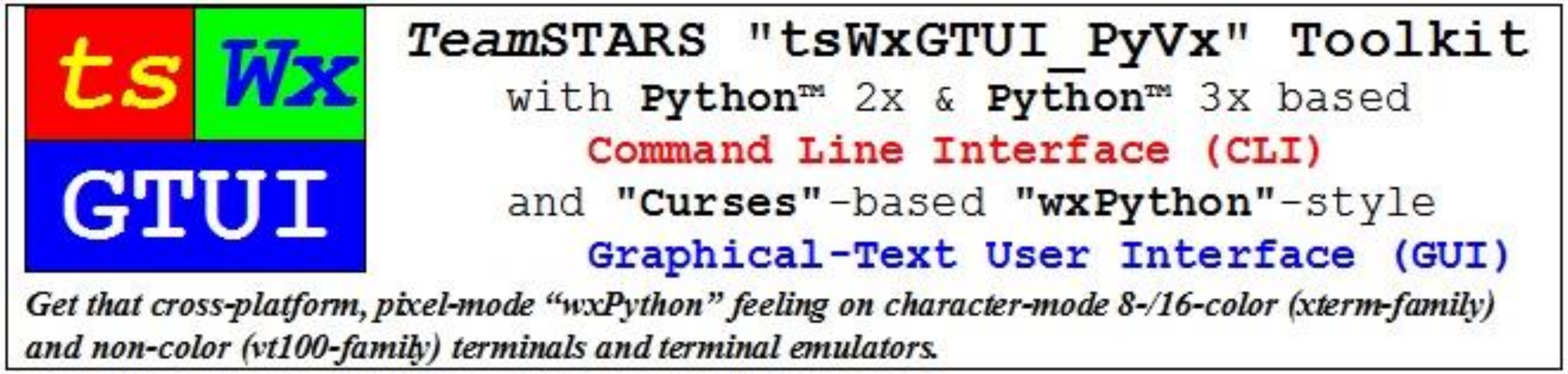


ts Wx GTUI

TeamSTARS "tsWxGTUI_PyVx" Toolkit

with **Python™** 2x & **Python™** 3x based

Command Line Interface (CLI)

and **"Curses"**-based **"wxPython"**-style

Graphical-Text User Interface (GUI)

Get that cross-platform, pixel-mode "wxPython" feeling on character-mode 8-/16-color (xterm-family) and non-color (vt100-family) terminals and terminal emulators.

Table of Contents *(with slide show Hyperlinks)*

Introduction (Table of Contents)

Introduction (Table of Contents)
*Team*STARS "tsWxGTUI_PyVx" Toolkit

- It is a powerful, productive, software development toolkit for rapidly prototyping platform-independent application programs for embedded systems.
- It takes advantage of the cross-platform capabilities of:
  - "**Python** (2x & 3x)" virtual machine programming languages and interpreters
  - "**wxPython** (Python wrapper for wxWidgets)" high level, pixel-mode, graphical widget application programming interface
  - "**Curses** (traditional or new Curses)" terminal control library and low level, text-mode, graphical-style widget application programming interface

Introduction (Table of Contents)
Python Programming Language

Excerpts From Wikipedia, the free encyclopedia:

- "Python is a widely used general-purpose, high-level programming language."
- "Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java."
- "The language provides constructs intended to enable clear programs on both a small and large scale."

Introduction (Table of Contents)
wxPython Graphical User Interface API

Excerpts From Wikipedia, the free encyclopedia:

- "wxPython is a wrapper for the cross-platform GUI API (often referred to as a 'toolkit') wxWidgets (which is written in C++) for the Python programming language."
  - "In computer programming, an application programming interface (API) is a set of routines, protocols, and tools for building software applications. An API expresses a software component in terms of its operations, inputs, outputs, and underlying types. An API defines functionalities that are independent of their respective implementations, which allows definitions and implementations to vary without compromising the interface. A good API makes it easier to develop a program by providing all the building blocks. A programmer then puts the blocks together."
- "It is implemented as a Python extension module (native code)."
- "Like wxWidgets, wxPython is free software."

Introduction (Table of Contents)
Curses Terminal Control Library 1 of 3

Excerpts From Wikipedia, the free encyclopedia:

- “Curses-based software is software whose user interface is implemented through the Curses library, or a compatible library (such as Ncurses).”
- “Curses is designed to facilitate GUI-like functionality on a text-only device, such as a PC running in console mode, a hardware ANSI terminal, a Telnet or SSH client, or similar.”

Introduction (Table of Contents)
Curses Terminal Control Library 2 of 3

- "Curses-based programs often have a user interface that resembles a traditional graphical user interface, including 'widgets' such as text boxes and scrollable lists, rather than the command line interface (CLI) most commonly found on text-only devices. This can make them more user-friendly than a CLI-based program, while still being able to run on text-only devices."

Introduction (Table of Contents)
Curses Terminal Control Library 3 of 3

- “Curses-based software can also have a lighter resource footprint and operate on a wider range of systems (both in terms of hardware and software) than their GUI-based counterparts. This includes old pre-1990 machines along with modern embedded systems using text-only displays.”
- “Curses is most commonly associated with Unix-like operating systems, although implementations for Microsoft Windows also exist.”

Use Cases (Table of Contents)

