

GNU Solfege 3.15.bzr-checkout Gvidlibro de Uzanto

Tom Cato Amundsen

[<tca@gnu.org>](mailto:tca@gnu.org)

Kopirajto © 2005 Tom Eykens

Kopirajto © 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008 Tom Cato Amundsen

Kopirajto © 2009. Felipe CASTRO (fefcas@gmail.com)

Permeso estas garantiita por kopii, disdoni kaj/aŭ modifi tiun ĉi dokumenton laŭ la kondiĉoj de la Ĝenerala Publika Permeso GNU kiel publikigite de Free Software Foundation; kaj versio 3 de la Permeso, kaj (laŭ via volo) iu ajn alia versio. La tuta teksto de la Permeso disponeblas ĉe [Apendico A. GNU General Public License version 3](#).

Enhavo

[1. Enkonduko](#)

[Bonvenon al GNU Solfege](#)

[Misoj](#)

[Rimedoĵ ĉe la reto](#)

[Elŝuti Solfege](#)

[Diskutlistoj](#)

[Fenestron pri preferoj](#)

[Instrumentoj](#)

[Uzanto](#)

[Eksteraj programoj](#)

[Interfaco](#)

[Praktiki](#)

[Son-agordo](#)

[Redaktilo de trejnaro](#)

[Redaktilo de aŭskultada testo-printaĵo](#)

[2. Help-sekcioj por la ekzercoj](#)

[Harmona intervalo](#)

[Agordado](#)

[Klav-komandoj](#)

[Melodia intervalo](#)

[Agordado](#)

[Klav-komandoj](#)

[Kanti intervalon](#)

[Agordi](#)

[Klav-komandoj](#)

[Identigi la akordon](#)

[Klav-komandoj](#)

[Identigi la kadencon](#)

[Multoblaj elekteblaj respondoj por muziko](#)

[Klav-komandoj](#)

[Kanti akordon](#)

[Klav-komandoj](#)

[Ritmo](#)

[Klav-komandoj](#)

[Frapetu la ritmon.](#)

[Diktaĵo](#)

[Klav-komandoj](#)

[Gamoj](#)

[Klav-komandoj](#)

[Intonacio](#)

[Klav-komandoj](#)

[Identigi tonon](#)

[Permana agordado](#)

[Frapoj por minuto](#)

[Klav-komandoj](#)

[Kanti 12 hazardajn notojn](#)

[Klav-komandoj](#)

[Nomi intervalojn](#)

[Diktaĵo de harmona progresio](#)

[3. Music theory](#)

[Gamoj](#)

[Intervals](#)

[Seconds](#)

[Thirds](#)

[Fourth](#)

[Fifth](#)

[Sixths](#)

[Sevenths](#)

[Inverting intervals](#)

[4. Extending GNU Solfege](#)

[Introduction](#)

[Lesson files](#)

[File encoding](#)

[Useful unicode characters](#)

[Comments](#)

[Types](#)

[Global variables](#)

[Lesson file contents](#)

[Header block](#)

[Question block](#)

[music objects](#)

[Functions](#)

[Operators](#)

[The chordvoicing module](#)

[The compareintervals module](#)

[The dictation module](#)

[The elembuilder module](#)

[The element block](#)

[The header block](#)

[The question block](#)

[The harmonicinterval module](#)

[The idbyname module](#)

[Question block](#)

[The identifybpm module](#)

[The idproperty module](#)

[The idtone module](#)
[The melodicinterval module](#)
[The nameinterval module](#)
[The rhythm module](#)
[The rhythmtapping module](#)
[The rhythmtapping2 module](#)
[The singsanswer module](#)
[The singchord module](#)
[The singinterval module](#)
[The twelvetone module](#)
[The mpd module](#)
[Midi instrument names](#)
[Percussion instrument names](#)

[A. GNU General Public License version 3](#)

Listo de Figuroj

3.1.
3.2.
3.3.
3.4.
3.5.
3.6.
3.7.
3.8.
3.9.
3.10.
3.11.
3.12.

Ĉapitro 1. Enkonduko

Enhavo

[Bonvenon al GNU Solfege](#)

[Misoj](#)

[Rimedoĵ ĉe la reto](#)

[Elŝuti Solfege](#)

[Diskutlistoj](#)

[Fenestron pri preferoj](#)

[Instrumentoj](#)

[Uzanto](#)

[Eksteraj programoj](#)

[Interfaco](#)

[Praktiki](#)

[Son-agordo](#)

[Redaktilo de trejnaro](#)

[Redaktilo de aŭskultada testo-printaĵo](#)

Bonvenon al GNU Solfege

Solfege estas [libera](#) aŭskulto-trejna programo. La programo estas parto de la [Projekto GNU](#). Kontrolu [la sekcio nomata “Rimedoĵ ĉe la reto”](#) por informoj pri retlistoj kaj pri kie akiri la lastan version de Solfege.

One of the ideas of this program is that you can extend the program without having to dig into the source code. If you want to practise some special chords or want to practise dictation with some music not included, you can write [lesson files](#) and put them into a `lessonfiles/` subdirectory in your `$HOME` directory. If you create good lesson files, you really should consider contributing them by sending them to the [mailinglist](#) so I can add them to the next version of this program.

Misoj

Raportu misojn al la miso-spurilo ĉe <http://bugs.solfege.org>. Alternative, vi povas sendi mesaĝon al [<bug-solfege@gnu.org>](mailto:bug-solfege@gnu.org). Ĝeneralaj demandoj kaj flikoj devos esti sendataj al [<solfege-devel@lists.sourceforge.net>](mailto:solfege-devel@lists.sourceforge.net).

Bonvolu detaligi viajn rapordojn pri misoj. "Mi trovas erar-mesaĝon en fenestro kiam mi provas lanĉi la programon." ne estas utila por mi. Dum raportado de misoj:

- Rakontu al mi kiun version de Solfege vi uzas. Bonvolu kontroli ĉu pli nova eldono disponeblas. Se vi nur volas uzi stabilajn eldonojn, tiel vi ne devas testi pli novajn disvolgiĝantajn eldonojn.
- Kiun operaciuman sistemon vi uzas? Version?
- Priskribu akurate kion vi faris kiam la eraro okazis.
- Sendu ĝustan kopion de la erar-mesaĝoj. Ili utileblas al la aŭtoro de Solfege eĉ se vi pensas ke ili ŝajnas esti ĉifritaj.

Rimedoĵ ĉe la reto

La hejmpaĝo por Solfege estas <http://www.solfege.org>. Krome, ekzistas pli eta paĝo kun pli fiksa informaro ĉe <http://www.gnu.org/software/solfege/>.

Elŝuti Solfege

La kodumaĵo disponeblas el <http://ftp.gnu.org/gnu/solfege>. Se vi estas aventurema, vi povas provi la malstabilan (erarplena, tamen povas enhavi novaĵojn) eldonojn el <http://alpha.gnu.org/gnu/solfege>. Tiuj ĉi eldonoj eble havas pli da eraroj, sed tiel vi havos ŝancon por provi novajn aferojn kaj trovinte erarojn, vi povos raporti ilin.

Kodumaĵo kaj kelkaj antaŭkompilitaj duumaĵoj disponeblas el http://sourceforge.net/project/showfiles.php?group_id=1465.

Se if uzas Debian vi povas **apt-get install solfege** por elŝuti kaj instali la programon.

Diskutlistoj

[<solfege-announce@lists.sourceforge.net>](mailto:solfege-announce@lists.sourceforge.net)

Tre malalta trafiko, estrata kaj ĝi estos uzata por avizi pri stabilaj eldonoj de Solfege. ([Aliĝo](#) | [Arkivo](#))

[<solfefe-devel@lists.sourceforge.net>](mailto:solfefe-devel@lists.sourceforge.net)

Se vi deziras raporti problemojn pri instalado aŭ funkciado de Solfege, aŭ vi havas demandojn, komentojn aŭ ideojn pri kiel plibonigi Solfege, bonvolu sendi tion ĉi al la diskutlisto anstataŭ uzi la forumon ĉe Sourceforge aŭ rekte kontakti la aŭtoron. Vi povas sendi al solfege-devel sen antaŭ-aliĝo. ([Aliĝo](#) | [Arkivo](#))

[<bug-solfefe@gnu.org>](mailto:bug-solfefe@gnu.org)

La kutima adreso GNU por sendi eraro-raportojn. Tiu ĉi listo estas nuntempe plusendata al [<solfefe-devel@lists.sourceforge.net>](mailto:solfefe-devel@lists.sourceforge.net)

Fenestron pri preferoj

Instrumentoj

Rapido

Set the tempo (beats per minute) for music and arpeggios. These values are used by exercises written with these exercise modules: `compareintervals`, `harmonicintervals`, `idtone`, `melodicinterval`, `singchord`, `singinterval`, `twelvetone`, `rhythm`, `identifybpm`, `nameinterval`. Other exercises will either have the tempo set in the lesson file or on the config page of the exercise.

Preferata instrumento

Difini la instrumenton midi kaj la laŭtecon uzatan por plej parto el la ekzercoj.

Akordaj instrumentoj

Solfege povas uzi tri malsamajn instrumentojn dum ludado de akordoj. Unu por la plej alta tono, unu por la tonoj en la mezo kaj unu por la basa tono. Tio ĉi povas esti utila se vi trovas malfacile aŭskulti apartajn tonojn en akordoj.

Preferata frapinstrumentoj

Difini la frapinstrumenton uzata por nombri antaŭ la ritmaj demandoj, kaj la instrumento por ludi la demandon.

Uzanto

Solfege uzas tiun ĉi informon en kelkaj ekzercoj kie la uzanto devas kanti.

Malplej/plej alta tono kiun la uzanto povas kanti

These spin buttons tell Solfege the highest and lowest tone the user can sing. These values are only considered advisory by the program. If for example the values are set to C to C' and you have configured the program to ask you to sing minor and major tenths, you will have to sing tones outside this range.

Sekso

Solfege bezonas scii ĉu la uzanto estas viro aŭ ino por krei iujn el la demandoj kie la uzanto kantos la respondon. Tio ĉar la vira voĉo sonas unu okton pli malalte ol la ina voĉo.

Eksteraj programoj



Solfege serĉos la PATH (vojo) por programoj kiujn vi indikis en tiu ĉi paĝo. Do vi nur devas enigi la kompletan vojon se la programoj estas instalitaj ekstere de la indikoj de PATH.

