Roadmap including past and future work; '+' = done; IP= in process; NOT= things that don't need doing anymore.
ctrl-shift-p to find commands

Bugs to squash

In Process

Features to add

- what and why pages in various places?
- distinguish C, N, and NC
- XLP export

Version 0.3 Nov 2020

- [x] finish logic of main window, make things depend on what they should (e.g. subcheck should depend on check.type (be either C, V, or CV –for now))
- [x] make regexes work, to include profile (of those available for ps), check (e.g., `V1=V2`), and subcheck (e.g., `a:CaC\2`).
- [x] Add padding to buttons and word labels
- [x] background color for windows: light blue or light green #b3ff99 #c6ffb3
- [x] enter main page with exit button inside title.
- [x] extend comparatives buttons, make window large enough for them all, or wordwrap
- [x] make script not die when ps doesn't appear in frames (i.e., be OK making a first frame)
- [x] make get lift work.
- [x] get check should do something (other than a blank frame!) when there are no check options for a ps/profile combo.
- [x] Make logic based on senses, not entries. This will lead to problems with people who put fine grained sense dinstinctions in their dictionaries, but hopefully that isn't for younger dictionaries, at least. But it will allow us to distinguish between zero derivations of a root, so verb frames will only work

with verb senses, even if the lexeme has a verb and noun sense.
And critically, it will facilitate putting this info in examples.

- [ ] remove all guid references from logic, unless needed for some reason (think about this!)
- [x] impliment splash screen
- [x] add icon to splash screen
- [x] write function to ask if there are unsorted senses for a particular check frame
- [x] if so, call sortT.
- [ ] OK w/o this:when calling a (tone?) report, first call function to check for entries to sort.
- [x] make entries (not frame) disappear when they are selected on verifyT.
- [x] make comparative words smaller font (between instructions and sorting word)
- [x] to sorting frames, add status to upper right corner (small), with sense x/y progress.
- [x] to verifying frames, add status to upper right corner (small), with group x/y progress.
- [ ] remove references to GUID, unless needed?
- [ ] Set up data collection:
- [ ] ask for word (with glosses prompting)
- [ ] ask for plural (after all of above, or each?)
  - [ ] if yes, add a ps @gi = "Noun", add form to examples
- [ ] ask for imperative (after all of above, or each, and even if plural given?)
- [ ] if yes, add a ps @gi = "Verb", add form to examples (this could result in second sense; do we want that?)

Hide config page details, make simpler interface.
add focus for confirmation buttons on add morpheme windows

Tone Frames:

- [ ] Fix this dependency!
  File "/home/kentr/bin/raspy/doc/dictionarychecker/main_lift.py", line 552, in framelocationsbyps
  l += [self.toneframes[ps][f]['location']]
  KeyError: 'location'
- [ ] make distictions for frame values and glossings by language, not 'form', gloss, etc.
- [ ] update framedentry to use iso values, too.

- [ ] check other functions that call framedentry, adjust them (remove framed['form'], make framed[self.analang])

addmorpheme:

- [x] assume ps (or
  take new ps from defaults file
  later: write window to put it there.
  this is only needed until that ps appears in the lexicon.
  Once done, can add frames per normal)
- [x] enter form
- [x] enter gloss
- [x] generate guid, entry @id, sense @id, datecreated/modified (and generalize use of these last two?)
- [ ] pull definitions, glosses, and semantic domain info from LIFT template.

+ sound.py:
+ make RecordButton class
+ _on_press: open to record
+ _on _release: stop recording, save as 96khz and 8khz, both, in two separate folders (make adjustable?)
+ filename: ps-profile-senseid-(form?)
+ make callback pyaudio work
+ put link to file in LIFT.
+ remove audio/ from links
+ make window to set and test sound card capacities
+ save defaults for recording

Examples to record window:
+ take example
+ senseids from defaults file,
+ later (or sooner?): make window to add these (based on tone grouping).
+ show example fields for that window
+ add record buttons

XLP reports: (low priority; get data into lift first)
use attributes variable like for lift
make get and put/contstruct functions
p
table/row/cell
section
gloss @glosslang
langdata @analang
recording href link nodes
tonegroup sections

For Readme: This to be run before alphabet chart, and before dictionary printing.
Make status/progress board based on progress dictionary object
be able to set defaults and test order variables:

