. We expected a new kind of software called a ***content resolver*** *to do the job of converting XSPF to a plain old list of files or URIs.*

-- XSPF spec

Demo: content resolution

Three songs:

```
{"creator": "Kate Bush", "title": "Hounds of Love"}
{"creator": "Madonna", "title": "Holiday"}
{"creator": "Ana Frango Elétrico", "title": "Saudade"}
```

- Resolve locally: `cpe tracker resolve-content`
- Resolve remotely: `cpe spotify resolve-content`

What's next?

Recommendations

big playlist → algorithm → small playlist

Case study: Special Mix

Special Mix generates a 1 hour playlist of discoveries from a specific year.

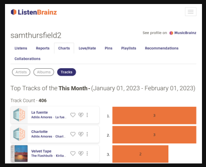
```
python3 -m calliope_examples.special_mix
```

Ingredients:

1. Listening history: `pylistenbrainz`
2. Content resolution: `beets`
3. Track selection: `simpleai`

1. Listening history

- Use Listenbrainz to track music you listen to
- Use Web Scrobbler browser extension to submit listens
- Use pylistenbrainz and `cpe listenbrainz` to access the data



samthursfield2

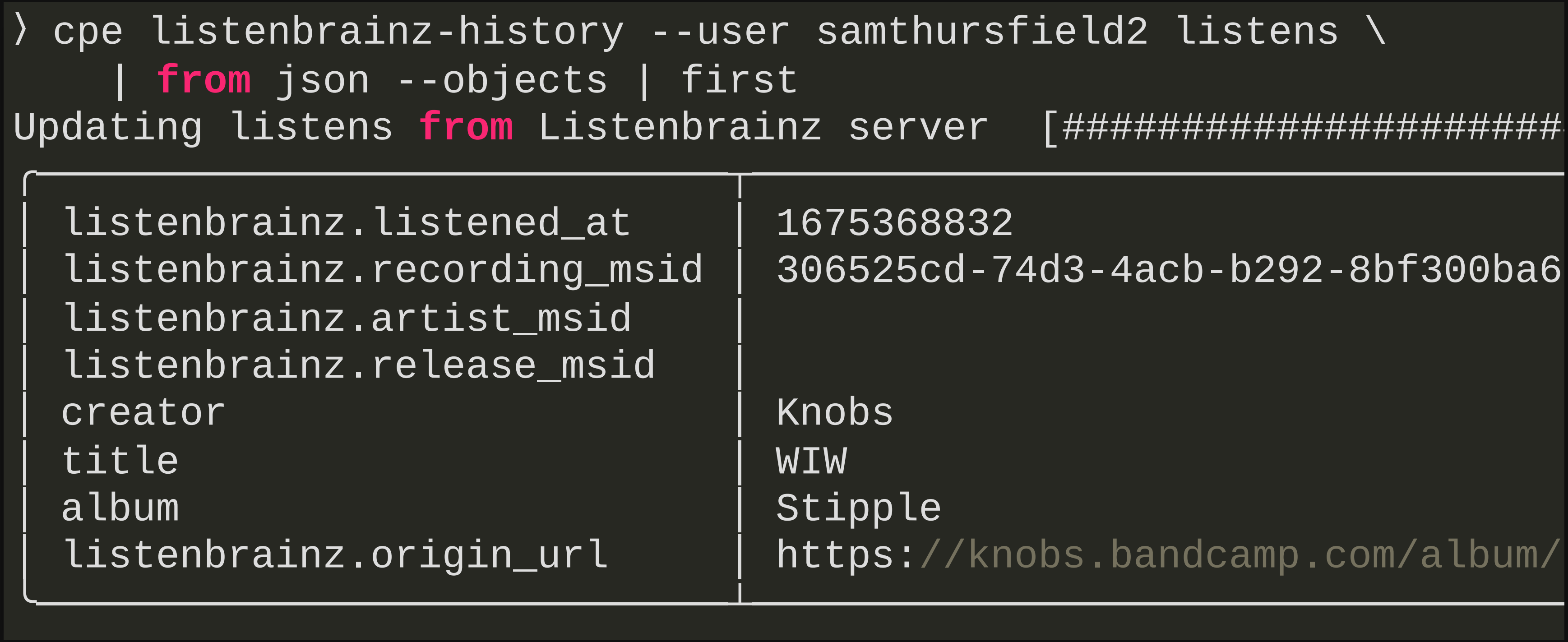
Top Tracks of the **This Month** - (January 01, 2023 - February 01, 2023)