Konvertiloj

Metu komandliniojn kiuj konvertas inter malsamaj sonformoj. `%(in)s` estos anstataŭataj per la nomo de la dosiero el kiu ni konvertas. Kaj `%(out)s`, per la nomo al kiu ni konvertas. Ne necesas enmeti `%(out)s` se la programo aŭtomate konservas al nova dosiero kun ĝusta dosier-sufikso.

Sonludiloj

Komand-linioj kiuj povas ludi malsamajn sonformojn. `%s` estos anstataŭataj per la nomo de la ludota dosiero. La dosiernomo estos aldonata ĉe la ĉeno-fino se vi ne inkluzivigos `%s`.

Miksaĵo

Kelketo da ekezcioj uzas la programojn CSound kaj MMA. Lilypond-book estas bezonata por produkti aŭskulto-trejnajn printaĵoj kun testoj, kaj latex estas bezonata kiam la printaĵo devas esti kreata laŭ la formo dvi. Sen latex, vi ankoraŭ povos krei eligon laŭ html.

Se la dosiero enmetita sufiksas per `.py`, tiam la skripto estos plenumata de la sama interpretilo python ankaŭ uzata de Solfege mem.

"Poŝta programo" difinas la komandon kiu ekigas la retpoŝtan programon kiam vi klaku en retadreso, en la gvidlibro de uzantoj.

Interfaco



Montrigi la uzant-gvidilon per ret-foliumilo: Uzu tion ĉi kiel alternativon al la help-foliumilo inkluzivita en la programo.

Reampleksigebla ĉefa fenestro: Ebligas al la uzanto reampleksigi la ĉefa fenestro de solfege.

Elekti lingvon: Vi povas permane elekti la lingvon kiu vi deziras se Solfege ne detektis tion korekte, aŭ se vi volas uzi Solfege kun alia lingvo ol tiu, de via operaciuma sistemo.

Praktiki

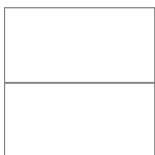


Ne permesi novan demandon antaŭ ol la antaŭa estu solvita: Malebligas la butonon 'nova' ĝin nova demando estu korekte respondita aŭ la uzanto klaku sur "rezigni".

Ripedi demandon se la respondo estis erara: Ludi la sonon refoje kiam la uzanto malkorekte respondas.

Expert mode: Enabling this option in exercises using the [idbyname](#) and [idproperty](#) modules will let you select to practise only a subset of the questions in the lesson file. Practising with expert mode enabled will not save any statistics.

Son-agordo



MIDI-agordo

Ekzistas tri manieroj ludi sonon:

Neniu sono:

Uzu tion ĉi por rafinado aŭ kiam vi portas Solfege. Neniu sono estas ludata, la eventoj midi estas eligataj al stdout.

Uzi aparaton:

La plej bona elekto tie ĉi ordinare estas `/dev/music` ĉar ĝi havas la plej bonan subtenon por frapinstrumetoj. `/dev/sequencer2` ordinare estas simbola ligo al `/dev/music`. Se via sistemo ne havas `/dev/music`, vi povas krei ĝin per tiu komando kiel administranto root (se vi uzas linuxan kernon versio 2.2 aŭ pli modernan):

```
cd /dev mknod music u 14 8
```

En MS Windows tiu ĉi elekto estas nomata `Windows multimedia output`.

Uzi eksteran ludilo MIDI:

Tio ĉi povas esti utila dum portado al sistemoj kiuj ne usas OSS, aŭ se vi havas aĉan sintezilon en via sonkarto kaj deziras uzi timidity.

Redaktilo de trejnaro



La redaktilo de trejnaro ebligas al vi krei dosierojn MIDI/WAV/MP3/OGG kun demandoj, tiel ke vi povos ŝuti ilin al via pda, portebla telefono kaj MP3-ludilo. Solvo-folio estos generita por ke vi printu ĝin. Tiam vi povos igi ke la MP3-ludilo ludu la dosierojn laŭ hazarda ordo, kaj vi povos uzi la solvo-folion por kontroli ĉu vi korekte identigis la muzikon.

Oni uzas la redaktilon de trejnaro por difini kiujn ekzercojn generi. Vi povas konservi vian difinon en dosiero por posta uzado. Ĉiu fojo kiam vi klakos sur **Eksporti**, nova aro da dosieroj estos generata en dosierujo elektita de vi. Vi devos permane alŝuti la generitajn son-dosierojn al via portebla aparato.

La programo ebligas al vi generi demandojn el tiom da lecionoj kiom vi deziras, sed la plej kutima maniero estas generi amason da demandoj el nur unu, aŭ el kelkaj dosieroj.

La programoj uzataj por konverti inter malsamaj dosierformoj estas difinataj en paĝo Gui de la [fenestro de preferoj](#). Bonvolu kontroli la difinojn tie se vi alfrontas problemojn pri konverto de dosieroj MIDI al la formoj WAV, MP3 au OGG.

Klarigo pri kolumno-kapoj

Nombro

La nombro da demandoj por generi el leciondosiero.

Ripeti

Kiom da fojoj ripeti ĉiun demandon.

Prokrasto

Kiom longe prokrasti inter la demandoj. Mezurite laŭ daŭro de kvaronaj notoj.

Redaktilo de aŭskultada testo-printaĵo

Tiu ĉi ilo disponeblas en la menuo **Dosiero**. Uzu ĝin por krei aŭskultadajn testojn por printi sur papero. Solfege produktos du versiojn de la dokumento: unu por la studentoj kompletigi, kaj unu kun la korekta respondo jam skribite.

La butono **Aldoni** ŝprucos menuon kun ĉiuj ekzercoj de la aktiva lerno-arbo el kiuj tiu ilo povas krei ekzercojn: la ekzerco-moduloj [idbyname](#), [melodicinterval](#) kaj [harmonicinterval](#). El leciondosieroj skribitaj por la modulo `idbyname`, nur la muzik-objektoj [chord](#), [rvoice](#) kaj [voice](#) estas subtenataj.

Ĉapitro 2. Help-sekcioj por la ekzercoj

Enhavo

[Harmona intervalo](#)

[Agordado](#)

[Klav-komandoj](#)

[Melodia intervalo](#)
[Agordado](#)
[Klav-komandoj](#)
[Kanti intervalon](#)
[Agordi](#)
[Klav-komandoj](#)
[Identigi la akordon](#)
[Klav-komandoj](#)
[Identigi la kadencon](#)
[Multoblaj elekteblaj respondoj por muziko](#)
[Klav-komandoj](#)
[Kanti akordon](#)
[Klav-komandoj](#)
[Ritmo](#)
[Klav-komandoj](#)
[Frapetu la ritmon.](#)
[Diktaĵo](#)
[Klav-komandoj](#)
[Gamoj](#)
[Klav-komandoj](#)
[Intonacio](#)
[Klav-komandoj](#)
[Identigi tonon](#)
[Permana agordado](#)
[Frapoj por minuto](#)
[Klav-komandoj](#)
[Kanti 12 hazardajn notojn](#)
[Klav-komandoj](#)
[Nomi intervalojn](#)
[Diktaĵo de harmona progresio](#)

Harmona intervalo



Tiu ĉi ekzerco estas unu el tiuj kiujn vi povas uzi por trejni intervalojn. La koncepto estas tre simpla: Vi premas la butonon **Nova intervalo** por ludi hazardan intervalon, kaj tiam vi devos diri kiu intervalo ĝi estis.

Se vi uzas la butonan interfacon, do vi povas dekstre-klaki sur la butonoj por aŭskulti la intervalon, kiun ili reprezentas.

Agordado

En la agord-paĝo de la ekzerco, ekzistas falmenuo kie vi povas elekti malsamajn manierojn respondi la demandon. Nune ekzistas fortepiano, gitaro, basgitaro kaj malmultaj tipoj de akordionoj aldone al la ordinaraj buton-interfaco. Sube estas bildo montranta la interfacon por fortepiano.



Klav-komandoj

- Nova intervalo: **Alt+n**
- Ripeti: **Alt+r**
- Ripeti melodian: **Alt+m**
- Rezigni: **Alt+R**

Minora duto: 1	Perfekta kvarto: 2	Maĵora seso: 3	Minora naŭto: 4
Maĵora duto: q	Tritono: w	Minora septo: e	Maĵora naŭto: r
Minora trito: a	Perfekta kvinto: s	Maĵora septo: d	Minor tenth: f
Maĵora trito: z	Minora seso: x	Perfekta okto: c	Major tenth: v

Melodia intervalo



Tiu ĉi ekzerco kreas hazardajn intervalojn kaj vi devos provi identigi ilin.

Se vi uzas la butonan interfacon, do vi povas dekstre-klaki sur la butonoj por aŭskulti la intervalon, kiun ili reprezentas.

Agordado

En la agord-paĝo de la ekzerco, ekzistas falmenuo kie vi povas elekti malsamajn manierojn respondi la demandon. Nune ekzistas fortepiano, gitaro, basgitaro kaj malmultaj tipoj de akordionoj aldone al la ordinaraj buton-interfaco. Sube estas bildo montranta la interfacon por fortepiano.



Klav-komandoj

- Nova demando: **Alt+n**
- Ripeti: **Alt+r**
- Rezigni: **Alt+R**

Minora duto: 1	Perfekta kvarto: 2	Maĵora seso: 3	Minora naŭto: 4
Maĵora duto: q	Tritono: w	Minora septo: e	Maĵora naŭto: r

Minora trito: a	Perfekta kvinto: s	Maĵora septo: d	Minor tenth: f
Maĵora trito: z	Minora seso: x	Perfekta okto: c	Major tenth: v

Kanti intervalon

En tiu ĉi ekzerco, Solfege montrigos unu aŭ pli intervalojn, kaj vi devos kanti ilin. Malfeliĉe, ankoraŭ ne eblas kanti kontraŭ mikrofonon kaj lasi Solfege decidi ĉu vi kantis korekte, do vi devos decidi per vi mem ĉu vi korekte kantas aŭ ne.

Agordi

La programo provos krei demandon kie ĉiuj tonoj estos ene la rango, kiun la uzanto kapablas kanti, kiel agordite en la [fenestro de prefereroj](#). Kelkfoje, ne eblas fari demandon ene de tiu rango, ekzemple kiam temas pri multaj intervaloj kie ĉiuj ili supreniras.

Klav-komandoj

- Nova intervalo: **Alt+n**
- Nova intervalo, lasta estis korekta: **Alt+n**
- Nova intervalo, lasta estis malkorekta: **Alt+m**
- Ripeti la unuan tonon **Alt+r**
- _Ludi la respondon **Alt+l**
- Ludi la lastan tonon **Alt+l**

Identigi la akordon

La celo de tiu ĉi ekzerco estas identigi la akordon ludatan.