one group can do verbs, one group can do nouns.
one group can do these letters/ another does another.
hardset these so they don't get unset except manually.
mark checks as they are done, allow automatic call on next step
clear entries that have had words added to them in a previous task (to be redone)
I need to exclude data already found when looking for syllable types… or exclude CC's by default…
make last select/change button also call runcheck
when sorting a segment type, have comparative columns,
set for one, two, or three (default =2?)
user scans column, clicks on buttons for entries that don't belong.
"don't belong" triggers "where"? dialog:
data is changed
check for the pile to which it goes is set as undone
check results frame is reloaded
make change move between columns (a v e, for instance, so each results pane resets)
click only on what is not right, one verify all button on the page).
If needed to open up to all vowels, don't give them in isolation, but show what the word would look like for each vowel.
Make first UI question: "Which letter do you want to verify?"
second UI Q: "by itself, or with a consonant/vowel?"
Then automate CV type and subcheck, based on number of relevant entries.
leave a button to go back to leaderboard to pick a different check.
+column align output and left justify
+remove brackets from spaced data
+remove info at the top of status (made a function for later)
+remove lexeme form
+remove language option if there is only one
instructions: "click on what doesn't belong.""
columns per choice, labeled and scrolling.
add picture on cards
?add larger font on word form
make button labels for languages, etc human readable (language.ldml?)
wrt default settings:
+imply gloss if only one
+imply gloss2 from gloss1 selection and presence of just one other gloss in database.
be able to change these things in the progress/leader board, if needed.
+remove "None of These"
make a way to mark duplicates(!)
Window NOT working: fix two vowels the same correction.
Think through how to feed orthography decisions:
add a new letter/split segment into two segments
add to database
add to program (extra segments setting?)
change a symbol (across database?)
update to PEP8

break lines at 79 characters (done to 850, where symbolic coding breaks down)
make more user friendly
+ add exit to main window
+ make newform work correctly; logic is currently bad.
IP: fix windows to the changes to entry,newform can be written.
Read and modify LIFT format in a predictable and non-destructive way
-go back and think through accessing lift file properly.
-most people won't have a computer structure like I do.
-select a lift file, determine lift-ness and ISO-639 code from there.
+ pull data from xml database to populate cards faithfully.
+ lift_get.py has the following functions:
+ tree and parse to read and write each of analyzable xml and whole tree.
+ functions to put out all of a given field indiscriminately (guids,forms, tuple of IDs/forms, gloss, gloss2, ps)
+ functions to count the number of a given field (entries)
+ functions to output guid for a given form
+ functions to output field for a given guid (illustration,gloss, gloss2, field types, number of fields, content of given field type)
+ functions to output tuple of ids/ps/forms matching a regex on forms.
+ functions to return a list of possible consonants, and a regex for consonants (and each for vowels)
+ function to remove duplicates in a list
+ function to join list to a single string
+ function to test if a particular glyph exists in any existing forms
+ function to iterate through glyphs to reduce the list to those in existing forms
+ functions to include portions of mono- di- and tri- graphs in more complex Consonant function
take response from UI, write changes to xml faithfully.
We might need a mechanism to interpret user response so correct (kind of) change is made.
We will know this is done when we see tests showing all and only the changes we're looking for, to the fields we're looking at, leaving everything else alone.
We should set which kinds of changes we are looking to make, and specifically test a list of them, as we implement features.
+ Generalize operation by use of regular expressions for filtering data, as well as for reading and writing, e.g.,

```
^([mn]{0,1})([[ptjfvmlryh]|[bdgkcsznw][hpby]{0,1}])([aiɨuʉɛɔɔʌ])([̌]{0,1})([mn]{0,1})([[ptjfvmlryh
^([mn]{0,1})([[ptjfvmlryh]|[bdgkcsznw][hpby]{0,1}])([aiɨuʉɛɔɔʌ]{1,2})([̌]{0,1})\1\2([aiɨuʉɛɔɔʌ]{1
```

+ Cross-platform - by developing in python.
Use Object oriented programming
+ test functionality of vbutton function
+ add guid
+ add fake v for x = v to call.
+ see which value for v is used → from original call (not within for v in vs loop):Failed →try method of buttonframe class?
+ make button methods, call with lambda function with variable in vs loop.

+ results frame
OK buttons
Not OK buttons
fix back buttons to work correctly (need to reference entry within a thread, even after parent window destroyed, when window with back button is created)
+ class buttonFrame(tkinter.Frame) done, not fully generalized. calls button class (passing function and arguments):
+ generalize iteration over a list, provided as argument to the class.
+ v()
+ c()
+ pss()
+ others? (supply list as argument)
+ class button(tkinter.Button) done with some working examples
fully generalize
+ fix window/frame inheritance problem
windows with (self,parent) may need to be named distinctly, to keep destruction and inheritance clear.
+ think through self,parent values, and need for additional window(m) values (e.g., to track window with a frame, (in a frame,) buttonframe, button)
+ fix problem with getps function in lift_do
!Go through all functions and clarify that variables are used according to new structure
call entry, not dependants (guid and problem/opt)
call window, not dependants (frame, ?)
call check, not dependants (name/check, subcheck/ori)
call ButtonFrame, not dependants (cmd) (?make button object use ButtonFrame.cmd, others?
class window(tkinter.Tk)
distinguish between windows that replace their parents (e.g., select a vowel/test), and those that don't (e.g., children of sorting windows)
+ inherit one class from the other.
+ wrapper has two lines: parent.destroy() and window.**init**()
class frame(tkinter.Frame)
+ make this fit in both kind of window, before or after other content
class entry(xmltree.object)
generic fn to take argument of xpath, return value
specific functions to keep name, but wrap around generic function.
a different generic function for each type of lookup function (split them by function structure, loops needed, etc.)
+ lift_do.Entry inherits from lift_get.Entry
+ Add doc for gui layout, to sort and organize window objects and their dependents.
+ tkadhoc
?tkhcecks
+ selections dependency table
get universal window variable of parent for back button
NOT (unnecessary in frame):button class/method which returns/stores final row number, so buttons can continue (or use in indepentent frame)