Track Count - **406**

La fuente	Adiós Amores - La fue...	1.	3
Charlotte	Adiós Amores - Charl...	2.	3
Velvet Tape	The Flashbulb - Kirila...	3.	2

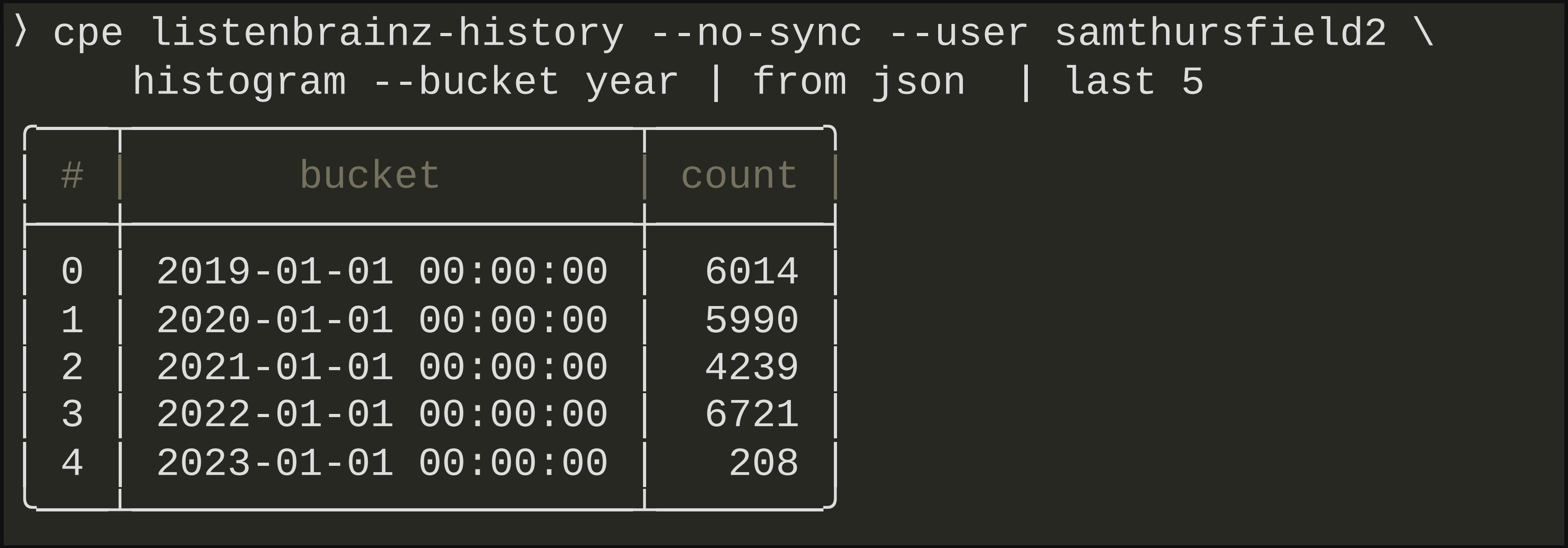
1. Listening history

cpe listenbrainz listens



```
〉 cpe listenbrainz-history --user samthursfield2 listens \
    | from json --objects | first
Updating listens from Listenbrainz server  [#######################
 ╭────────────────────────────┬─────────────────────────────────────
 │ listenbrainz.listened_at    │ 1675368832
 │ listenbrainz.recording_msid │ 306525cd-74d3-4acb-b292-8bf300ba6
 │ listenbrainz.artist_msid    │
 │ listenbrainz.release_msid   │
 │ creator                     │ Knobs
 │ title                       │ WIW
 │ album                       │ Stipple
 │ listenbrainz.origin_url     │ https://knobs.bandcamp.com/album/
 ╰────────────────────────────┴─────────────────────────────────────
```