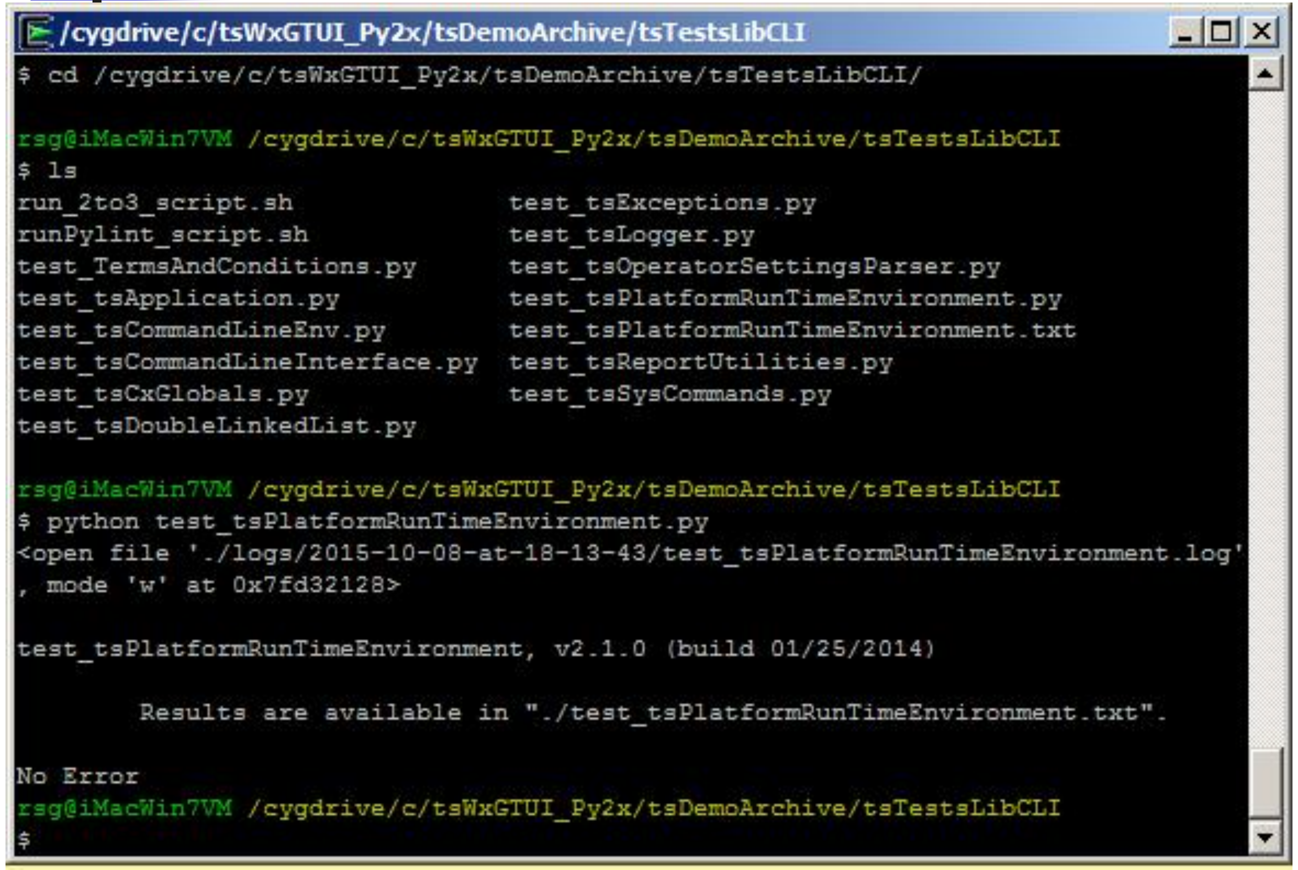
- Sample Screen Shots Command Line Interface (CLI) & Graphical User Interface (GUI)
- Model Platforms Readily Available Consumer-oriented Configurations
- Command Line Interface (CLI)
- Graphical User Interface (GUI)
- Remote Monitoring / Control

Use Cases: (Table of Contents)
Sample Screen Shots

- Command Line Interface (CLI)
  - Sample CLI Display
- Graphical User Interface (GUI)
  - Sample GUI Widgets
  - Sample GUI Scrolled Windows (xterm 8-color)
  - Sample GUI Scrolled Windows (vt100 Black-on-White) & (vt100 White-on-Black)

Use Cases: (Table of Contents)

Sample CLI Display



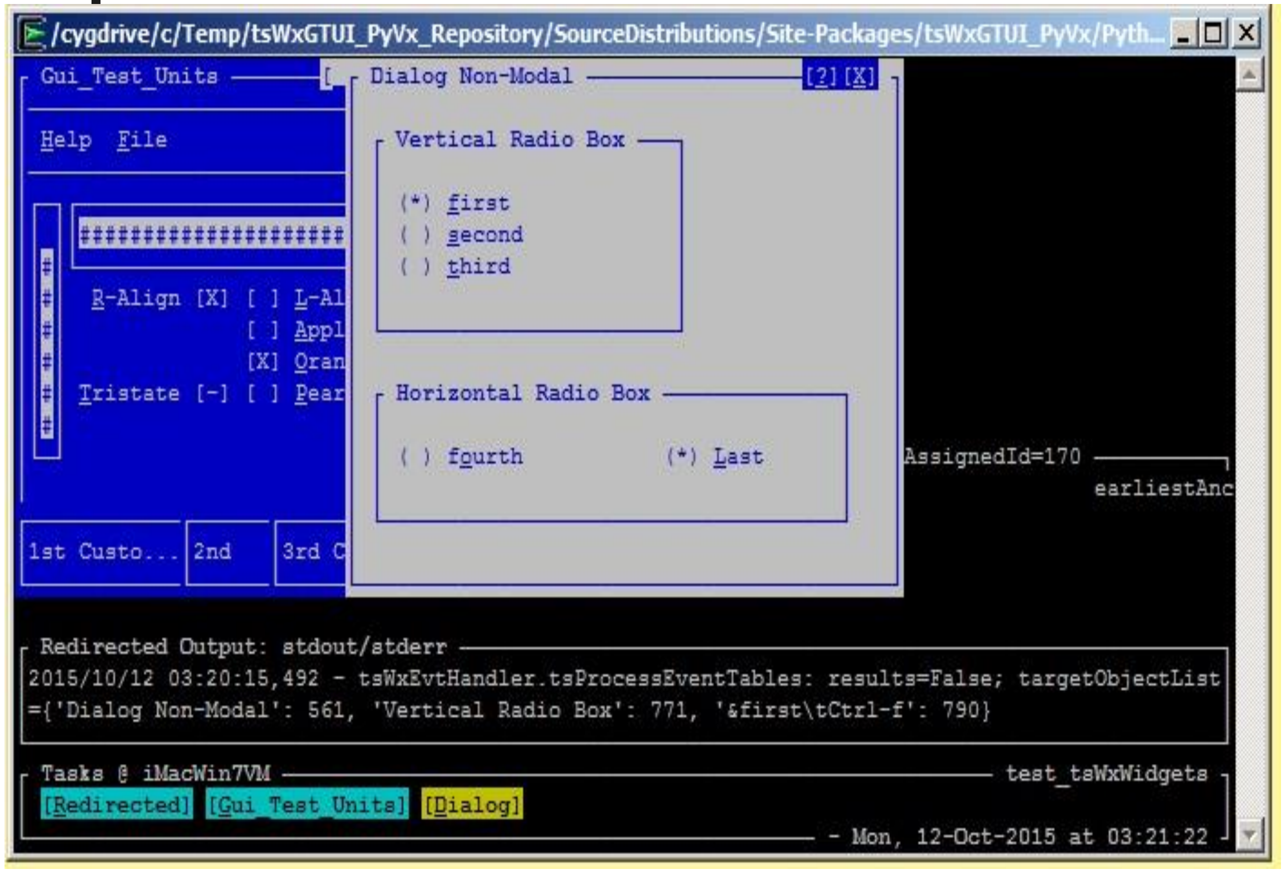
/cygdrive/c/tsWxGTUI_Py2x/tsDemoArchive/tsTestsLibCLI

```
$ cd /cygdrive/c/tsWxGTUI_Py2x/tsDemoArchive/tsTestsLibCLI/

rsg@iMacWin7VM /cygdrive/c/tsWxGTUI_Py2x/tsDemoArchive/tsTestsLibCLI
$ ls
run_2to3_script.sh              test_tsExceptions.py
runPylint_script.sh             test_tsLogger.py
test_TermsAndConditions.py      test_tsOperatorSettingsParser.py
test_tsApplication.py           test_tsPlatformRunTimeEnvironment.py
test_tsCommandLineEnv.py        test_tsPlatformRunTimeEnvironment.txt
test_tsCommandLineInterface.py  test_tsReportUtilities.py
test_tsCxGlobals.py             test_tsSysCommands.py
test_tsDoubleLinkedList.py

rsg@iMacWin7VM /cygdrive/c/tsWxGTUI_Py2x/tsDemoArchive/tsTestsLibCLI
$ python test_tsPlatformRunTimeEnvironment.py
<open file './logs/2015-10-08-at-18-13-43/test_tsPlatformRunTimeEnvironment.log'
, mode 'w' at 0x7fd32128>

test_tsPlatformRunTimeEnvironment, v2.1.0 (build 01/25/2014)

        Results are available in "./test_tsPlatformRunTimeEnvironment.txt".

No Error
rsg@iMacWin7VM /cygdrive/c/tsWxGTUI_Py2x/tsDemoArchive/tsTestsLibCLI
$
```