Ekigu la ekzercon per klako sur **Nova**. Tiam Solfege ludos akordon, kaj vi devos identigi ĝin per klako sur unu el la butonoj sub la malplena liniaro.

Se vi konjektas korekte, la programo montrigos akordon en la liniaro kaj fulmetos la mesaĝon "Korekte" en la stat-breto. Tiam vi povos klaki la butonon **Nova** por havi novan demandon.

Se vi konjektas malkorekte la mesaĝo "Malkorekte" estos montrata en la stat-breto.

Klav-komandoj

- Nova akordo: **Alt+n**
- Ripeti: **Alt+r**
- Ripeti arpeĝon: **Alt+a**
- Rezigni: **Alt+R**

Identigi la kadencon

Identigu la kadencon per klaku sur la butono kun ties nomo. Kiel agnoskite en [miso #5](#) ni bezonas kadencajn ekzercojn en Solfege.

En tiu ĉi eldono de Solfege, ni nur havas unu ekzercon kun kadencoj majore. En tiu ĉi ekzerco, majora gamo estas ludata por difini la tonikon. Eble tio estas tro malmulte? Ĉu ni bezonas kompletan I-IV-V-I antaŭ ol demandi? Aŭ eble pli bone estus skribi veran muzikon, kiu finiĝas per la kadenco kiun ni deziras trejni? Tioj estas aferoj kiujn ni bezonas decidi antaŭ ol lanĉi 3.12.0. Komentoj kaj muziko povas esti aldonataj al la [miso #5](#).

Multoblaj elekteblaj respondoj por muziko



Tiu ĉi paĝo estas ĝenerala helpo-paĝo por ĉiuj ekzercoj skribitaj uzante la ekzerc-modulon `idproperty`. Ekzercoj ordinare skribas pli specifan help-tekston ol tiu ĉi.

La suba ekranbildo montras unu ekzemplon pri kiel ekzerco povas ŝajni.

La ekzerco montrigos demandon kaj ludos iun muzikon, kaj vi devas elekti unu respondon el ĉiu kolumno en la buton-tabelo. Kiam vi elektis la korektan respondon en kolumno, la etikedo iĝos dika, kaj la mesaĝo "Korekte" fulmetos en la stat-breto.

La ekzerco havos **Ripeti arpeĝon** se unu aŭ pli el la demandoj estas de la tipo, kiu la programo povas ludi arpeĝe.

Klav-komandoj

- Nova akordo: **Alt+n**
- Ripeti: **Alt+r**
- Ripeti arpeĝon: **Alt+a**
- Rezigni: **Alt+R**

Kanti akordon



Se vi kondukas koruson, vi devas kanti la komencajn tonojn por malsamaj voĉoj, kaj se vi ne havas fortepianon proksime, vi devas uzi agordilon. Se vi estas viro, vi kantas la tonojn por virinoj, unu okto pli profunde, kaj male male.

La programo ludos la tonon A (440 hz) por vi, kaj ĝi montrigos akordon por ke vi kantu. Solfege ankoraŭ ne subtenas mikrofonon, do estas vi mem kiu decidus ĉu via respondo estas korekta aŭ ne.

Klav-komandoj

- Nova: **Alt+n**
- 440hz: **Alt+z**
- Ripeti respondon: **Alt+p**

Ritmo



La programo ludas hazarde generitajn ritmojn, kaj la uzanto devos reprodukti la ritmon. La uzanto enigas la ritmon per klakado sur la butonoj reprezentantaj malsamajn ritmerojn.

Kiam vi enigas sufiĉe da ritmero, Solfege kontrolos vian respondon. Se ĉio estos korekte ĝi montrigos feliĉan mienulon, male malĝojan, kaj ĉiuj eraraj ritmoj estos markitaj.

Se iu el viaj respondoj estos erara, ĉio ekde la unua malkorekta ero estos forigita tiam (tenante iun ajn korektan ritmon ĉe la komenco de via respondo), kiam vi klakos sur la malĝoja mieno, kaj kiam vi klakos sur la ritmo-butonoj supre de la paĝo.

Vi povas klaki la butonon 'Ludi' por aŭdi vian sugeston.

La demandoj prezentataj de tiu ĉi ekzenco aktuale estas farataj per elekto de ritmero hazarde. Tiu ĉi ne estas la plej bona maniero, kaj ni esperas ke pli interesa maniero generi demandojn estos disponebla en venonta lanĉo.

Klav-komandoj

- Nova: **Alt+n**
- Ripeti: **Alt+r**
- Rezigni: **Alt+R**
- Retropaŝo: **Backspace**

Frapetu la ritmon.

La programo ludos hazarde generitan ritmon, kaj la uzanto devos reprodukti ĝin. La uzanto enigas la ritmon per frapetado sur la butono kun etikedo *Frapetu tie ĉi*.

Diktaĵo



Tiu ĉi ekzerco estas nomata diktaĵo, tamen se la bezonataj leciondosieroj estos skribitaj ĝi povos estis uzata laŭ pluraj manieroj:

- Vi povas lasi ke Solfege ludu iun muzikon por vi, kiun vi devas skribi surpapere. Klaku sur la butonoj kun bildo pri kvarona noto por ripeti pli malgrandajn partojn de la muziko. Vi devas klaku sur la butonon Montri kaj mem kontroli viajn notojn por konstati ĉu vi iel eraris.
- Vi povas uzi tiun ĉi ekzercon por trejni tuj-kantadon: Kiam vi ekigas la ekzercon, premu Montri kaj tiam provu kanti la muzikon. Tiam vi povas uzi la butonon Ludi la tutan muzikon aŭ la butonojn kun kvaronaj notoj por lasi ke la programo ludu la muzikon. Vi devas mem decidi ĉu vi sukcesis aŭ ne.

Klav-komandoj

- Ludi la tutan muzikon: **Alt+l**
- Montri: **Alt+m**

Gamoj

Gamoj estas malsimpla afero. Ekzemple, la greka lida (C-D-E-F-G-A-B-C) estas malsama ol la mezepoka kaj moderna lida (C-D-E-F#-G-A-B-C). Vi povas legi pri ĉiuj gamoj uzataj en GNU Solfege [ĉi tie](#).

Solfege havas tri variaĵojn de gam-ekzercoj ĝis nun.

- Solfege ludos gamon, kaj vi devos identigi ĝin per klaku sur la butono kun sama nomo de la gamo.
- Solfege ludos gamon, kaj vi devos identigi la strukturon de la gamo. Oni prezentos al vi aron da butonoj kun etiketoj numeraj '1', '2' kaj '3'. Tiuj ĉi numeroj reprezentas la intervalojn minora duta, maĵora duta kaj minora trita kiuj estas inter la tonoj de la gamo.
- Solfege ludos gamon, kaj vi devos identigi la gradon. Ekzemple, Solfege povas preni la naturan minoran gamon, kaj ludi ĝin el iu ajn el la tonoj en la gamo, kaj vi devos diri ekde kiu tono ĝi komenciĝas.

Klav-komandoj

- Nova: **Alt+n**

- Ripeti: **Alt+r**
- Ripeti malrapide: **Alt+m**
- Reziĝni: **Alt+R**

Intonacio



En tiu ĉi ekzerco, Sofege ludos intervalon, kaj vi devos diri kiel la intervalo estas intonaciata. Vi faras tion per klako sur unu el la butonoj kun etikedo 'tro eta', 'pura' aŭ 'tro granda'. Eblas ankaŭ ke unu el tiuj ĉi tri butonoj manku.

Klav-komandoj

- Nova: **Alt+n**
- Ripeti: **Alt+r**
- Reziĝni: **Alt+R**
- Montri: **Alt+m**

Identigi tonon



Tio ĉi estas miksa ekzerco por memoro kaj intervaloj. Iuj kredas ke tiu ĉi tipo de ekzerco donas al vi absolutan sonsensan, sed mi ne kredas tion.

La bazo estas: la programo ludas tonon kaj vi devas identigi ĝin per komparado al la lasta tono ludita por vi.

Por enkonduki vin la programo ludos unu tonon kaj montrigos ties nomon ĉe la statbreto. Vi identigas la tonojn per klako sur la klavaro de la fortepiano aŭ uzante la klavkomandojn kiuj estas la literoj skribitaj en ĉiu klavo.

Deskre-klaku sur la klavaro de la fortepiano por aŭskulti noton sen fakte konjekti ĝin. (Iuj nomas tion friponado...)

Permana agordado

You can configure this exercise as you like if you select select “Configure yourself” front the default front page.

Ekzistas pluraj manieroj per kiu vi povas uzi tiun ĉi ekezcio. Persone, mi ne uzadas multe tiun ĉi ekzercon, kaj la suba sekcio temas nur pri sugestoj.

Pli kaj pli notoj

Start with only the notes c-d-e at weight 1. When your score is at least 96% correct, you add the tone f and continue. Exercises are defined that will add one and one tone until you practise with all 12 tones.

Peza A

Agordu la tonon 'a' al pezo 11 (aŭ pli granda) kaj la resto de la tonoj al pezo 1. Tiel la programo ludos la tonon 'a' tre ofte, tiel ke vi parkeros la tonon, kaj de tiam vi uzos 'a' kiel referencan tonon por identigi la aliajn tonojn. Kiam vi trejnadis dum iom da tempo, vi povas malpliigi la pezon de 'a' por pli malfaciligi la ekzercon.

Agordi

If run from “Configure yourself”, on the top of the config page you tell the program how important the different tones are. If you for example give the tone a 11 points and the rest 1 point each, then $(11+11*1)/11*100 = 50\%$ of the random tones will be an a.

Sub tio vi elektas el kiuj oktoj la hazardaj tonoj povas veni.

Tiam vi povas elekti ĉu Solfege devos liveri al vi novan demandon aŭtomate kiam vi solvis la pasintan.

En la suba kadro vi povas difini memklarigajn opciojn pri kio okazos kiam vi erare respondos.

The keyboard shortcuts can be configured from the [preferences window](#).

Frapoj por minuto

La programo ludos tempindikon, kiel metronomo. Vi devos provi konjekti kiom frapojn por minuto estas ludata. Ĉiu butono reprezentas unu tempon, kaj la programo nur ludos laŭ tempindikoj kiuj havas butonon kun dikeca teksto. Dekstre-klaku sur butonoj por ŝanĝi la staton de tempindiko.