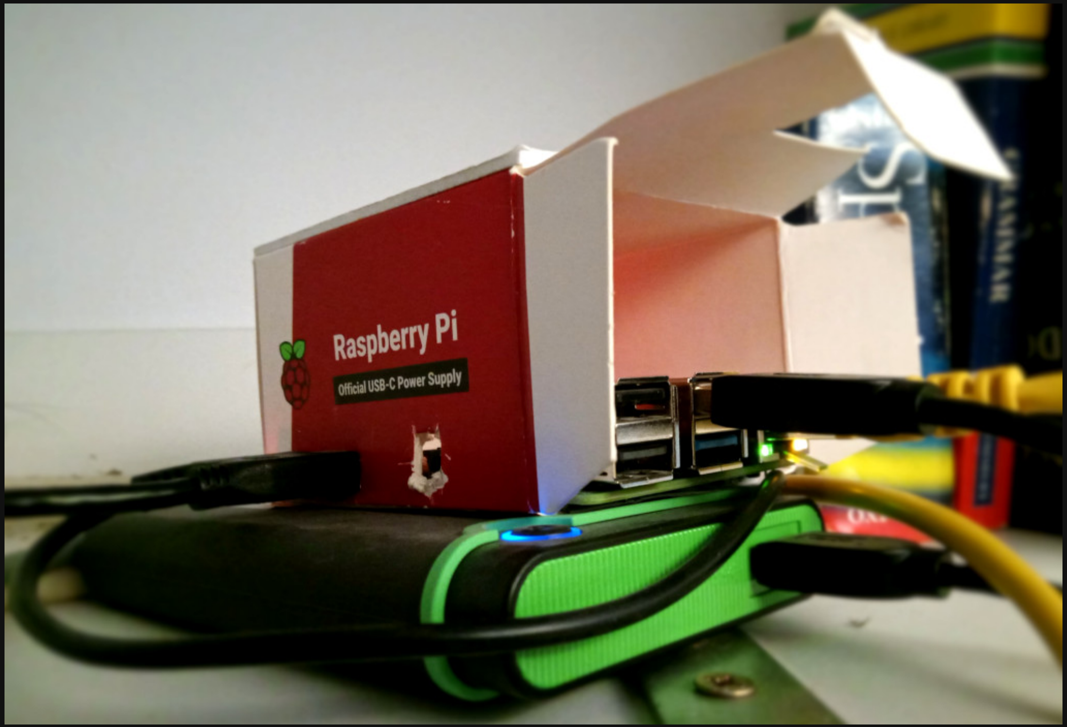
1. Listening history



```
〉 cpe listenbrainz-history --no-sync --user samthursfield2 \
     histogram --bucket year | from json  | last 5
╭───┬─────────────────────┬───────╮
│ # │       bucket        │ count │
├───┼─────────────────────┼───────┤
│ 0 │ 2019-01-01 00:00:00 │  6014 │
│ 1 │ 2020-01-01 00:00:00 │  5990 │
│ 2 │ 2021-01-01 00:00:00 │  4239 │
│ 3 │ 2022-01-01 00:00:00 │  6721 │
│ 4 │ 2023-01-01 00:00:00 │   208 │
╰───┴─────────────────────┴───────╯
```