- The **cd** command, also known as **chdir**, changes the directory as specified.
- The **ls** command lists files in the directory.
- The **python** command executes the named program which displays the location of its results before terminating.

Use Cases: (Table of Contents)

Sample GUI Widgets

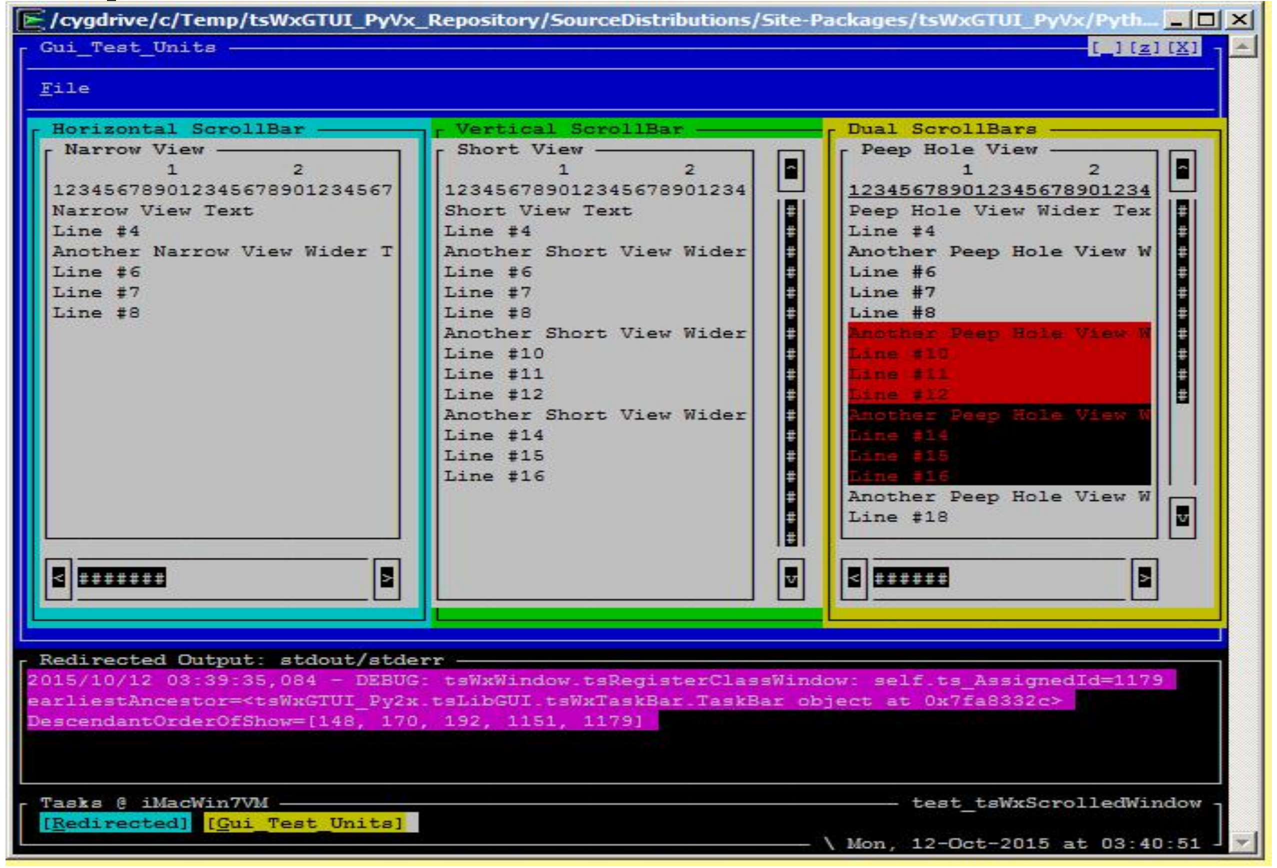


```
/cygdrive/c/Temp/tsWxGTUI_PyVx_Repository/SourceDistributions/Site-Packages/tsWxGTUI_PyVx/Pyth...
Gui_Test_Units ──────[  Dialog Non-Modal ─────────────────────── [?][X]
Help  File                Vertical Radio Box ──
                          (*) first
 ####################     ( ) second
                          ( ) third
 R-Align [X] [ ] L-Al
             [ ] Appl
             [X] Oran
 Tristate [-] [ ] Pear    Horizontal Radio Box ─────────────
                          ( ) fourth          (*) Last        AssignedId=170 ──────────
                                                                            earliestAnc
1st Custo... 2nd  3rd C

Redirected Output: stdout/stderr ─
2015/10/12 03:20:15,492 - tsWxEvtHandler.tsProcessEventTables: results=False; targetObjectList
={'Dialog Non-Modal': 561, 'Vertical Radio Box': 771, '&first\tCtrl-f': 790}

Tasks @ iMacWin7VM ──────────────────────────────────────────── test_tsWxWidgets
[Redirected] [Gui_Test_Units] [Dialog]
──────────────────────────────────────────── - Mon, 12-Oct-2015 at 03:21:22
```