Rimarko: la ritmo dependas je la funkcio `gtk_timeout_add` por ludi la ritmon, do ĝi ne estas tiomege ĝusta.

Klav-komandoj

- Nova tempindiko: **Alt+n**
- Rezigni: **Alt+R**

Kanti 12 hazardajn notojn

En tiu ĉi ekzerco, la programo montrigos ĉiujn 12 tonojn en la gamo laŭ hazarda ordo kaj ĝi ludos la

unuan. Tiam vi devos kanti ĉiujn notojn kaj kontroli ĉu la lasta noto kongruas. Do, tio ĉi similas pli al ekzameno pri tuja elkantado ol al ekzerco pri lernado kiel kanti la intervalojn. Por tio, vi devos provi iun el la aliaj intervalaj ekzercoj.

Klav-komandoj

- Nova: **Alt+n**
- Ludi unuan noton: **Alt+u**
- Ludi lastan noton: **Alt+l**
- Ludi ĉion: **Alt+o**

Nomi intervalojn

En tiu ĉi ekzerco, Solfege montrigos kaj ludos intervalon, kaj vi devos identigi la intervalon. Tio ĉi estas ekzerco pri muzik-teorio, kaj ne aŭskulto-trejna ekzerco. Por lerni kiel nomi intervalojn vi devos legi [la sekcio nomata “Intervals”](#).

Vi identigas la intervalon per klako sur unu butono, kiu montras la specifan nomon kaj la ĝeneralan nomon.

Diktaĵo de harmona progresio

En tiu ĉi ekzerco, Solfege ludos iun muzikon, kaj vi devos klaki sur la butonojn por konstrui reprezenton de harmonaj progresioj en la ekzerco.

Ĉapitro 3. Music theory

Enhavo

[Gamoj](#)

[Intervals](#)

[Seconds](#)

[Thirds](#)

[Fourth](#)

[Fifth](#)

[Sixths](#)

[Sevenths](#)

[Inverting intervals](#)

Gamoj

Davide Bonetti has contributed a large set of scale exercises and some pages describing all the scales. You can see the pages [here](#).

Intervals

In music theory we use the word interval when we talk about the pitch difference between two notes. We call them harmonic intervals if two tones sound simultaneously and melodic intervals if they sound successively.

Interval names consist of two parts. Some examples are "major third" and "perfect fifth". In Walter Pistons "Harmony" the two parts are called *the specific name* and *the general name* part. Wikipedia talk about *interval quality* and *interval number*. I have seen people talk about an intervals *numerical size*.

You find the general name by counting the steps on the staff, ignoring any accidentals. So if the interval you want to name goes from E to G#, then we count to 3 (E F G) and see that the general name is *third*.



The specific name say the exact size of the interval. Unisons, fourths, fifths and octaves can be diminished, pure or augmented. Seconds, thirds, sixths and sevenths can be minor, major, diminished or augmented. A minor interval is one semitone smaller than a major interval. A diminished interval is one semitone smaller than a pure or a minor interval, and a augmented interval is one semitone larger than a pure or major interval.

Accidentals change the size of intervals. The interval becomes one semitone larger if you add a sharp to the highest tone or a flat to the lowest tone. And it becomes one semitone smaller if you add a flat to the highest tone or a sharp to the lowest tone. In the following sections naming of the intervals will be shown in greater detail.

Seconds

Seconds are easy to recognise: the two notes are neighbours on the staff. One note is on a staff line, and the other one is in the space above or below. A minor second is one semitone step, also called a half step. A major second is two semitone steps, also called a whole step.

To learn to identify seconds, you first have to learn which seconds there are between the natural tones. As you can see in [Figuro 3.1, ""](#), only the intervals E-F and B-C are minor seconds. The rest are major intervals. You can check that [Figuro 3.1, ""](#) is correct by looking at a piano. You will see that there are no black keys between E and F and between B and C.

Figuro 3.1.



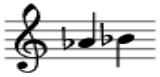
If the second has accidentals, then we have to examine them to find out how they change the size of the interval. Let us identify a few intervals!

Figuro 3.2.



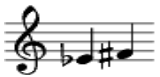
We remove the accidental from the interval in [Figure 3.2, “”](#) and see that the interval F-G is a major second. When we add the flat to the highest tone, the interval becomes one semitone smaller, and becomes a minor second.

Figure 3.3.



We remove the accidentals, and see that the interval A-B is a major second. You still do remember [Figure 3.1, “”](#), don't you? Then we add the flat to the A, and the interval become a augmented second. And when we add the flat to the B, and the interval becomes a major second.

Figure 3.4.



We remove the accidentals, and see that the interval E-F is a minor second. When we add a flat to the lowest tone, the interval becomes one semitone larger, and becomes a major second. And when we add a sharp to the highest tone, the interval becomes one semitone larger, and becomes an augmented second.

Thirds

A minor third is one minor and one major second, or three semitones. A major third are two major seconds, or four semitone steps. [Figure 3.5, “”](#) show the thirds between all the natural tones. You should memorise the major intervals, C-E, F-A and G-B. Then you know that the other four intervals are minor.

Figure 3.5.

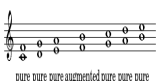


Then you examine the accidentals to see if they change the specific name. This is done exactly the same way as for seconds.

Fourth

A pure fourth is 2½ steps, or two major seconds and a minor second. [Figure 3.6, “”](#) show all fourths between natural tones. You should memorise that the fourth F-B is augmented, and that the other six are pure.

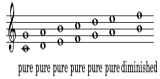
Figure 3.6.



Fifths

A pure fifth is $3\frac{1}{2}$ steps, or three major seconds and a minor second. [Figuro 3.7, “”](#) show all fifths between natural tones. You should remember that all those intervals are pure, except B-F that is diminished.

Figuro 3.7.



If the interval has accidentals, then we must examine them to see how they change the size of the interval. A diminished fifth is one semitone smaller than a pure interval, and a augmented fifth is one semitone larger. Below you will find a few examples:

Figuro 3.8.



We remember from [Figuro 3.7, “”](#) that the interval B-F is a diminished fifth. The lowest tone in [Figuro 3.8, “”](#) is preceded by a flat that makes the interval one semitone larger and changes the interval from a diminished to a pure fifth.

Figuro 3.9.



We know from [Figuro 3.7, “”](#) that interval E-B is a perfect fifth. In [Figuro 3.9, “”](#) the E has a flat in front of it, making the interval augmented. But then the B is preceded by a doble flat that makes the interval two semitone steps smaller and changes the interval to a diminished fifth.

Sixths

Sixths are easiest identified by [inverting the interval](#) and identifying the third. Then the following rule apply:

- If the third is diminished, then the sixth is augmented
- If the third is minor, then the sixth is major
- If the third is major, then the sixth is minor
- If the third is augmented, then the sixth is diminished

If you find inverting intervals difficult, then you can memorise that the intervals E-C, A-F and B-G are minor. The other four are major. Then you examine the accidentals to see if they change the specific name. This is done exactly the same way as for seconds.

Figuro 3.10.



Capítulo 4. Extending GNU Solfege

Enhavo

[Introduction](#)

[Lesson files](#)

[File encoding](#)

[Useful unicode characters](#)

[Comments](#)

[Types](#)

[Global variables](#)

[Lesson file contents](#)

[Header block](#)

[Question block](#)

[music objects](#)

[Functions](#)

[Operators](#)

[The chordvoicing module](#)

[The compareintervals module](#)

[The dictation module](#)

[The elembuilder module](#)

[The element block](#)

[The header block](#)

[The question block](#)

[The harmonicinterval module](#)

[The idbyname module](#)

[Question block](#)

[The identifybpm module](#)

[The idproperty module](#)

[The idtone module](#)

[The melodicinterval module](#)

[The nameinterval module](#)

[The rhythm module](#)

[The rhythmtapping module](#)

[The rhythmtapping2 module](#)

[The singanswer module](#)

[The singchord module](#)

[The singinterval module](#)

[The twelvetone module](#)

[The mpd module](#)

[Midi instrument names](#)

[Percussion instrument names](#)

Introduction

GNU Solfege is written so that it can easily be extended, even if you do not know any computer programming. The steps are:

- Create a lesson file and save it in the first directory listed when you select **User Exercises** from

the **File** menu. Create the directory if it does not exist.

- Select **User Exercises** once again to see the file show up in the list.
- Click the link to your lesson file and enjoy!

To get started, you can copy one of the lesson files included in GNU Solfege. The lesson files are located in the `exercises/standard/lesson-files` subdirectory of the installation directory. You can find the installation directory by selecting **File Locations** from the **Help** menu. It is important to store the lesson files you create in the directory intended for user created lesson files, and not in the applications directory. This to avoid loosing files when you upgrade the program.

If you create many lesson files, you might want to group them together in a separate subdirectory and attach them to a front page file. This way you have a set of files in a subdirectory that you can easily distribute to other students. So create a new directory side by side the `user` directory you found by selecting **User Exercises** earlier this article. Your files might be structured like this:

```
myfiles/myfrontpage.txt
myfiles/lesson-files/chords-1
myfiles/lesson-files/chords-2
```

To create a new front page file, you should select **Edit Front Page** from the **File** menu, and then click **New** on the toolbar of the dialog that pops up.

Lesson files

In GNU Solfege, each exercise is created by a lesson file interpreted by one of the exercise modules.

Deprecated modules: chord, harmonicprogressiondictation,

Missing documentation: chordvoicing, identifybpm, twelvetone

File encoding

Solfege by default expects the content of lesson files to be in UTF-8 encoding. Modern editors often let you specify the encoding in the "Save As" dialog. One example is gedit. Other programs, like vim and emacs let you specify the encoding inside the text file.

If this sounds complicated, you can safely ignore the whole encoding issue if you restrict yourself to use only standard ascii characters. That is only the letters a to z.

If you create lesson files with a different encoding, you have to declare the encoding in a special comment at the top of the file. This because Solfege and the tools used to translate Solfege cannot guess the encoding safely. We follow the same conventions as the Python language. See [PEP-0263](#) for the details.

What you have to do is add a comment to one of the first two lines of the lesson file, where part of the line matches `coding=encoding` or `coding: encoding`. Extra characters on the line are ignored, so if you use the emacs or vim editors, you can conveniently tell the editor about the file encoding. The following example sets the charset to ISO 8859-1, a charset commonly used in many west-european languages:

```
# -*- coding: iso-8859-1 -*-
```

Russians might want to use koi8-r:

```
# -*- coding: koi8-r -*-
```

Same as above, but in a format that works with the vim:

```
# vim: set fileencoding= koi8-r :
```

The program use the python libs to convert to unicode, so it should understand almost any encoding you can think of. If you see some characters are missing, for example when the name of questions are displayed on buttons, then most likely you have done something wrong with the encoding.

