🚫 Limitations

Before you begin reading, I would like to **emphasize** that FLP is a closed and undocumented format. The knowledge needed for understanding the internals is published nowhere, except for a few notes lying around here and there and some existing implementations which I deeply thank for saving my time.

Whatever PyFLP does, is on a best-effort level. Things can go wrong so its always wise to have **backups** and *avoid* **overwriting**.

Most properties are discovered; their representations aren't

You will find almost all the properties you could imagine. However FLP being a binary format stores any and all kinds of stuff as integers. Its actually harder to calculate the formula used for representing stuff like frequency, volume and other such non-linear stuff.

Another thing is musical timings, check #75 for a more info.

Items cannot be added or removed, only modified

I am working on making it possible to add or remove items (like channels, patterns, MIDI notes, arrangements etc.) currently.

Certain things like MIDI notes are simpler to add, but things like patterns and channels will be harder to get correct at. For those things, a **clone** like operation is easier to implement.

> **Note**
>
> It is possible, however (as of PyFLP 2.0.0a4) to add or remove events. If you don't know what they are, you probably shouldn't be handling them. If you are confident about working with events directly, you can very much add new events to effectively do things like adding your own items.

Why is it *slow*?

Slow is a relative term - to some it might not be noticeable at all. Although in my opinion, PyFLP has become way slower since I migrated to `construct`, which provides a lot more benefits than what I did earlier.

`construct` has an opt-in compilation feature which although isn't usable for all kinds of structs, is available for most of them, which gived quite a speed boost for structs that occur a lot (MIDI notes, playlist items to name a few.)

- Due to PyFLP's lazily evaluated nature, most delays don't occur upfront i.e during `pyflp.parse()`.
- Python enums are quite slow, to the point that adding the `f-enum` library patch, reduced parse time by 50%.
- `pyflp._events.EventTree` class' need for `sortedcontainers.SortedList` which is implemented in pure Python.

Difficult to make ports

The current working of PyFLP is non-replicable in most other languages. Descriptors are a Python specific feature I have yet to find anywhere else. Therefore, the possibility of a port that's as clean (and featured) as PyFLP is less. Most languages however have some sort of 3rd party Python interop library available, so its not like PyFLP is completely unuseable from other languages.

A quick search on Github will return some FLP parsers available for other languages, but almost all of them are pretty much unmaintained or archived.

Unit-testing is paramount

Due to the lazy nature of models and their descriptors, each of them should be tested so as to ensure that no changes in the event handling affect or break.

For a long time, I used only a single FLP to test all of PyFLP's API. Things have changed now and I use presets exported from FL Studio itself for the testing of a huge chunk of API to ensure isolation of test results.

The problem is that all the test data comes from FL Studio itself and can be only really validated in the same. That's the reason I usually don't raise any errors event if I know quite surely that, for example a value out of range is set for some property.