- **Blue Frame** (partially hidden)
  - With Menu Bar, Horizontal & Vertical Gauges, Check Boxes and Status Bar
- **White Dialog**
  - With Window Control Buttons and Radio Boxes & Buttons
- **Black Redirected Output: stdout/stderr Frame**
  - Output of date & time stamped debug statements
  - Output of colorized, date, time & severity-level stamped event notifications (**not shown**)
- **Task Bar Frame**
  - With Network & Program Name, Task (Frame & Dialog) Focus Control Buttons, Idle Time Spinner and Current Date & Time

Use Cases: (Table of Contents)

Sample GUI Scrolled Windows (xterm 8-color)

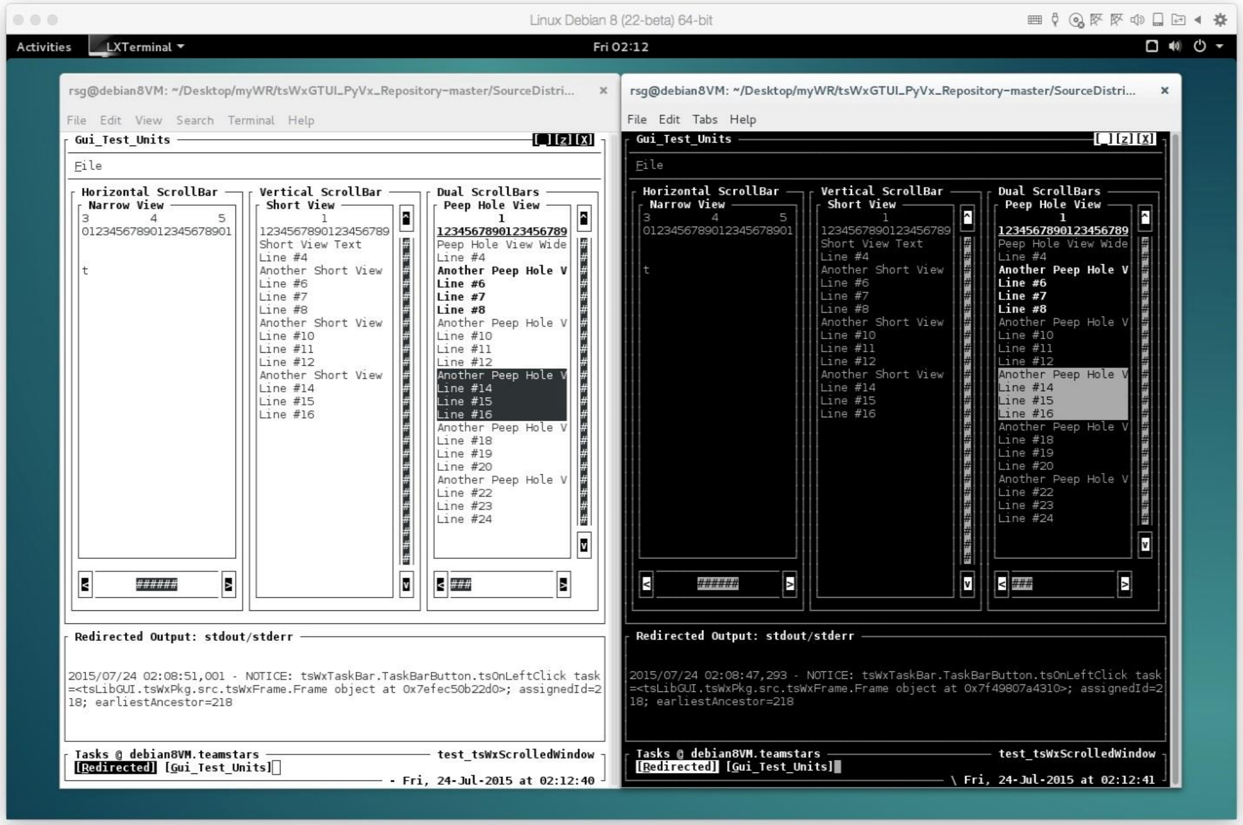


```
/cygdrive/c/Temp/tsWxGTUI_PyVx_Repository/SourceDistributions/Site-Packages/tsWxGTUI_PyVx/Pyth...
 Gui_Test_Units                                                                      [ ] [z] [X]
 File
 Horizontal ScrollBar              Vertical ScrollBar              Dual ScrollBars
  Narrow View                       Short View                      Peep Hole View
            1         2                       1         2                     1         2
  123456789012345678901234567       1234567890123456789012345  ^    1234567890123456789012345  ^
  Narrow View Text                  Short View Text            #    Peep Hole View Wider Tex   #
  Line #4                           Line #4                    #    Line #4                    #
  Another Narrow View Wider T       Another Short View Wider   #    Another Peep Hole View W   #
  Line #6                           Line #6                    #    Line #6                    #
  Line #7                           Line #7                    #    Line #7                    #
  Line #8                           Line #8                    #    Line #8                    #
                                    Another Short View Wider   #    Another Peep Hole View W   #
                                    Line #10                   #    Line #10                   #
                                    Line #11                   #    Line #11                   #
                                    Line #12                   #    Line #12                   #
                                    Another Short View Wider   #    Another Peep Hole View W
                                    Line #14                   #    Line #14
                                    Line #15                   #    Line #15
                                    Line #16                   #    Line #16
                                                               #    Another Peep Hole View W
                                                               #    Line #18                   v
  < #######                  >                                 v    < ######                 >

 Redirected Output: stdout/stderr
2015/10/12 03:39:35,084 - DEBUG: tsWxWindow.tsRegisterClassWindow: self.ts_AssignedId=1179
earliestAncestor=<tsWxGTUI_Py2x.tsLibGUI.tsWxTaskBar.TaskBar object at 0x7fa8332c>
DescendantOrderOfShow=[148, 170, 192, 1151, 1179]

 Tasks @ iMacWin7VM                                                 test_tsWxScrolledWindow
 [Redirected] [Gui_Test_Units]
                                                        \ Mon, 12-Oct-2015 at 03:40:51
```