Useful unicode characters

Unicode has some characters that you might want to use to make labels look more professionally. If your editor use unicode by default, you may copy-and-paste the characters you need from here, if you are viewing this documentation in a web browser. The number is a hexadecimal number.

ø 00F8 LATIN SMALL LETTER O WITH STROKE

Half-diminished seventh chord.

° 00B0 DEGREE SIGN

Diminished seventh chord.

△ 25B3 WHITE UP-POINTING TRIANGLE, Δ 0394 GREEK CAPITAL LETTER DELTA

Major seventh chord. We do not know which character to recommend. Solfege does not care, so you can use the symbol you like.

♭ 266D MUSIC FLAT SIGN

This sign can be used instead of the letter 'b' for a flat sign.

♯ 266F MUSIC SHARP SIGN

This can be used instead of the letter '#' for the sharp sign.

Comments

Everything after # on a line is ignored. Example:

```
# This line is ignored. The next line is not.  
question { bla bla }
```

Types

Strings

Strings are quoted with the " character. Example:

```
"this is a string"
```

Use tripple quotes for strings that contain line breaks, or if the string itself has to contain the " character:

```
description = ""<h1>Long desription<h1> This lessonfile need  
very much descriptions. Qoutes (") are ok here. bla bla bla""
```

NB: All strings have to be unicode strings. If you get error messages like this one:

```
In line 21 of input: does not recognise this string ';' as a valid token.'  
(line 20): question {  
(line 21): question {  
(line 22):   name = _("Ionia")
```

then you must check the encoding of your file, and maybe you should read [la sekcio nomata "File encoding"](#). You can change the encoding of a file using the **iconv** program:

```
iconv -f YOUR_ENCODING -t utf8 your.file
```

Tempo

The tempo of music is entered as **bpm/beatlen**. The following example will set the tempo to 120 beats per minute, each beat being a quarter note.

```
tempo = 120/4
```

Global variables

Global variables can save you a few key strokes.

```
s = "\score\relative c'{ %s }  
question {  
# instead of music = music("\score\relative c'{ c d e f g2 g2 }")  
music = music(s % "c d e f g2 g")  
}
```

Lesson file contents

A lesson file consist of one header block and zero or more question blocks:

```
header {  
  ASSIGNMENT  
  ASSIGNMENT  
  ...  
}  
question {  
  ASSIGNMENT  
  ...  
}
```

Header block

The header block can be placed anywhere in the file, but by convention it should be the first block in the file. And there is a limitation that the header has to be within the first 40000 characters of the file.

Variables shared by many exercise modules

module

Tell what exercise module that will run the lesson file. This variable is required for all lesson files. (The variable was added in Solfege 2.9.0 where it replaced the `content` variable.). Example:

```
module = idbyname
```

replaces

A string or list of strings with hash values of lesson files that this lesson file can replace without dropping the statistics. Use this only when you know what you are doing. The hash value is calculated by `solfege.lessonfile.hash_of_lessonfile()`. `tools/hash-of-file.py` can be used to get the hash value of files before modifying them.

```
replaces = "bf7dd374206451bff43d61fc8191f5fb3e88d007"  
replaces = "bf7dd374206451bff43d61fc8191f5fb3e88d007",  
"cdb2f9415171650ee7682028788c1c42c62fdbf"
```

lesson_id

This variable is deprecated in Solfege 3.15.3. It should remain in existing lesson files for some time for backward compatibility. But it should not be added to new lesson files.

Each file need a unique identifier. The identifier can be any string you like, and if you don't add one, Solfege will add one for you. Solfege will also offer to create a new `lesson_id` if you have two files with identical `lesson_id`. Example:

```
lesson_id = "5b30c9ae-09f1-40b3-9333-4789638dc851"
```

version

Tell the version of solfege the lessonfile is known to work with. This variable is not required, but it should be used because it can (but don't guarantee to) help avoid trouble if the lesson file format changes in the future. Example:

```
version = "3.0.7"
```

title

Short one-line description that will be used for creating the menu entry for the exercise. You should add this to all lesson files. Example:

```
title = "Minor and major chords in root position"
```

lesson_heading

A short heading that will be displayed above the exercise. It should say what the purpose of the exercise is. Some modules provide a default value, others leave the string empty. Example:

```
lesson_heading = _("Identify the chord")
```

help

This variable say which help file from the user manual will be displayed when the user presses F1. Example:

```
help = "idbyname-intonation"
```

By default, Solfege will display the help file that has the same name as the exercise module being used in the lesson file.

theory

This variable says which help file from the user manual will be displayed when the user presses F3. Pressing F3 should display music theory about the exercise. Don't include this variable if there are no music theory written. Example:

```
theory = "scales/maj"
```

random_transpose

In some exercises the program can transpose the music to create variation. The default value is **yes**. (The default value changed from **no** to **yes** in Solfege 3.0.)

Used in modules: `chord`, `chordvoicing`, `harmonicprogressiondictation`, `idbyname`, `singanswer`, `singchord`

Possible values

`random_transpose = no`

No transposition will be done.

`random_transpose = yes`

The exercise will do random transposition. What kind of transposition depends on the exercise, but you get a ok result from this. This is the default value.

`random_transpose = accidentals, INTEGER1, INTEGER2`

Transpose the question by random and make sure the key signature of the question does not get more than a certain number of accidentals. In this context, the number of accidentals can be described by an integer value. A negative value denotes a number of flats (b), and a positive number denotes a number of sharps (#). Zero means no accidentals. The integers `INTEGER1` and `INTEGER2` define a range of allowed number of accidentals.

For this transposition mode to work properly, the music in the lessonfile has to be in the keys c major or a minor, or the question must have a `key` variable telling the key signature.

`random_transpose = key, INTEGER1, INTEGER2`

Transpose the music `INTEGER1` steps down or `INTEGER2` steps up the circle of fifth. In this context up is more sharps and down is more flats. This is real transposition where both the key and the notes are transposed.

`random_transpose = semitones, INTEGER1, INTEGER2`

Transpose the music at most `INTEGER1` semitones down or `INTEGER2` semitones up. This is real transposition where both the key and the notes are transposed. You will easily end up with music in the keys with LOTS of accidentals.

`enable_right_click = no`

By default, Solfege will let the user right-click on buttons to hear the music they represent without guessing. Set this variable to `no` for lesson files where it does not make sense, for example in a `idbyname` lesson file where many questions have the same name.

Modules: `idbyname`, `chordvoicing` and `chord`.

`disable_unused_intervals = no`

By default, Solfege will make the buttons insensitive for intervals that are not being asked. Set this variable to `no` if you want all buttons to be sensitive.

Modules: `harmonicinterval` and `melodicinterval`.

`ask_for_intervals_0`

Select which intervals to ask for. 1 for minor second, 2 for major second, 3 or minor third etc. Use a negative number for descending intervals. To ask for more than one interval create the variables `ask_for_intervals_1`, `ask_for_intervals_2` etc. In the following example Solfege will ask for two intervals. The first will be either a minor second or a major second, both intervals going up. And the second interval will be either major second or minor third, both intervals going down.

```
ask_for_intervals_0 = [1, 2]
ask_for_intervals_1 = [-2, -3]
```

Modules: `melodicinterval` and `singinterval`.

`intervals`

This variable tells which intervals should be asked for in exercises using the `harmonicinterval` module. 1 for minor second, 2 for major second, 3 or minor third etc. Example that will practise thirds:

```
intervals = [3, 4]
```

Modules: `harmonicinterval`.

`test`

This variable defines the test for the exercise. In a test, Solfege will ask all the questions in the lesson file a number of times. This variable is always used together with `test_requirement`. In the following example, each question will be asked 3 times:

```
test = "3x"
```

Modules: `harmonicinterval`, `idbyname`, `melodicinterval` and `singinterval`.

`test_requirement`

This variable defines how large percentage of the questions has to be answered correctly to pass the test. Example:

```
test_requirement = "90%"
```

Modules: `harmonicinterval`, `idbyname`, `melodicinterval` and `singinterval`.

`have_repeat_arpeggio_button = yes`

Set to **yes** if you want the exercise to have a "Repeat arpeggio" button.

Modules: `singanswer`.

`have_music_displayer = yes`

Set to **yes** if you want the question to have a music displayer.

In the `idbyname` module, setting this variable will add a music displayer where the program will display the answer when the user gives up or answers the question correctly. You might also want to read about [at_question_start](#).

In the `singanswer` module, setting this variable will add a music displayer where the music will be displayed when the question is displayed.

Modules: `idbyname`, `elembuilder` and `singanswer`.

`music_displayer_stafflines = INTEGER`

The number of empty staff lines to display when we have no music to display.

Modules: `idbyname` and `elembuilder`.

`at_question_start`

This variable changes what happens when the user clicks **New**. By default, Solfege will play the music when the user clicks **New**, and only display the music when the question is answered correctly and the `have_music_displayer` variable is set to **yes**. Setting this variable will also set `have_music_displayer` to **yes**.

`at_question_start = show`

The exercise will get a **Play music** button. When the user clicks **New** the music will be displayed in the music displayer, but no music is played. Click **Play music** to hear the music.

`at_question_start = play`

The exercise will get a **Display music** button. When the user clicks **New** the music is played. Click **Display music** to see the music.

`at_question_start = show, play`

When the user clicks **New** the music is both played and displayed.

Modules: `idbyname`, `elembuilder` and `rhythmtapping2`.

`vmusic`

This variable holds a representation of the question intended to be displayed. This can be necessary if the music is a `.wav` or `.mp3` file. It will be used when the user clicks Show music or when the question is answered correctly (if we have a musicdisplayer). Added to `idbyname` in Solfege 2.5.1 and to `elembuilder` in 3.9.2.

Modules: `idbyname` and `elembuilder`.

rhythm_elements

A list of integers (1-34) telling what elements we should use when creating questions. Example:

```
rhythm_elements = 0, 1, 2, 3, 4
```



Modules: `rhythm` and `rhythmtapping2`

Variables that has been obsoleted

`number_of_intervals = INTEGER`

Made obsolete in Solfege 3.1.5. Solfege will find this number automatically now, so this variable is ignored.