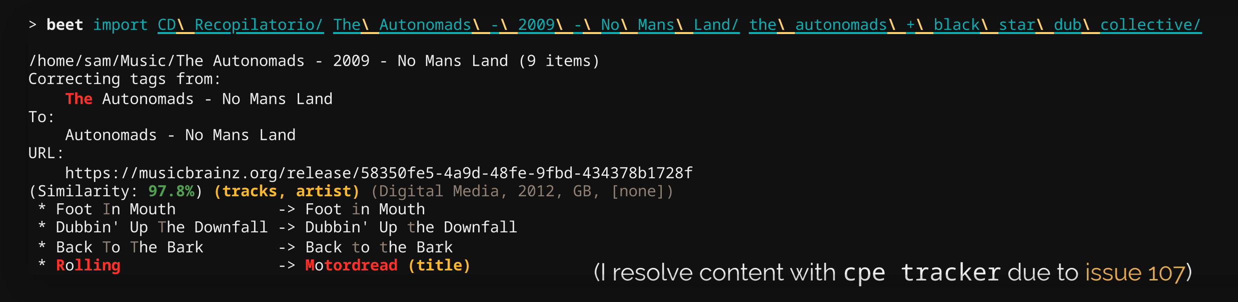
...choose a year, select by `first_listen_date`: now we have a **playlist**

2. Content resolution



2. Content resolution

Beets is the media library management system for obsessive music geeks.



```
> beet import CD\ Recopilatorio/ The\ Autonomads\ -\ 2009\ -\ No\ Mans\ Land/ the\ autonomads\ +\ black\ star\ dub\ collective/

/home/sam/Music/The Autonomads - 2009 - No Mans Land (9 items)
Correcting tags from:
    The Autonomads - No Mans Land
To:
    Autonomads - No Mans Land
URL:
    https://musicbrainz.org/release/58350fe5-4a9d-48fe-9fbd-434378b1728f
(Similarity: 97.8%) (tracks, artist) (Digital Media, 2012, GB, [none])
 * Foot In Mouth           -> Foot in Mouth
 * Dubbin' Up The Downfall -> Dubbin' Up the Downfall
 * Back To The Bark        -> Back to the Bark
 * Rolling                 -> Motordread (title)
```

(I resolve content with `cpe tracker` due to issue 107)

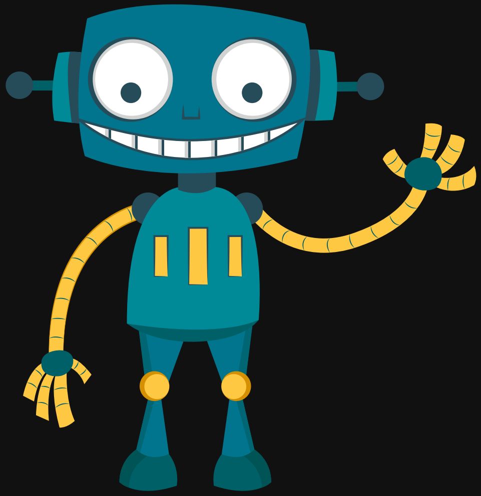
Content resolvers are pluggable and Special Mix can use any...

...now we have a playlist **with track URLs and durations**.