- **Blue Application Frame**
  - With Menu Bar, Window Size & Close Control Buttons and three scrollable panels.
- **Three Scrollable Application Panels (Cyan Horizontal, Green Vertical & Yellow Dual)**
  - Each with Multi-Colored & Non-Colored Text, Horizontal and/or Vertical scroll bars, associated clickable arrow buttons and clickable gauge depicting relative size and position or displayed text.
- **Black Redirected Output: stdout/stderr Frame**
  - Output of colorized, date, time & severity-level stamped event notifications (**debug-level shown**)
- **Task Bar Frame**
  - With Network & Program Name, Task (Redirected & Gui_Test_Units) Focus Control Buttons, Idle Time Spinner and Current Date & Time

Use Cases: ([Table of Contents]) Sample GUI Scrolled Windows

(vt100 Black-on-White) & (vt100 White-on-Black)



rsg@debian8VM: ~/Desktop/myWR/tsWxGTUI_PyVx_Repository-master/SourceDistri...

File Edit View Search Terminal Help

```
Gui_Test_Units                                                   [ ][z][X]
File
Horizontal ScrollBar    Vertical ScrollBar        Dual ScrollBars
 Narrow View             Short View                Peep Hole View
3         4         5               1                       1
0123456789012345678901  1234567890123456789       1234567890123456789
                        Short View Text           Peep Hole View Wide
                        Line #4                   Line #4
t                       Another Short View        Another Peep Hole V
                        Line #6                   Line #6
                        Line #7                   Line #7
                        Line #8                   Line #8
                        Another Short View        Another Peep Hole V
                        Line #10                  Line #10
                        Line #11                  Line #11
                        Line #12                  Line #12
                        Another Short View        Another Peep Hole V
                        Line #14                  Line #14
                        Line #15                  Line #15
                        Line #16                  Line #16
                                                  Another Peep Hole V
                                                  Line #18
                                                  Line #19
                                                  Line #20
                                                  Another Peep Hole V
                                                  Line #22
                                                  Line #23
                                                  Line #24

Redirected Output: stdout/stderr

2015/07/24 02:08:51,001 - NOTICE: tsWxTaskBar.TaskBarButton.tsOnLeftClick task
=<tsLibGUI.tsWxPkg.src.tsWxFrame.Frame object at 0x7efec50b22d0>; assignedId=2
18; earliestAncestor=218

Tasks @ debian8VM.teamstars                          test_tsWxScrolledWindow
[Redirected] [Gui_Test_Units]
                                             - Fri, 24-Jul-2015 at 02:12:40
```

rsg@debian8VM: ~/Desktop/myWR/tsWxGTUI_PyVx_Repository-master/SourceDistri...

File Edit Tabs Help

```
Gui_Test_Units                                                   [ ][z][X]
File
Horizontal ScrollBar    Vertical ScrollBar        Dual ScrollBars
 Narrow View             Short View                Peep Hole View
3         4         5               1                       1
0123456789012345678901  1234567890123456789       1234567890123456789
                        Short View Text           Peep Hole View Wide
                        Line #4                   Line #4
t                       Another Short View        Another Peep Hole V
                        Line #6                   Line #6
                        Line #7                   Line #7
                        Line #8                   Line #8
                        Another Short View        Another Peep Hole V
                        Line #10                  Line #10
                        Line #11                  Line #11
                        Line #12                  Line #12
                        Another Short View        Another Peep Hole V
                        Line #14                  Line #14
                        Line #15                  Line #15
                        Line #16                  Line #16
                                                  Another Peep Hole V
                                                  Line #18
                                                  Line #19
                                                  Line #20
                                                  Another Peep Hole V
                                                  Line #22
                                                  Line #23
                                                  Line #24

Redirected Output: stdout/stderr

2015/07/24 02:08:47,293 - NOTICE: tsWxTaskBar.TaskBarButton.tsOnLeftClick task
=<tsLibGUI.tsWxPkg.src.tsWxFrame.Frame object at 0x7f49807a4310>; assignedId=2
18; earliestAncestor=218

Tasks @ debian8VM.teamstars                          test_tsWxScrolledWindow
[Redirected] [Gui_Test_Units]
                                             \ Fri, 24-Jul-2015 at 02:12:41
```

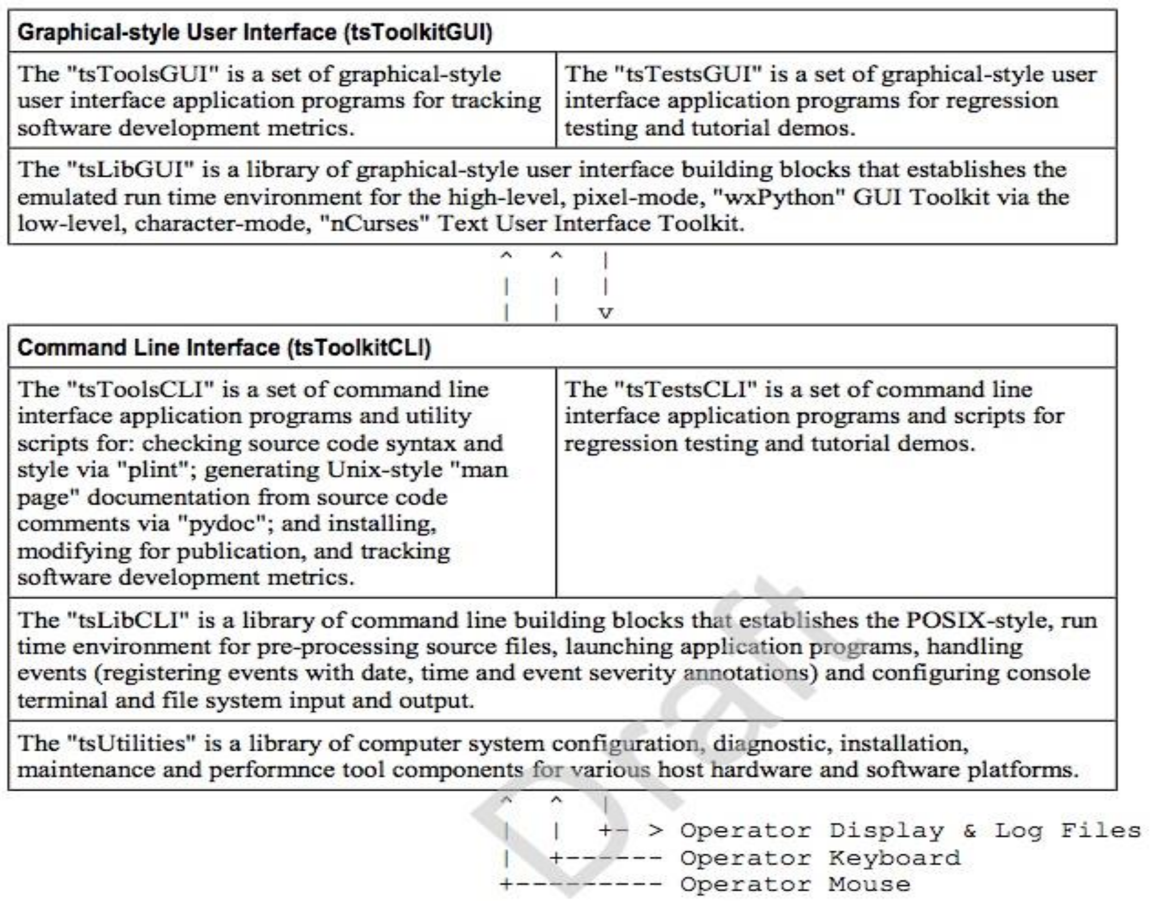
- **Outer-Most Application Frames**
  - With Menu Bar, Window Size & Close Control Buttons and three scrollable panels.
- **Three Scrollable Application Panels (Horizontal, Vertical & Dual)**
  - Each with Multi-Colored & Non-Colored Text, Horizontal and/or Vertical scroll bars, associated clickable arrow buttons and clickable gauge depicting relative size and position or displayed text.
- **Black Redirected Output: stdout/stderr Frame**
  - Output of colorized, date, time & severity-level stamped event notifications (**debug-level shown**)
- **Task Bar Frame**
  - With Network & Program Name, Task (Redirected & Gui_Test_Units) Focus Control Buttons, Idle Time Spinner and Current Date & Time