evaluate use of a lift class
test use of functions as methods for lift and entry classes.
+ids(self)
test use of class methods to add/modify button rows (do all parameters need to be set on calling, or can they be changed after the fact?)
Save incremental changes in mercurial package format, for mailing by WhatsApp, etc., or online when internet is present?
Log each change to external text file (and maybe to comment in LIFT?), including datestamp and guid, with original and change for reconstruction if needed.
Make setup dialogs use a default value (to allow user to return to work quickly), which is stored on exit, with options to changed when needed
language - to file picker to get lift file.
grammatical category
check
segment
Compartmentalize functions (all of the following should be filtered by ps: nouns or verbs, then the other):
1.Get two of these working, as proof of concept first.
input of citation forms (initially, taken from WeSay or FLEx)
input citation tone forms
input/analyze second forms (plurals for noun, imperatives for verbs, or another appropriate form for either, by language)
Each of these will require three things:
populating second field (plural/imperative/other) with secondary data (as added, or constructed)
user inputs raw second form, then we parse the two forms for common segments?
this has the advantage of being a simple data entry for the user
disadvantage: random errors will confuse parser.
user selects from suggested forms, based on possible/likely/known affixes, make changes based on selection
this should work better once some parsing is done, or if affixes are already known.
fewer random mistakes, with user selecting correct option
might introduce bad data, if the user feels constrained by options provided. (Have a big “None of these” button?)
described at https://kent.atoznback.org/2011/10/12/proposal-revisited/
Either method outputs correct parsing to script, which modifies as below:
parsing “word”/lexeme-unit to include only root
check for citation as subset of second form, or vice versa.
if not, check for other correspondence (e.g., if both have affixation)
maximize common segments in the two forms, call that the root (this would need to be confirmed, of course)
moving original “word”/lexeme-unit data to citation field
populate form’s second field, either as input or as selected from suggestion
mark grammatical category (ps) correctly (based on parsing just done), note if a change. (should this be it’s own function?)
citation, second, ps and lexeme forms should be modified concurrently (leaving no

record in a half-modified state)
input second tone forms (check, if same, at least)
input appropriate pictures (currently from WeSay)
input appropriate sound file (currently from WeSay)
check for consistency of ps values (e.g., “n” v “Noun”)
check for consistency of other fields (e.g., Plural v “Morphemes - Plural Form”)
2.check for canonical syllable profiles
iterate through combinations of C’s and V’s, producing regex’s
give a count and produce a stable
give access to data for checking (from the table?)
avoid double dipping by
looking for straight CV(CV) forms first, store guids of found records
then CVCCV, etc., not including already found guids. (to avoid being counted as both C and CC)
multiple sets of controlled sorts:
“check” includes multiple things:
+ filter on a regex for only entries that fit the pattern (one V or C at a time).
+ present each list of entries in a list so the user can read and understand intended glosses (no pictures yet).
this needs to refresh on any change to the data.
+ allow user selection of correct/and or incorrect, or same/different.
IP:allow user correction of data as appropriate (how to constrain this? simple button/UI?)
+ fix vowel button labels,
+ fix v1 < ≠ v2 placement
+ ask for ps, then V (or C?)
+ back button function, with destroy() for last window, one window open at a time.
+ exit button function
+ set vowel selection as buttons, not section box, for V1 = V2 and V1 ≠ V2 windows.
set up multiple windows to ask for V1, then V2 (or is there a better way to do this?)
IP:set up newform function to make modifications as desired, to be confirmed and written to lift later
this needs to be modified multiple times, even if the reference point for change (e.g., of a vowel) is different in the lexeme form.
IP:this needs to show lexeme info if it is the only form info available.
write changes back to xml faithfully, include changes in next sort.
Note where what has been checked (once or twice?) and skip or not, as appropriate.
in the log file?
in the lift file?
start another data checking file?
Do vowels first:
IP:check V1 = V2 roots for V consistency (v1isv2 is only working results function!!!)
follow chain of logic stewarded by computer: each decision has consequences,
follow them to the end of confirming in database and/or writing changes to file.
check V1 for V consistency
check V2 for V consistency

check other (V1 ≠ V2) combos
look for harmony (report output)
Do the same for consonants
Simple UI - for use by illiterate people with little language skills. (WeSay's UI is a great model).
show words on virtual 'cards', with multiple glosses and an image (if available).
make configurable, to include only what is desirable in a given context.
show multiple cards on a screen, with a prompt and response option
clicking one of two/three cards
clicking one of two/three buttons
+ Application Window, with check attributes displayed and modifiable as needed, and "Run Check" button.
+ "Run Check" window, calling appropriate functions for check and Subcheck
+ V1 = V2 (v1isv2)
OK windows (what to do with these? are they needed?)
Not OK windows, with options of how to fix the data.
Evaluate GUI options, including for android:
install and test kivvy: https://kivy.org/doc/stable/installation/installation-linux.html; sudo add-apt-repository ppa:kivy-team/kivy