Question block

Variables you can define in the question block

`name`

Questions written for the [idbyname](#) or [elembuilder](#) exercise modules need a name. A name is optional for [dictation](#) module.

`music`

For most lesson files the music representing the question is assigned to this variable. Note that there is a shortcut. Instead of:

```
question {  
  name = "Lisa gikk til skolen"  
  music = music(...)  
}
```

you can write:

```
question {  
  name = "Lisa gikk til skolen"  
  music(...)  
}
```

```
}
```

Music objects are documented in [la sekcio nomata "music objects"](#).

tempo

Set the tempo for this questions music. The variable is defined "beats per minute" / "notelen per beat". Example:

```
tempo = 150 / 4
```

This variable can also be defined globally for the whole lesson file. Do do so you should put it in the beginning of the file, outside any question blocks.

Modules: `idbyname`, `chord`, `chordvoicing` and `rhythmtapping`.

instrument

By default, Solfege will use the instrument specified on the [preferences window](#) when playing questions. This variable let you select a different instrument. Example:

```
instrument = "cello", 100
```

The instrument name has to be quoted. The integer is the volume, and it should be in the range 0-127. You can see a list of instrument names in [la sekcio nomata "Midi instrument names"](#). For lesson files where it makes sense, it is possible to specify three set of instruments. The following example will play bass for the lowest tone, piano in the middle and clarinet on the top tone:

```
instrument = "bass", 100, "acoustic grand", 100, "clarinet", 100
```

This variable can also be defined globally for the whole lesson file. Do do so you should put it in the beginning of the file, outside any question blocks.

Modules: `idbyname`, `chord`, `singanswer` and `chordvoicing`

set

The `set` variable is used by some exercise modules to select which question to play when the user right clicks on one of the answer buttons. This can be useful if the lesson file has many questions with the same name, and you want solfege to play the question that is most closely related to the question being asked. You can assign whatever value you want. A good suggestion is to use integers.

In lesson files that does not use the `set` variable, solfege will play the first question it can find with the same name as the button the user right clicks on.

If the lesson file uses the `set`, or more precisely, if the question being asked has the variable defined, the program will first try to find a question where the `set` variable matches the question being asked, and the name matches the button clicked. If no match is found, the program will select a question to play as if the `set` variable was not used at all.

Modules: `idbyname` and `chordvoicing`.

music objects

Each question in your lesson files will define one or more `music` objects.

`music(musiccode)`

This is music entered completely following the music format FIXME spec. This means you have to enter complete code with a `\staff` command. Example:

```
variable = music("\staff\relative c' { c' d' }")
```

`music3(musiccode)`

The music object can be used for music that has 3 or more staves. It works the same way as [music](#), but if "Use different instruments for chords and harmonic intervals" is checked in the preferences window, the 3 instruments you can select the same place will be used instead of the preferred MIDI instrument.

`chord(musiccode)`

Enter the tones from the lowest to the highest tone, like this:

```
variable = chord("c' e' g'")
```

`satb(musiccode)`

This type of music is used by the singchord exercises. It let you say which tones of a chord the different voices in a choir will sing. Take this, for example:

```
variable = satb("c' | e' | g | c")
```

The `c'` will be sung by the soprano, `e'` by the alto, `g` by the tenor and `c` by the bass. Please notice that when this music is played in arpeggio, the tones to be sung by the women, will be played one octave deeper, of the user is a male. And vice versa if the user is a female or a child.

`voice(musiccode)`

This musictype saves some key strokes if you want to enter a melody.

```
variable = voice("c'4 c' g' g' | a' a' g'2")
```

is the same as

```
variable = music("\staff{ c'4 c' g' g' | a' a' g'2")
```

`rvoice(musiccode)`

`rvoice` is similar to `voice` except that the music is in `\relative` mode, relative to the first tone. The following two statements produce the same music:

```
variable = rvoice("c'4 c g' g | a a g2")  
\staff\relative c'{ c4 c g' g' | a a g2 }
```

`percussion(percussioncode)`

This music object provides a simple way to play rhythms with percussion instruments. Each tone

represents a percussion instrument as defined in [la sekcio nomata](#) “[Percussion instrument names](#)”. In the following example, the tone *c* is translated to the midi sound *Side Stick* and *d* to a *Mute triangle*.

```
variable = rhythm("d4 d d d c8 c8 c4")
```

rhythm(musiccode)

This music object let you write questions that taps rhythms with the two instruments defined in the preferences window. The tone *C* will play the rhythm representing the question, and the tone *d* can be used if you want to write some sort of "count-in" before the question starts. Example:

```
rhythm("d4 d d d c8 c8 c4 c c8 c8")
```

You should only use two pitches, *c* and *d*. Other pitches will print a warning, but will still work in the current implementation. To play real percussion with many different instruments you should use the [percussion](#) music object.

midifile(filename)

Play a midi file. The path given to the file is relative to the directory the lesson file is stored in. Example:

```
variable = midifile("share/example.mid")
```

wavfile(filename)

Play a *.wav* file. The path given to the file is relative to the directory the lesson file is stored in. Example:

```
variable = wavfile("share/fifth-small-220.00.wav")
```

mp3file(filename)

Play a MP3 file. Similar to *wavfile*.

oggfile(filename)

Play an Ogg Vorbis file. Similar to *wavfile*.

csound(orchestra, score)

Given a CSound orchestra and score, this music object will generate a WAV file and play it. Example:

```
csound(load("share/sinus.orc"), ""  
  f1 0 4096 10 1  
  i1 0 1 220.0  
  i1 + 1 329.04  
  """)
```

mma(mmacode), **mma**(groove, mmacode)

Create a music object that use [MMA](#) to generate music that it will play. If you create the object with one argument, *mmacode* should be a string with complete MMA code. With two arguments, *groove* is a string with the name of the groove, and *mmacode* is comple MMA code, except it

could be missing the initial "Groove" instruction. The groove from `groove` will be prepended the string.

cmdline(*shell code*)

Run an external program. Example:

```
cmdline("./bin/csound-play-harmonic-interval.sh 220.000000 320.100000")
```

Functions

● **_**(*message*)

Return the translation of *message* if it exist. Return the string unchanged if not.

```
title = _("Bla bla title")
```

● **include**(*filename*)

Read the file *filename* into the lesson file and parse it as a part of the file. The filename is relative to the location of the lesson file.

```
include("singchord-1")
```

The lesson header variables will be taken from the including lesson file. Only if a variable is only defined in the included lesson file, and not in the including lesson file, then the value will be taken from the included file.

● **load**(*filename*)

Read the file *filename* from disk and return it as a string. The filename is relative to the location of the lesson file.

```
orc = load("share/sinus.orc")
```

Label functions

We call these functions *label functions* because we use them to create the label for some questions in the program. You should only use these functions where they are documented to work.

● **pangomarkup**(*pangostring*)

Return a label that the program can put on a button. The label is created using GTK pangomarkup. [Google for "pango markup"](#) to get the markup explained. Notice that you have to use triple quotes around the string.

```
pangomarkup("""<span size="xx-large">V</span>""")
```

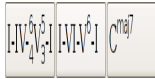
● **progressionlabel**(*str*)

This function has existed in Solfege for a while, but it has not been documented until now. Should we find a shorter function name? An alias can be added so that the old long function name still works.

Return a label. *str* is interpreted like this:

- Each letter outside of a parentheses is displayed with a large serif font.
- The text inside parentheses is displayed as a superscript: smaller letters above the baseline.
- If the text inside the parentheses is divided by a comma, the text before the comma is superscript and after the comma is subscript.

```
progressionlabel("I-IV-(6,4)V(5,3)-I")
progressionlabel("I-VI-V(6)-I")
progressionlabel("C(maj7)")
```



● **rnc(str)**

Display a sequence of roman numeral chords. The chords are separated by whitespace and an optional hyphen. The exact implementation of this is still open for discussion. The current developent version of Solfege will divide each chord in 3 parts and give them different font sizes, and also try to make the chord compact, so that it should not take too much space on screen.

- The first part of the chord is the roman numeral, including an optional **b** or **♭** (unicode character U+266D MUSIC FLAT SIGN).
- The second part is the letters (if any) between the first and the third part.
- The third part is from the first digit and the rest of the chord.

```
rnc("Imaj7-IIIm7-V9-Imaj7")
```



Spaces are not allowed in the chord name.

New in version 3.11.0.

● **chordname(str)**

Display a sequence of chords. The chords are separated by whitespace. Each chord consist of up to four parts, and part two to four are optional:

```
[notename] [txt1] [:txt2] [/bass]
```

notename and **bass** music be a notename in the format understood by the music parser. You can read more about this in [la sekcio nomata "The mpd module"](#). Example:

```
g:11b9 cm/g ges:Δ besm:7/f
```



New in version 3.11.1.

Operators

Operators can only be used on strings. + is used for joining strings, and % is similar to what you find in python, but it is very limited. It only know about %S. One example:

```
"\staff\relative c'{%s}" % "c d e"
```

evaluates to

```
\staff\relative c'{c d e}
```

The chordvoicing module

Still undocumented...

The compareintervals module

Here is a minimal lesson file:

```
header {  
  countin\_perc = compareintervals  
  title = "Compare intervals"  
  lesson\_id = "9f830e12-1f50-4fa9-8688-1e04469692fa"  
}
```

This file will make an exercise that ask you to compare harmonic intervals. And since you do not say which intervals, it will ask for all intervals from a small second up to a major tenth.

`first_interval_type`, `second_interval_type`

Let you select if the intervals you are asked to compare should be a melodic or a harmonic interval. The default value is `melodic`. Possible values: `harmonic` and `melodic`.

```
first_interval_type = melodic  
second_interval_type = harmonic
```

Modules: `compareintervals`.

`first_interval`, `last_interval`

Select which intervals to select from when creating the questions. This variable should be defined the same way as [ask_for_intervals_0](#). If these two variables are not defined, then the user will be able to select which intervals to practise from the Config page of the exercise.

Modules: `compareintervals`.