3. Track selection

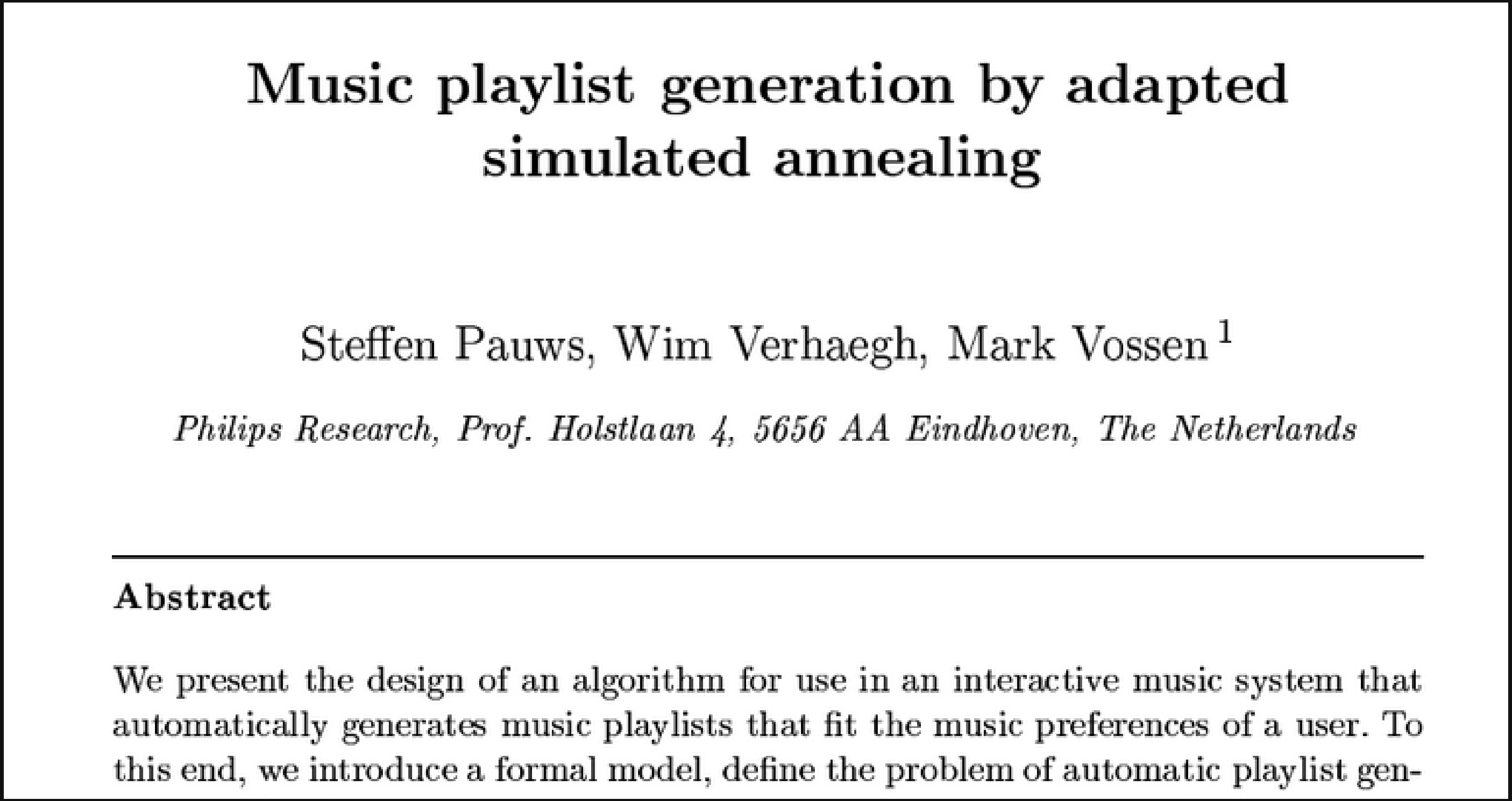
The `cpe select` module wraps the Python `simpleai` package.

You define **constraints** for the playlist, then run a **local search** algorithm to try and find a suitable combination of tracks.



No neural network required.

3. Track selection



Music playlist generation by adapted simulated annealing

Steffen Pauws, Wim Verhaegh, Mark Vossen[1]

Philips Research, Prof. Holstlaan 4, 5656 AA Eindhoven, The Netherlands

Abstract

We present the design of an algorithm for use in an interactive music system that automatically generates music playlists that fit the music preferences of a user. To this end, we introduce a formal model, define the problem of automatic playlist gen-

Pauws, Steffen & Verhaegh, Wim & Vossen, Mark. (2008). Music playlist generation by adapted simulated annealing. Information Sciences. 178. 647-662.

