LilyPond Quick Reference

2017

Invocation

LilyPond is a command-line tool, and it must be run from inside your computer's *terminal*.

Compile one file: `lilypond my-file.ly`

Compile multiple files: `lilypond one.ly two.ly three.ly`

Get help: `lilypond --help`

Print LilyPond's version: `lilypond --version`

Choose where the output goes: `lilypond -o out.pdf in.ly`

General Syntax

Functions and variables: [1] `\xxxx`

Code blocks: `{ ... }`

Variable assignment: `var = { ... }`

Line comment:

```
% this is a line comment
```

Block comment:

```
%{
    this is a multi-line
    block comment
%}
```

Post-events: [2] `c\foo`

Directions: `c^"up" c_"down" c-"center"`

Scheme lists, pairs, booleans and symbols: [3]

```
#'(1 2 3)  % a Scheme list of three numbers
#'(X . Y)  % a Scheme pair of the variable X and Y
##t ##f    % booleans
#'symbol   % a symbol
```

Includes: `\include "another-file.ly"`

Version: [4] `\version "2.19.0"`

---

[1] When referencing a previously defined function or variable in LilyPond, preceed the name with a slash. This syntax style is taken from LaTeX, a "what-you-see-is-what-you-mean" typesetting program that inspired the original LilyPond developers.

[2] "Post-events" include any commands following a note definition that are pertinent to that note, including articulations, markup, and starting or stopping "spanners" such as beams, slurs and glissandi.

[3] LilyPond embeds *another* programming language, called "Scheme", for advanced control over typography and typesetting behavior. Scheme is a variant of Lisp. LilyPond uses the # character to indicate when the syntax is changing from LilyPond's native syntax into Scheme.

[4] Every *top-level* LilyPond file you compile should start with a version command, otherwise LilyPond will complain.

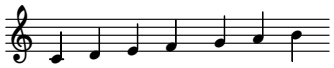
Common Notation

Input language:

```
\language "english"
```

Note names: [5] [6]

```
c d e f g a b
```

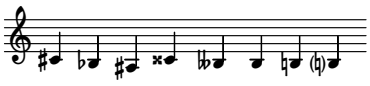


[5] LilyPond's original developers were from the Netherlands, so notes use the Dutch spelling by default. Abjad's developers studied music primarily in the United States, so we use the English spellings instead. LilyPond supports many international spellings.

[6] The octave indications are missing in the input for this example.

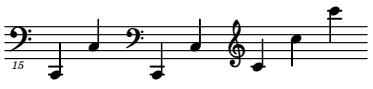
Alterations:

```
cs bf as css bff b b! b?
```



Octaves: [7]

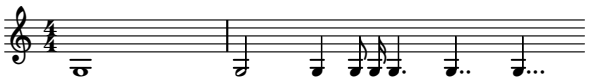
```
c,,, c,, c, c c' c'' c'''
```



[7] LilyPond uses European octave tick notation.

Durations: [8]

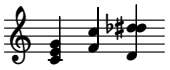
```
g1 g2 g4 g8 g16 g4. g4.. g4...
```



[8] LilyPond allows users to omit durations when entering notes. The last entered duration will be used for the current note. Abjad *always* prints the duration of every note.

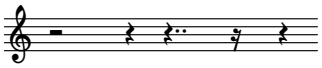
Chords:

```
<c' e' g'>4 <f' c''>4 <d' df'' ds''>4
```



Rests and skips: [9]

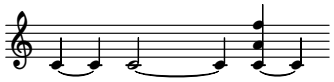
```
r2 r4 r4.. r16 s4 r4
```



[9] Typographic *skips* take up musical time but do not have any graphic representation. They're typically used for positioning notational elements - e.g. intermittent voices - relative to each other.

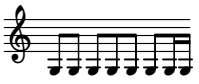
Ties:

```
c'4 ~ c' c'2 ~ c'4 <c' a' f''> ~ c'
```



Beams: [10]

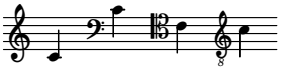
```
g8 g8 g8 [ g g ] g8 [ g16 g16 ]
```



[10] LilyPond applies beams automatically unless you tell it otherwise.