The dictation module

Example:

```
header {  
  module = dictation
```

```

lesson_id = "a265df62-e007-4a1b-9057-cd05397e88a2"
title = _("Norwegian children songs")
version = "2.1.10"
}

question {
name = "Bæ, bæ, lille lam"
tempo = 130/4
breakpoints = 2/1, 4/1, 8/1, 10/1, 12/1, 14/1
music = rvoice("""
\time 4/4
c'2 g' | e4 e c2 | d4 d g, g | c1 |
c2 g' | e4 e c2 | d4 d g, g | c1 |
a'4 f f f | g2. e4 | f d d d | e2. c4 |
a'2 f | g e4 e | f b, b b | c1 |
""")
}

question {
# this tempo definition overrides the global
tempo = 160/4
name = "Lisa gikk til skolen"
breakpoints = 2/1, 4/1, 6/1
music = rvoice("""
\time 4/4
c' d e f | g2 g2 | a4 a a a | g1 |
f4 f f f | e2 e | d4 d d d | c1
""")
}

question {
name = "Det satt to katter på et bord..."
tempo = 96/4
music = rvoice("""
\key g \major \time 2/4
d'8 | [g g] [fis e] | [fis g] a4 | [d,16 d d d] [e8 fis] | g2 """)
}

```

By default, the dictation exercise will show the first column of music, and then the user should write the rest. But if the first column is not good enough, for example if there are only rests on the first beat, these two variables can tell the program how much music to display:

clue_end

The following example will display the music on all staves in the first quarter note:

```
clue_end=1/4
```

clue_music

This is an alternative to `clue_end`. The music assigned to `clue_music` will be shown to the user when he should start the dictation. You should not use both `clue_end` and `clue_music` in the same question.

breakpoints

Set breakpoints in the music, so you can hear the music in parts when doing the dictation.

The `elembuilder` module

Here is a minimal lesson file:

```

element progI { label = "I" }
element progIV { label = "IV" }
element progV { label = "V" }

header {
  lesson\_id = "3f3872c0-ef2e-4132-9fb1-97f75c7b28fd"
  module = elembuilder
  title = "progression test"
  elements = auto
  # uncomment if you want a music displayer.
  # have_music_displayer = yes
}

question {
  music = rvoice("<c' e g> <b d g> <c e g>")
  elements = progI, progV, progI
  name = "I-V-I"
}
question {
  music = rvoice("<c' e g> <c f a> <c e g>")
  elements = progI, progIV, progI
  name = "I-IV-I"
}

```

The element block

This block defines the elements the user can put together to answer the question. Each block is named by the string between `element` and `{`. The block defines one variable, `label` that is the label the button will get.

`label` can either be a plain string or one of the [label functions](#).

The header block

`elements`

This variable defines which elements to display. Set this to `auto` to display all elements that are needed to answer the questions in the lesson file. You can display more elements that needed to make it more difficult for the user. An example:

```
elements = progI, progIV, progV, progIV, progV_6
```

`music_displayer_stafflines`

Set this if you want the music displayer to show more than one empty staff line when the music displayer have no music to display.

See also [at_question_start](#) and [music_displayer_stafflines](#).

The question block

`elements`

This variable defines which elements defines the question. It can be elements, as defined in the example above, or strings or labels defined by the [label functions](#).

`tonic`

The exercise will have a "Play tonic" button if this variable is defined in a question in the lesson file. The variable should contain some music to play to the user so that he knows the tonic of the question. This can be useful in harmonic progressions that does not start on the tonic. This variable is optional. Example:

```
tonic = chord("c e g")
```

name

The name is needed for storing statistics. A string or a label created by the [label functions](#).

See also [vmusic](#).

The `harmonicinterval` module

User documentation is in [la sekcio nomata "Harmona intervalo"](#).

Here is a minimal lesson file:

```
header {
  module = harmonicinterval
  lesson\_id = "a400df62-e007-4a1b-9057-cd05397e88a2"
  version = "3.1.4"
  title = "Seconds"
  intervals = [1, 2]
  test = "3x"
  test\_requirement = "90%"
}
```

Additional variables you can put in the header. Click on the link to get an explanation:

- [disable_unused_intervals](#)
- [lesson_heading](#)

The `idbyname` module

This is a very generic exercise. In its most basic form, the program will play some sound, and you have to select among several buttons that in some way represents the music.

Here is a minimal lesson file:

```
header {
  module = idbyname
  lesson\_id = "a400df62-e007-4a1b-9057-cd05397e88a2"
  version = "3.1.4"
  title = "Menuitem title"
}
question {
  name = "Major"
  music = chord("c' e' g'")
}
question {
  name = "Minor"
  music = chord("c' es' g'")
}
```

Optional idbyname header variables

`filldir = vertic`

Tell the direction the buttons are filled. Default value is `horiz`.

Modules: `idbyname`.

`fillnum`

Tell how many buttons there are in each row or column. The default value is 1.

Modules: `idbyname`.

`labelformat = progression`

The default value is `normal`. Set to `progression` for lesson files where the name of the questions is a harmonic progression, written in a undocumented, but not difficult format. Check some existing lesson file to see how it works.

Averno

Using this variable is deprecated. Do not use it for new lesson files.

Modules: `idbyname`

`have_repeat_slowly_button = yes`

Set to `yes` if you want the exercise to have a "Repeat slowly" button.

Modules: `idbyname`.

See also [at_question_start](#) and [music_displayer_stafflines](#).

Question block

Required question variables

- [name](#). Can be a string or a label created by the [label functions](#).
- [music](#)

Optional question variables

`vmusic`

See [vmusic](#).

`cuemusic`

Will be displayed in the music displayer when the user clicks New. Ignored if `at_question_start = play`, `show` or `at_question_start = show`, because then the content of `music` or `vmusic` is displayed when the user clicks New. (Added in Solfege 2.5.1)

The `identifybpm` module

Still undocumented...

The `idproperty` module

The `idproperty` module let you create exercises where solfège will play some music and you have to identify different properties of the music.

Below is a minimal lesson file. It will create an exercise that will play a minor or major chord and the user answers with two buttons labeled "Minor" and "Major" and two buttons representing the inversion. Notice that unused properties, `toptone` in this example, are hidden.

```
header {
  module = idproperty
  flavour = "chord"
  title = "Minor and major chords"
  lesson_id = "e263d40a-d8ff-4000-a7f2-c02ba087bf72"
}
question {
  name = "Major"
  music = chord("c' e' g'")
  inversion = 0
}
question {
  name = "Minor"
  music = chord("es' g' c'")
  inversion = 1
}
```

`flavour = "chord"` will add the following definitions to the lesson file header, unless if they are missing:

```
new_button_label = _("New chord")
lesson_heading = _("Identify the chord")
qprops = "name", "inversion", "toptone"
qprop_labels = _("Name"), _("Inversion"), _("Toptone")
```

`new_button_label` is the label to put on the **New** button. The default value is `_("New")`.

`lesson_heading` will set the heading to be displayed when you practise. The default value is an empty string, that will hide the heading.

The properties are defined by the `props` variable in the lesson file header, and there should be a variable `prop_labels` that defines the label to use. `props` and `prop_labels` must be lists of equal length.

The exercise will have a **Repeat arpeggio** button if one or more of the questions can be played arpeggiated. Set the lesson file header variable `have_repeat_arpeggio_button` to `no` to disable hide the button.

If the exercise have a `inversion` property, it will be treated special. If assigned integer values, like in the example, the integer values will be replaced with strings. So `0` is replaced with "root position", `1` with "1. inversion" etc.

The `idtone` module

Here is a minimal lesson file:

```
header {
  module = idtone
  title = "Id tone 3"
  lesson_id = "e263d70a-d8ff-4000-a7f2-c02ba087bf72"
  black_keys_weight = 0, 0, 0, 0, 0
  white_keys_weight = 1, 1, 1, 0, 0, 0, 0
}
```

The 'weight' of a tone tell how big chance is it that the program will select this tone as the next to identify. Think of the weight of a tone as the number of lottery tickets with the name of the tone.

The variable `black_keys_weight` set the weight of the tones `c#`, `d#`, `f#`, `g#` and `a#`, and `white_keys_weight` will set the weight of the tones `c`, `d`, `e`, `f`, `g`, `a`, `b`. In the example above, the tones `c`, `d` and `e` get an equal weight of 1, the other tones 0. This mean that the only tones that will be asked for are `c`, `d` and `e`, and that the three tones share the same probability to be selected.

The `melodicinterval` module

User documentation is in [la sekcio nomata “Melodia intervalo”](#).

Here is a minimal lesson file:

```
header {
  module = melodicinterval
  lesson_id = "a400df62-e007-4a1b-9057-cd05397e88a2"
  version = "3.1.4"
  title = "Seconds and thirds"
  ask_for_intervals_0 = [1, 2, 3, 4, -1, -2, -3, -4]
  test = "3x"
  test_requirement = "90%"
}
```

Additional variables you can put in the header. Click on the link to get an explanation:

- [disable_unused_intervals](#)
- [lesson_heading](#)

Tests are only partially implemented for the `melodicinterval` exercise module: tests where each question is made by more than one interval does not work yet.

The `nameinterval` module

Here is a minimal lesson file:

```
header {
  lesson_id = "5623c43e-f529-4376-a0c9-c7d533050360"
  module = nameinterval
  title = _("Fifths")
  intervals = p5, a5, d5
}
```

intervals

A list the the intervals to ask for. The intervals are written in a short form, a letter and a number, like `d5` or `m7`. The letters are telling the interval quality are 'd' for diminished, 'a' for augmented, 'm' for minor, 'M' for major and 'p' for perfect.

tones

This variable sets the range of tones that can be used when constructing the intervals. The note names as to be quoted. The default value is `"b", "g"`. Example:

```
tones = "c'", "f'" # valid
tones = c', f'     # not valid
```

accidentals

This variable defines how many accidentals the tones making the interval can have. The value 0 means no accidentals, 1 means that flats and sharps are allowed, and 2 means that double flats and double sharps are allowed. The default value is 1. Example:

```
accidentals = 2
```

clef

Set which clef to use. The default value is `violin`. Possible values: `violin`, `treble`, `subbass`, `bass`, `baritone`, `varbaritone`, `tenor`, `alto`, `mezzosoprano` and `french`. Example:

```
clef = bass
```

The rhythm module

A simple rhythm exercise. Solfege will randomly generate rhythm patterns that the user should recreate by clicking on buttons.