3. Track selection

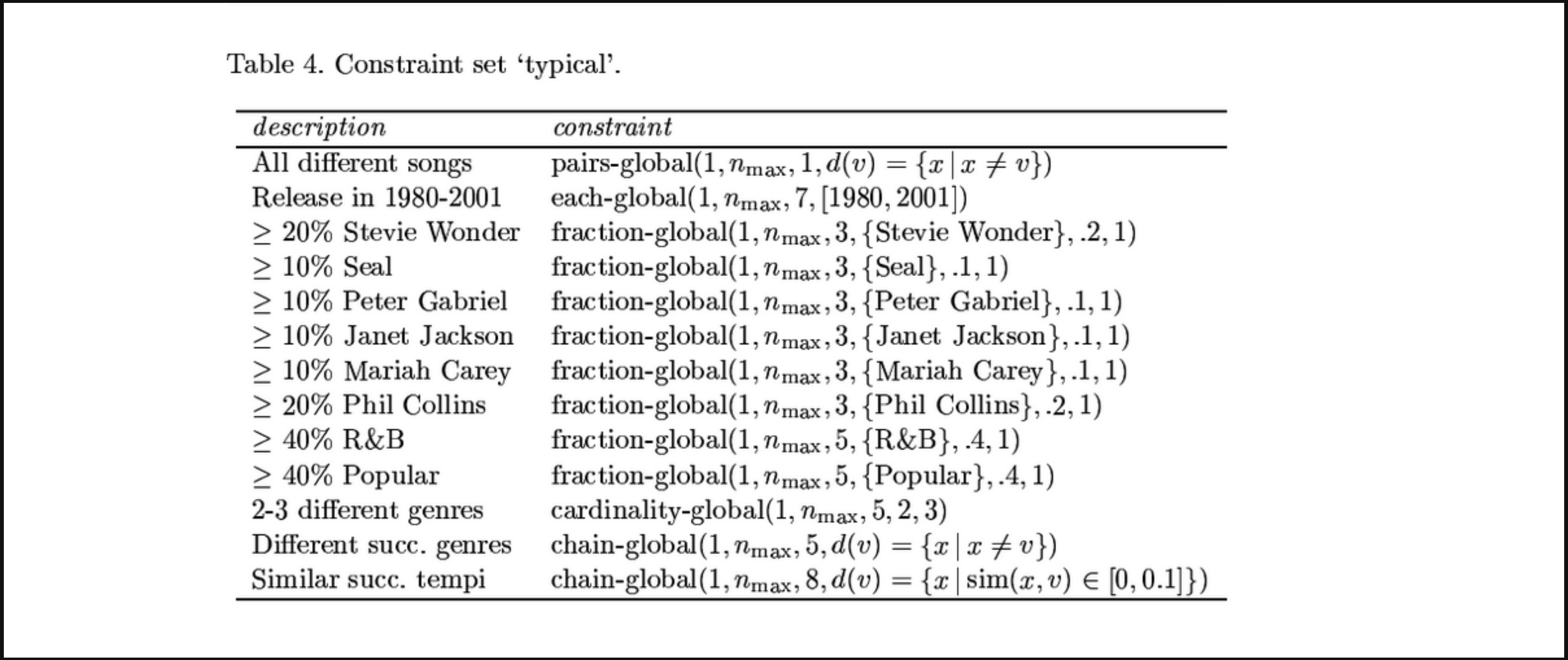


Table 4. Constraint set ‘typical’.