Use Cases: (Table of Contents)
Block Diagrams

- Toolkit Building Block Diagrams
- Non-Networked (Stand-Alone) System (HW-SW) Block Diagram
- Networked (Stand-Among) System (HW-SW) Block Diagram

Use Cases: (Table of Contents)

Toolkit Building Block Diagram



Graphical-style User Interface (tsToolkitGUI)

The "tsToolsGUI" is a set of graphical-style user interface application programs for tracking software development metrics.	The "tsTestsGUI" is a set of graphical-style user interface application programs for regression testing and tutorial demos.

The "tsLibGUI" is a library of graphical-style user interface building blocks that establishes the emulated run time environment for the high-level, pixel-mode, "wxPython" GUI Toolkit via the low-level, character-mode, "nCurses" Text User Interface Toolkit.

```
^   ^   |
|   |   |
|   |   v
```

Command Line Interface (tsToolkitCLI)

The "tsToolsCLI" is a set of command line interface application programs and utility scripts for: checking source code syntax and style via "plint"; generating Unix-style "man page" documentation from source code comments via "pydoc"; and installing, modifying for publication, and tracking software development metrics.	The "tsTestsCLI" is a set of command line interface application programs and scripts for regression testing and tutorial demos.

The "tsLibCLI" is a library of command line building blocks that establishes the POSIX-style, run time environment for pre-processing source files, launching application programs, handling events (registering events with date, time and event severity annotations) and configuring console terminal and file system input and output.

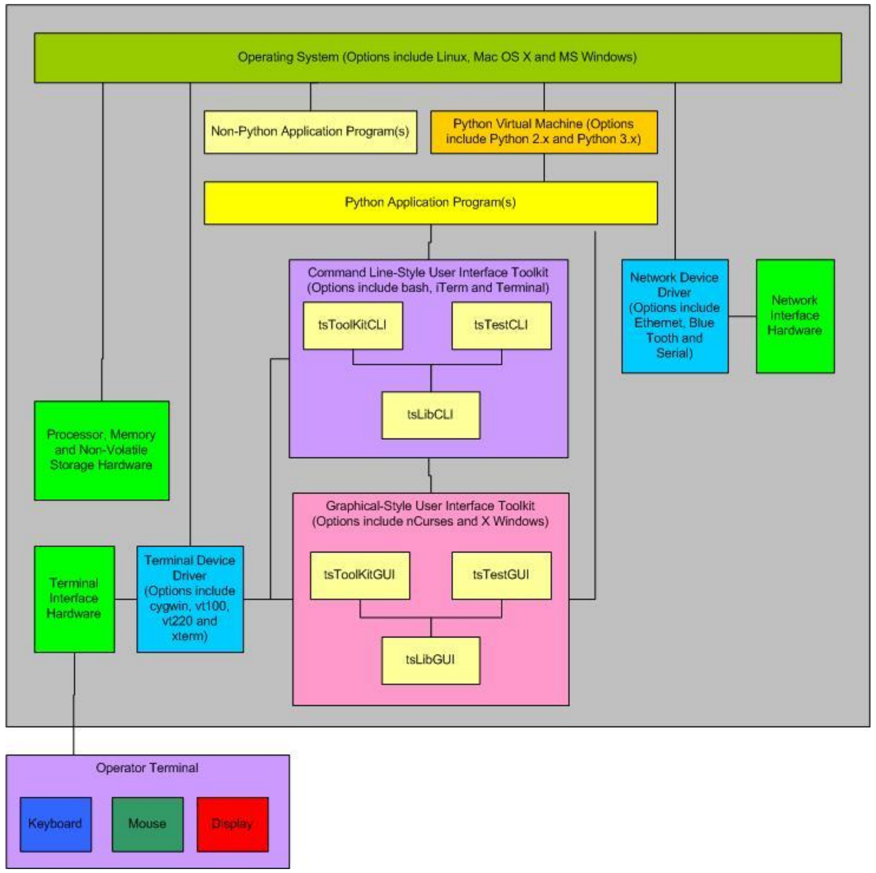
The "tsUtilities" is a library of computer system configuration, diagnostic, installation, maintenance and performnce tool components for various host hardware and software platforms.

```
^   ^   |
|   |   +- > Operator Display & Log Files
|   +------- Operator Keyboard
+----------- Operator Mouse
```