Here is a minimal lesson file:

```
header {
  module = rhythm
  lesson_id = "7a4910be-de17-4ce3-9d15-78d48ccf945e"
  version = "3.1.4"
  title = "Easy rhythms"
  rhythm_elements = 1, 2, 3, 4
}
```

visible_rhythm_elements

Define this variable if you want more rhythm elements that the one to be asked for. This variable must include both the rhythm elements defined in `rhythm_elements` and the extra elements. Example:

```
rhythm_elements = 0, 1, 2, 3, 4, 5, 6
```

countin_perc

An integer value between 35 and 81, representing the percussion instrument used to give you the

beat before the question. The default value is 80. Example:

```
countin_perc = 35
```

```
35 Acoustic Bass Drum 51 Ride Cymbal 1 67 High Agoga
36 Bass Drum          52 Chinece Cymbal 68 Agogo Low
37 Side Stick         53 Ride Bell      69 Cabasa
38 Acoustic Snare     54 Tambourine     70 Maracas
39 Hand Clap          55 Splash Cymbal 71 Short Whistle
40 Electric Snare     56 Cowbell        72 Long Whistle
41 Low Floor Tom      57 Crash cymbal 2 73 Short Guiro
42 Closed Hi Hat      58 Vibraslap      74 Long Guiro
43 High Floor Tom     59 Ride Cymbal 2 75 Claves
44 Pedal Hi Hat       60 Hi Bongo       76 Hi Wood Block
45 Low Tom            61 Low Bongo      77 Low Wood Block
46 Open HiHat         62 Mute Hi Conga 78 Mute Cuica
47 Low-Mid Tom        63 Open High Conga 79 Open Cuica
48 Hi-Mid Tom         64 Low Conga      80 Mute Triangle
49 Crash Cymbal 1     65 High Timbale  81 Open Triangle
50 High Tom           66 Low Timbale
```

Modules: rhythm

rhythm_perc

Same as [countin_perc](#), but setting the instrument used to play the question. The default value is 37.

Modules: rhythm

count_in

The number of beats as count in. The default value is 2.

Modules: rhythm

bpm

The tempo, in beats per minute. The default value is 60.

Modules: rhythm

num_beats

The number of elements the question is made of. The default value is 4.

Modules: rhythm

The rhythmtapping module

Exercises using this module will play some music and then the user should tap the rhythm. The program will then say if the users rhythm is similar enough to the rhythm played by the computer.

Here is a minimal lesson file:

```
header {
  module = rhythmtapping
  lesson_id = "82b718e8-f174-446f-8297-58ddd17dae03"
  version = "3.7.0"
```

```

  title = "Rhythm tapping test"
}
question {
  music = rhythm("c4 c8 c8")
}
question {
  music = music("\staff\relative c'{c4 d8 e f4}\addvoice\relative c'{c4 b8 c a4}")
  rhythm = rhythm("c4 c8 c c4")
}

```

The first question in the example is very simple and self explaining. Solfege will play the rhythm defined in the `music` variable, and the user should tap that rhythm.

The second question is a little more complicated. Here Solfege will play the music defined in the `music` variable. And when the user taps the rhythm, Solfege will compare the users rhythm with the rhythm defined in the `rhythm` variable. The reason for using two variables is that Solfege is not smart enough to figure out the rhythm if you enter polyphonic music. It make noe difference if you set the `rhythm` variable to be a `rhythm` music object, or another single voice type like `rvoice`. This might change in the future. You as a lesson file author must make sure the rhythms in the two variables are in fact the same.

The `rhythmtapping2` module

Solfege will play a generated rhythm, and the user should tap the same rhythm.

Here is a minimal lesson file:

```

header {
  module = rhythmtapping2
  lesson\_id = "7a4916be-de47-42e3-9d15-78d48ccf945e"
  version = "3.7.0"
  title = "Rhythm tapping test"
  rhythm\_elements = 1, 2, 3, 4
}

```

See also [at_question_start](#).

The `singanswer` module

Here is a minimal lesson file:

```

header {
  module = singanswer
  lesson\_id = "a400df62-e007-4a1b-9057-cd05397e88a2"
  version = "3.1.4"
  title = "Sing the root of the chord"
}
question {
  question_text = "Sing the root"
  music = chord("c' e' g'")
  answer = chord("c'")
}
question {
  question_text = "Sing the root"
  music = chord("a' c' e'")
  answer = chord("a'")
}

```

Additional variables you can put in the header. Click on the link to get an explanation:

- [have_repeat_arpeggio_button](#)

The **singchord** module

Questions for this exercise need to have the **key** variable set if the key signature is anything else than "c" major (or "a" minor). Example:

```
header {
  module = singchord
  lesson_id = "a404df62-e037-6a1b-9027-cd05397e88a2"
  version = "3.1.4"
  title = "Simple chords"
}
question { music = satb("c'|e'|g|c") }
question { music = satb("a'|e'|c'|a") }
question { key="d \major" music = satb("a'|fis'|d'|d") }
question { key="f \minor" music = satb("as'|f'|c'|f") }
```

See also [la sekcio nomata "Kanti akordon"](#).

The **singinterval** module

This is an exercise where the program display an interval and play the first tone. Then the user should sing the interval, and then click a button to hear the correct answer. There is no microphone support yet.

User documentation is in [la sekcio nomata "Kanti intervalon"](#).

Here is a minimal lesson file:

```
header {
  module = singinterval
  lesson_id = "a400df62-e007-4a1b-9057-cd05397e88a2"
  version = "3.1.4"
  title = "Thirds"
  ask_for_intervals_0 = [3, 4]
  test = "3x"
  test_requirement = "90%"
}
```

The **twelvetone** module

Still undocumented...

The **mpd** module

The module is not documented yet. The input format is similar to the one used by GNU Lilypond, but only the simplest construct works.

Quick note: Notenames understood by the program are c, d, e, f, g, a, b, with 'is', 'isis', 'es', or 'eses' added. For example 'fis', 'bes', 'gisis'.

Midi instrument names

acoustic grand	contrabass	lead 7 (fifths)
bright acoustic	tremolo strings	lead 8 (bass+lead)
electric grand	pizzicato strings	pad 1 (new age)
honky-tonk	orchestral strings	pad 2 (warm)
electric piano 1	timpani	pad 3 (polysynth)
electric piano 2	string ensemble 1	pad 4 (choir)
harpsichord	string ensemble 2	pad 5 (bowed)
clav	synthstrings 1	pad 6 (metallic)
celesta	synthstrings 2	pad 7 (halo)
glockenspiel	choir aahs	pad 8 (sweep)
music box	voice oohs	fx 1 (rain)
vibraphone	synth voice	fx 2 (soundtrack)
marimba	orchestra hit	fx 3 (crystal)
xylophone	trumpet	fx 4 (atmosphere)
tubular bells	trombone	fx 5 (brightness)
dulcimer	tuba	fx 6 (goblins)
drawbar organ	muted trumpet	fx 7 (echoes)
percussive organ	french horn	fx 8 (sci-fi)
rock organ	brass section	sitar
church organ	synthbrass 1	banjo
reed organ	synthbrass 2	shamisen
accordion	soprano sax	koto
harmonica	alto sax	kalimba
concertina	tenor sax	bagpipe
acoustic guitar (nylon)	baritone sax	fiddle
acoustic guitar (steel)	oboe	shanai
electric guitar (jazz)	english horn	tinkle bell
electric guitar (clean)	bassoon	agogo
electric guitar (muted)	clarinet	steel drums
overdriven guitar	piccolo	woodblock
distorted guitar	flute	taiko drum
guitar harmonics	recorder	melodic tom
acoustic bass	pan flute	synth drum
electric bass (finger)	blown bottle	reverse cymbal
electric bass (pick)	skakuhachi	guitar fret noise
fretless bass	whistle	breath noise
slap bass 1	ocarina	seashore
slap bass 2	lead 1 (square)	bird tweet
synth bass 1	lead 2 (sawtooth)	telephone ring
synth bass 2	lead 3 (calliope)	helicopter
violin	lead 4 (chiff)	applause
viola	lead 5 (charang)	gunshot
cello	lead 6 (voice)	

Percussion instrument names

The first column is the integer value for the instrument. The second column tell the name of the note you should enter in the [rhythm](#) music object.

35 b,,	Acoustic Bass Drum	59 b	Ride Cymbal 2
36 c,	Bass Drum 1	60 c'	Hi Bongo
37 cis,	Side Stick	61 cis'	Low Bongo
38 d,	Acoustic Snare	62 d'	Mute Hi Conga
39 dis,	Hand Clap	63 dis'	Open High Conga
40 e,	Electric Snare	64 e'	Low Conga
41 f,	Low Floor Tom	65 f'	High Timbale
42 fis,	Closed Hi Hat	66 fis'	Low Timbale
43 g,	High Floor Tom	67 g'	High Agogo
44 gis,	Pedal Hi Hat	68 gis'	Agogo Low
45 a,	Low Tom	69 a'	Cabasa
46 ais,	Open HiHat	70 ais'	Maracas

47 b,	Low-Mid Tom	71 b'	Short Whistle
48 c	Hi-Mid Tom	72 c''	Long Whistle
49 cis	Crash Cymbal 1	73 cis''	Short Guiro
50 d	High Tom	74 d''	Long Guiro
51 dis	Ride Cymbal 1	75 dis''	Claves
52 e	Chinese Cymbal	76 e''	Hi Wood Block
53 f	Ride Bell	77 f''	Low Wood Block
54 fis	Tambourine	78 fis''	Mute Cuica
55 g	Splash Cymbal	79 g''	Open Cuica
56 gis	Cowbell	80 gis''	Mute Triangle
57 a	Crash Cymbal 2	81 a''	Open Triangle
58 ais	Vibraslap		

Apendico A. GNU General Public License version 3

Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program—to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of

products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

“This License” refers to version 3 of the GNU General Public License.

“Copyright” also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

“The Program” refers to any copyrightable work licensed under this License. Each licensee is addressed as “you”. “Licensees” and “recipients” may be individuals or organizations.

To “modify” a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version” of the earlier work or a work “based on” the earlier work.

A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate” a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey” a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The “source code” for a work means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is

widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component”, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work’s System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users’ Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work’s users, your or third parties’ legal rights to forbid circumvention of

technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a. The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b. The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".
- c. You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d. If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a. Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b. Convey the object code in, or embodied in, a physical product (including a physical distribution

medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

- c. Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d. Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e. Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either (1) a “consumer product”, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

“Installation Information” for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

“Additional permissions” are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a. Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b. Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c. Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d. Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e. Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f. Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered “further restrictions” within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party’s predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on

which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely

from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License “or any later version” applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy’s public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE

POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

one line to give the program's name and a brief idea of what it does.
Copyright (C) year name of author

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

program Copyright (C) year name of author
This program comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type 'show c' for details.

The hypothetical commands ‘show w’ and ‘show c’ should show the appropriate parts of the General Public License. Of course, your program’s commands might be different; for a GUI interface, you would use an “about box”.

You should also get your employer (if you work as a programmer) or school, if any, to sign a “copyright disclaimer” for the program, if necessary. For more information on this, and how to apply and follow the

GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.