description	*constraint*
All different songs	pairs-global$(1, n_{\max}, 1, d(v) = \{x \mid x \neq v\})$
Release in 1980-2001	each-global$(1, n_{\max}, 7, [1980, 2001])$
$\geq$ 20% Stevie Wonder	fraction-global$(1, n_{\max}, 3, \{\text{Stevie Wonder}\}, .2, 1)$
$\geq$ 10% Seal	fraction-global$(1, n_{\max}, 3, \{\text{Seal}\}, .1, 1)$
$\geq$ 10% Peter Gabriel	fraction-global$(1, n_{\max}, 3, \{\text{Peter Gabriel}\}, .1, 1)$
$\geq$ 10% Janet Jackson	fraction-global$(1, n_{\max}, 3, \{\text{Janet Jackson}\}, .1, 1)$
$\geq$ 10% Mariah Carey	fraction-global$(1, n_{\max}, 3, \{\text{Mariah Carey}\}, .1, 1)$
$\geq$ 20% Phil Collins	fraction-global$(1, n_{\max}, 3, \{\text{Phil Collins}\}, .2, 1)$
$\geq$ 40% R&B	fraction-global$(1, n_{\max}, 5, \{\text{R\&B}\}, .4, 1)$
$\geq$ 40% Popular	fraction-global$(1, n_{\max}, 5, \{\text{Popular}\}, .4, 1)$
2-3 different genres	cardinality-global$(1, n_{\max}, 5, 2, 3)$
Different succ. genres	chain-global$(1, n_{\max}, 5, d(v) = \{x \mid x \neq v\})$
Similar succ. tempi	chain-global$(1, n_{\max}, 8, d(v) = \{x \mid \text{sim}(x, v) \in [0, 0.1]\})$

Each constraint defines a *function* to score a playlist from 0 to 1.

`cpe select` searches for the playlist with the highest score given the constraints.

Using **local search** to find a solution

Example:

- All songs must be 2 to 4 minutes long.
- The playlist must be 10 minutes long.

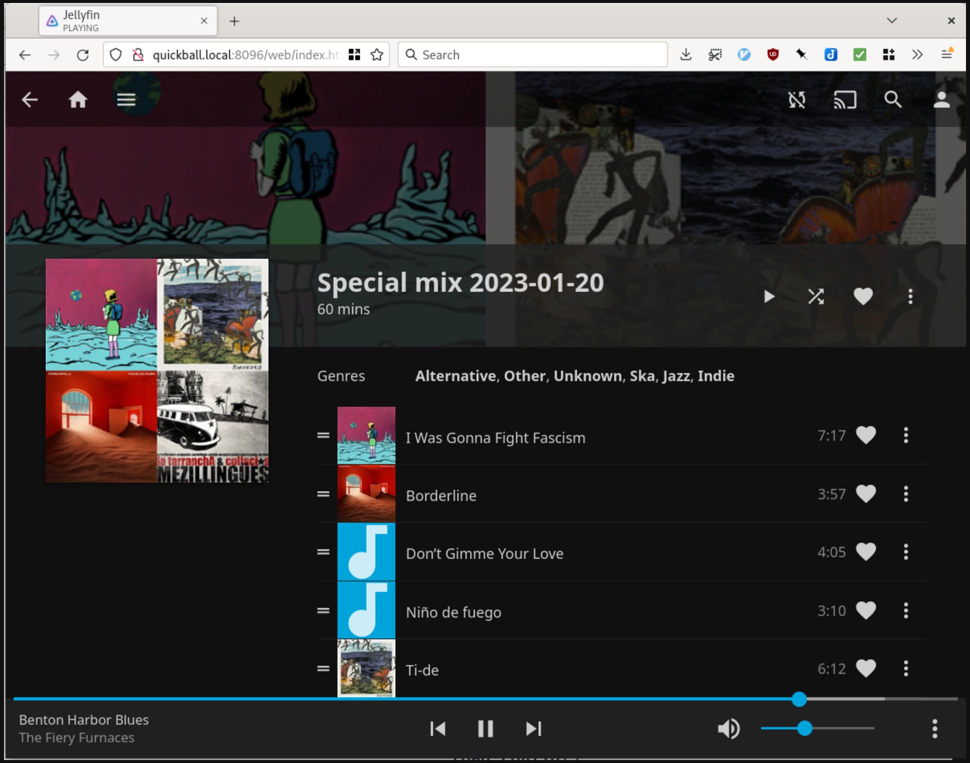
```python
from simpleai.search.viewers import ConsoleViewer, WebViewer
from calliope.playlist import Playlist, PlaylistItem
from calliope.select import ItemDurationConstraint, PlaylistDurationConstraint
import calliope.playlist, calliope.select, calliope.shuffle
import sys
MINUTES = 60

constraints = [
    ItemDurationConstraint(vmin=2 * MINUTES,vmax=4 * MINUTES),
    PlaylistDurationConstraint(vmin=10 * MINUTES,vmax=10 * MINUTES),
]

corpus = Playlist([
    PlaylistItem({"calliope.id": "👸", "title": "Amazing Tune", "duration": 2 * MINUTES}),
    PlaylistItem({"calliope.id": "🎸", "title": "Punk Classic", "duration": 1 * MINUTES}),
    PlaylistItem({"calliope.id": "♬", "title": "Lengthy Opus", "duration": 12 * MINUTES}),
    PlaylistItem({"calliope.id": "🌄", "title": "Ambient Noise", "duration": 7 * MINUTES}),
])

viewer = WebViewer()
input_playlist = calliope.shuffle.shuffle(corpus)
output_playlist = calliope.select.select(input_playlist, constraints, viewer=viewer)

calliope.playlist.write(output_playlist, sys.stdout)
sys.stderr.write(f"Total duration: {sum(item['duration'] for item in output_playlist)}\n")
```