- Cross-platform, Character-mode **Curses**-based emulation of Pixel-mode cross-platform **wxPython** GUI Application Programming Interface
- Cross-platform, Linux-/Unix-like **POSIX**-based Command Line Interface
- **Operator's Computer Terminal** with Display, Keyboard and Mouse

Use Cases: (Table of Contents)

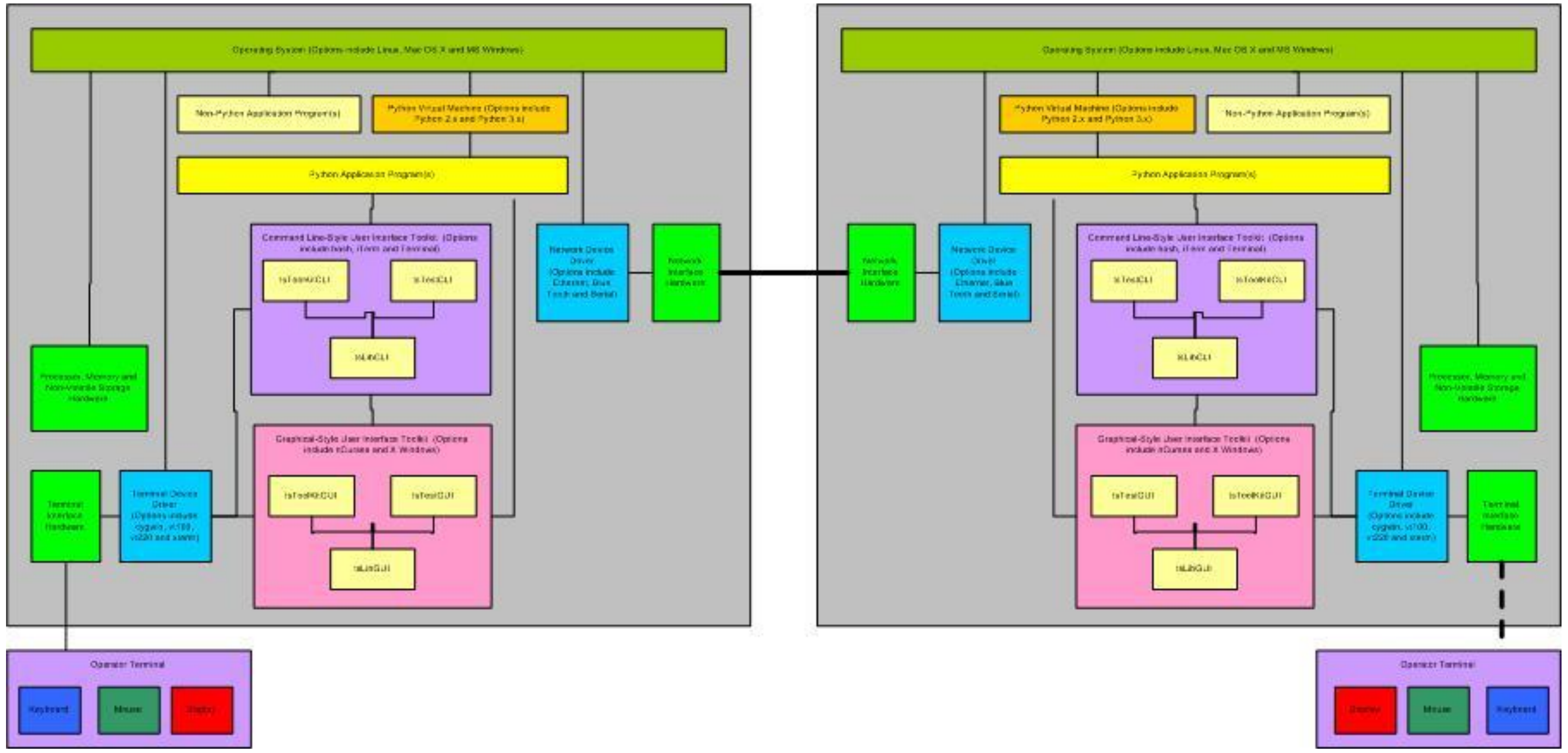
Non-Networked (Stand-Alone) Mode System (HW-SW) Block Diagram



Operating System (Options include Linux, Mac OS X and MS Windows)
Non-Python Application Program(s)
Python Virtual Machine (Options include Python 2.x and Python 3.x)
Python Application Program(s)
Command Line-Style User Interface Toolkit (Options include bash, iTerm and Terminal)
tsToolKitCLI
tsTestCLI
tsLibCLI
Network Device Driver (Options include Ethernet, Blue Tooth and Serial)
Network Interface Hardware
Processor, Memory and Non-Volatile Storage Hardware
Graphical-Style User Interface Toolkit (Options include nCurses and X Windows)
tsToolKitGUI
tsTestGUI
tsLibGUI
Terminal Interface Hardware
Terminal Device Driver (Options include cygwin, vt100, vt220 and xterm)
Operator Terminal
Keyboard
Mouse
Display

Use Cases: (Table of Contents)

Networked (Stand-Among) Mode System (HW-SW) Block Diagram



Operating System (Options include Linux, Mac OS X and MS Windows)
Non-Python Application Program(s)
Python Virtual Machine (Options include Python 2.x and Python 3.x)
Python Application Program(s)
Command Line-Style User Interface Toolkit (Options include bash, iTerm and Terminal)
tsToolkitCLI
tsTestCLI
tsLibCLI
Network Device Driver (Options include Ethernet, Blue Tooth and Serial)
Network Interface Hardware
Processor, Memory and Non-Volatile Storage Hardware
Graphical-Style User Interface Toolkit (Options include nCurses and X Windows)
tsToolkitGUI
tsTestGUI
tsLibGUI
Terminal Interface Hardware
Terminal Device Driver (Options include cygwin, vt100, vt220 and xterm)
Operator Terminal
Keyboard
Mouse
Display

Use Cases: (Table of Contents)

Model Platforms

Readily available consumer-oriented configurations suitable for use as software & documentation development systems and as tag along “simulations” of customized application-specific embedded systems.

- Basic “Development” Laptop and Pseudo “Embedded” System
  - Platform with minimal resources: single-core processor with low horse-power and only enough memory to support host operating systems with both command line and graphical user interfaces.
- Accessorized “Development” Laptop and Guest “Embedded” System
  - Platform with moderate resources: dual-core processor with average horse-power and enough memory to support host and a single guest operating system with both command line and graphical user interfaces.
- Accessorized “Development” Desktop and Guest “Embedded” Systems
  - Platform with additional resources: quad-core processor with sufficient horse-power and memory to support host and several guest operating systems with both command line and graphical user interfaces.