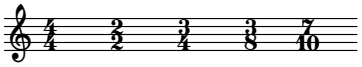
Clefs:

```
\clef "treble" \clef "bass"
\clef "tenor" \clef "treble_8"
```



Time signatures: [11]

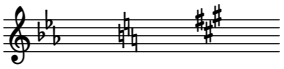
```
\time 4/4 \time 2/2
\time 3/4 \time 3/8
\time 7/10
```



[11] LilyPond doesn't model measures like other notation software. Measures are not containers, and notes, time signatures and bar lines are drawn independently of one another.

Key signatures:

```
\key c \minor
\key c \major
\key b \dorian
```

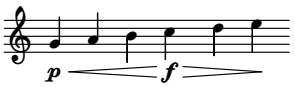


Slurs and phrasing slurs:

```
c' ( a' ) c' \( b ( a ) g f \)
```

Dynamics, hairpins, etc.: [12]

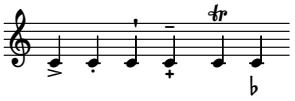
```
g' \p \< a' b' c'' \f \> d'' e'' \!
```



[12] Crescendi and decrescendi will span over music until they encounter another dynamic event. Use `\!` to terminate a dynamic line without the use of final dynamic.

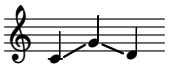
Articulations and text markup:

```
c'4 -> c' -. c' ^| c' ^- _+
c' \trill c' _\markup{ \flat }
```



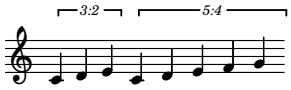
Glissandi:

```
c' \glissando g' \glissando d'
```



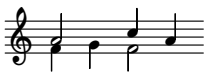
Triplets and tuplets

```
\times 2/3 { c'4 d' e' }
\times 4/5 { c' d' e' f' g' }
```



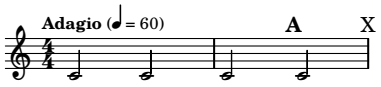
Parallel music:

```
<< { a'2 c''4 a' } \\ { f'4 g'4 f'2 } >>
```



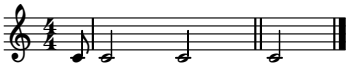
Metronome and rehearsal marks:

```
\tempo "Adagio" 4=60 c'2 c' c'
\mark \default c' \mark "X"
```



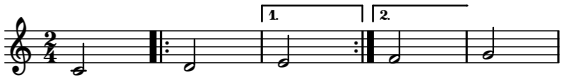
Partial measures, bar line checks, bar lines:

```
\time 2/4
\partial 8 c'8 | c'2 | c'
\bar "||" c' \bar "|."
```



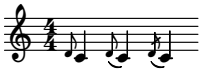
Repeats:

```
\time 2/4
c'2 \repeat volta 2 { d' }
\alternative { { e' } { f' } } g'
```



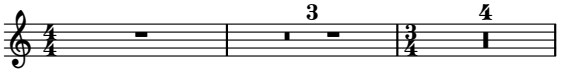
Grace notes:

```
\grace d'8 c'4
\appoggiatura d'8 c'4
\acciaccatura d'8 c'4
```



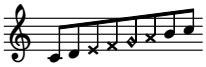
Full/Multi-measure rests:

```
R1 | R1 * 3 | \time 3/4 R2. * 4
```



Typographic overrides:

```
c'8 [ d'
\override NoteHead.style = #'cross e' f'
\once \override NoteHead.style = #'harmonic g'
a' \revert NoteHead.style b' c'' ]
```



Nested Contexts

LilyPond can "nest" musical containers — called "contexts"[13] — together to create examples of parallel music. The following example creates the structure necessary to typeset a typical Bach chorale:

```
\new Score <<
    \new ChoirStaff <<
        \new Staff = "Upper" <<
            \clef "treble"
            \new Voice = "Soprano" { \voiceOne c''2 d''2 }
            \new Voice = "Alto" { \voiceTwo c'2 d'2 }
        >>
        \new Staff = "Lower" <<
            \clef "bass"
            \new Voice = "Tenor" { \voiceOne c2 d2 }
            \new Voice = "Bass" { \voiceTwo c,2 d,2 }
        >>
    >>
>>
```

[13] LilyPond provides a variety of different common contexts out of the box which handle many compositional use-cases, and allows users to define their own. Custom contexts can express their own typographic styles, e.g. smaller musical fonts for cue voices, or drastic graphical changes to create graphic notation. Contexts can also be "named", which supports LilyPond's context concatenation behavior.