# Background

My python program execution is giving me problems. The primary problem presents itself as an unexplained second instance of the primary Turtle display. This appears to be a result of an unexpected call/execution of the initial turtle program wx\_square\_loop\_colors.py

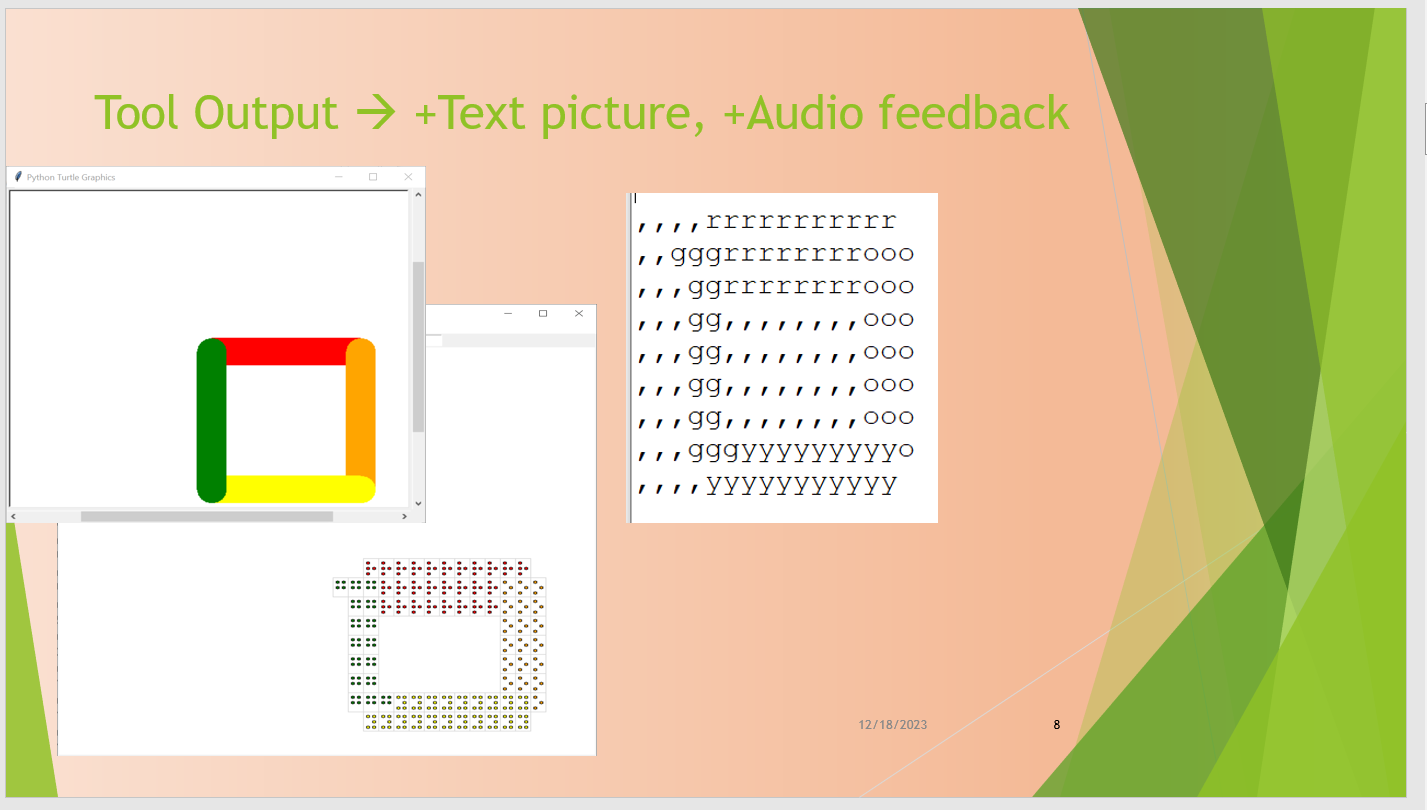
Source code repository: <https://github.com/raysmith619/resource_lib>

Source files: src directory

Setup instructions below

The program supports the execution of simple turtle graphics programs. The purpose of this tool is to facilitate “viewing” of the graphics creation of a beginning blind programmer. The primary program output is a low-resolution text picture. The target for this picture is a Braille embossing machine whose resulting braille output is viewed by the programmer. In addition, there is a graphics portrayal of the braille picture. This window can be traversed, via keyboard actions, giving audio feedback

The following picture summarizes the program actions:



# Problem Description

This problem has occurred during my conversion from a python tkinter/turtle implementation to a tkinter/turtle + wxPython implementation. The conversion has been required for the Braille window implementation because the wxPython window provides a much better interface with screen readers than does tkinter.

When turtle program wx\_square\_loop\_colors.py is executed:

1. The turtle window is created
2. The colored square is drawn
3. The Braille picture window is created
4. **A second version of the turtle window is created**
5. **The colored square is drawn here**

A slight alteration

When one wx\_braille\_display.py is executed directly, only the steps 1,2,3 occur.

wx\_braille\_display.py ends with:

if \_\_name\_\_ == "\_\_main\_\_":

    import wx\_square\_loop\_colors

    #import braille\_display\_test2

Flow of control description:

Execution begins with a modified turtle program as follows:

* Modified turtle program wx\_square\_loop\_colors.py
* from turbo\_braille\_link executes setting up environment
* wx\_turbo\_braille\_link.py is executed
* from wx\_turbo\_braille.py is executed
* creates **bd** an instance of BrailleDisplay()
* overrides done()
* wx\_braille\_display is imported
* Standard turtle graphic functions width, …, right draw picture
* Turtle window is created and picture is drawn
* wx\_square\_loop\_colors.py:done(), overridden by wx\_turtle\_braille.py:done(), calls
* wx\_turtle\_braille.py:done(), which calls, via **bd**, an instance of wx\_braille\_display.py:BrailleDisplay’s done
* wx\_braille\_display.py:done() calls wx\_braille\_display.py:mainloop()
* mainloop() calls display()
* display() outputs the text picture for braille
* display() creates the AudioWindowDisplay – window with audio feedback
* mainloop() “falls” into the wxPython event loop (app.MainLoop()) and sets up the loop to “keep alive” the turtle window

# Setup Instructions

My apologies for the lack of documentation and distribution package. We’re still prototyping.

Our biggest issue is we are in the midst of converting to wxPython based AudioDrawWindow from a tkinter version. The fact that we must continue to use tkinter to support our turtle interface.

## Required packages

### General

numpy

### Graphics

wx – WxPython - used to implement screen reader friendly display window

### Talking Support

pyttsx4 – pyttsx3 has worked but we converted

### Tone Creation

winsound – simple beep creation – we may remove this in favor of more general sounddevice/pysinewave use

sounddevice – stereophonic sound creation

pysinewave – utilities for pitch conversion