Export to music player

```
> head 'Special mix 2023-01-20.m3u'
#EXTM3U
#PLAYLIST:Discoveries of 2020
../../Music/Soccer96 - Tactics EP [2020]/01 I Was Gonna Fight Fas
../../Music/Tame Impala - The Slow Rush [2020]/03 Borderline.mp3
../../Music/Vic Ruggiero - On the Ragtime [2009]/09 Don't Gimme Y
../../Music/Echte Übersee Records_ Finest Latino Ska and Punk Fro
../../Music/KOKOROKO - KOKOROKO [2019]/02 Ti-de.mp3
```

```
> head 'Special mix 2023-01-20.m3u.log'
DEBUG:calliope.config:Reading config from /home/pi/.config/calliope/calliope.conf
INFO:root:Using history provider: 'listenbrainz_history'
INFO:root:Using resolver: 'tracker'
DEBUG:root:<class 'calliope_examples.special_mix.special_mix.DiscoveredInTimePeriod'>.setup()
DEBUG:root:Choose one period from: ['2005-01-01 00:00:00', '2006-01-01 00:00:00', '2007-01-01 00:00:00', '2008-01-01 00:00:00', '2
INFO:root:Query tracks for period 2020-01-01 00:00:00 -> 2021-01-01 00:00:00
DEBUG:calliope.listenbrainz.listens:SQL:
              WITH
                  listens_with_track_id AS (
                      SELECT (artist_name || ',' || track_name) AS track_id, *
```

Export to music player



Special mix 2023-01-20
60 mins

Genres Alternative, Other, Unknown, Ska, Jazz, Indie

I Was Gonna Fight Fascism	7:17
Borderline	3:57
Don't Gimme Your Love	4:05
Niño de fuego	3:10
Ti-de	6:12

Benton Harbor Blues
The Fiery Furnaces

Recap: Special Mix

```
python3 -m calliope_examples.special_mix
```

Ingredients:

1. Listening history: `pylistenbrainz`
2. Local music collection: `beets`
3. Track selection: `simpleai`

What's next?

Discussion

Project:

- Code: https://gitlab.com/samthursfield/calliope
- Package: `pip install calliope-music`
- Documentation: https://calliope-music.readthedocs.io

Forums:

- Beets forum: "Calliope - antisocial music recommendations"
- Metabrainz forum: "Commandline tool for working with Listenbrainz data"

Keep it simple!