Use Cases: Model Platforms
Basic “Development” Laptop and Pseudo “Embedded” System

- **1998 Dell Inspiron 7000 Hardware**
  - 366 MHz **Intel Pentium II** processor
  - 384 MB RAM
  - 15.6” **VGA** (640x480) / **SVGA** (1024x768) pixel LCD display
  - Two Interchangeable 32 GB (4200 RPM) ATA hard drives
    - **Microsoft Windows XP**
    - **Ubuntu Linux** 12.04 LTS
  - **Xircom** Ethernet and 3Com WiFi Wireless Plug-in Network adapters for **Microsoft Windows XP**
  - **Linksys** WiFi Wireless Plug-in Network adapter for **Ubuntu Linux** 12.04 LTS
- **Development / Pseudo (non-optimized) Embedded Software**
  - **Microsoft Windows XP**
    - Cygwin 1.7 (Python 2x & 3x)
    - Office 2002
    - Xemacs
- **Development / Pseudo (non-optimized) Embedded Software**
  - **Ubuntu Linux** 12.04 LTS
    - GNOME Desktop
    - LibraOffice
    - Xemacs
    - Python 2x & 3x

Use Cases: Model Platforms

Accessorized “Development” Laptop and Guest “Embedded” System

- **2007 Apple MacBook Pro Hardware**
  - 2.33 GHz Intel Core 2 Duo processor
  - 4 GB RAM
  - 17” 1920x1200 pixel LCD display
  - 160 GB (5400 RPM) SATA 1.5 Gb/s internal hard drive
  - 1.5 TB (7200 RPM) SATA 3 Gb/s external hard drive
  - Ethernet Network Adapter
  - WiFi Wireless Network Adapter
- **Development / Embedded Software**
  - MAC OS X 10.7.5 Lion
  - Wing IDE 3-4
  - LibreOffice
  - Xemacs
  - Python 2x & 3x
- **Guest (non-optimized) Embedded Software**
  - **Parallels Desktop** 8 Hypervisor for running GuestOS:
    - Linux (Fedora 20 32-bit, OpenSuSE 12.2 32-bit, Scientific (CentOS) 6.4-6.5 64-bit, Ubuntu 12.04 32-bit) with Python 2.7 and 3.2 with Wing IDE 3, LibraOffice and XEmacs
    - Microsoft Windows (XP, 7, 8 & 8.1 each with Cygwin 1.7.8) with Wing IDE 3, AuthorIt-5, Office 2002 & XEmacs
    - Unix (PC-BSD 9.2-10.0, OpenIndiana 151a3 & OpenSolaris 11) with LibraOffice and Xemacs
  - **VMware Fusion** 5 Hypervisor for running GuestOS:
    - Linux (OpenSuSE 13.1)
    - Microsoft Windows (3.1)

Use Cases: Model Platforms

Accessorized "Development" Desktop and Guest "Embedded" System

- **2013 Apple iMac Desktop Hardware**
  - 3.5 GHz Intel Quad Core i7 processor
  - 16 GB RAM
  - 27″ 2560x1440 pixel LCD display
  - 3 TB (7200 RPM) SATA 6 Gb/s internal hard drive with 128 GB Solid State Flash memory
  - Ethernet Network Adapter
  - WiFi Wireless Network Adapter
- **Development / Embedded Software**
  - MAC OS X 10.11 El Capitan
  - Wing IDE 5
  - LibreOffice
  - Microsoft Office for Mac 2011
  - Xemacs
  - Python 2x & 3x
- **Guest (non-optimized) Embedded Software**
  - **Parallels Desktop** 11 Hypervisor for running GuestOS:
    - Linux (Centos 7, Debian 8, Fedora 22, OpenSuSE 13.2, Scientific 7 & Ubuntu 14.04 LTS & 15.04) with Wing IDE 5, LibraOffice and XEmacs
    - Microsoft Windows (XP, 7, 8, 8.1 & 10) with Wing IDE 5, AuthorIt-5, Office 2002 & XEmacs
    - Unix (FreeBSD 11/PC-BSD 11, OpenIndiana 151a8 & OpenSolaris 11) with LibraOffice and Xemacs
  - **VMware Fusion** 7 Hypervisor for running GuestOS:
    - Linux (OpenSuSE 13.1)
    - Microsoft Windows (2000)

Use Cases: (Table of Contents)
Command Line Interface (CLI)

- Output to the User:
  - A chronological sequence of lines of text written from top to bottom and then scrolling off the top as each new line is written to the bottom of the terminal display.
- Input from the User:
  - Via a computer terminal keyboard with input echoed to the display below the previous output.

Use Cases: (Table of Contents)

CLI Building Blocks (tsLibCLI)

- Application Building Blocks
  - tsCommandLineInterface
  - tsDoubleLinkedList
  - tsOperatorSettingsParser
  - tsReportUtilities
- Application Diagnostics
  - tsExceptions
  - tsLogger
- Application Configuration
  - tsCxGlobals
  - tsGistGetTerminalSize
  - tsPlatformRunTimeEnviroment
- Application Launchers
  - tsApplication
  - tsCommandLineEnv
  - tsSysCommands

Use Cases: (Table of Contents)
CLI Tools (tsToolsCLI) 1 of 2

- Developer Tools
  - tsStripComments
  - tsStripLineNumbers
  - tsTreeCopy
  - tsTreeTrimLines.py
- Troubleshooter Tools
  - tsPlatformQuery

Use Cases: (Table of Contents)
CLI Tools (tsToolsCLI) 2 of 2

- Developer Productivity Tracking Tools
  - tsLinesOfCodeProjectMetrics
    - Total Files, Lines of Code & Comments
    - Subtotals (by language) Files, Lines of Code & Comments
    - Estimates labor cost/value, staffing and schedule per COCOMO-81 model. It computes software development effort as a function of program size and a set of "cost drivers" that include subjective assessments of product, hardware, personnel, and project attributes.

Use Cases: (Table of Contents)
Graphical User Interface (GUI)

- Output to the user of character strings to application-specified column and row (line) fields on a computer terminal display.
- Input from the user via a pointing device (such as mouse, trackball, touchpad or touch screen) that is moved into a position by its operator before a mouse button is clicked to activate a function button, radio button , checkbox or keyboard input position.
- Input from the user via a computer terminal keyboard that is echoed as output to a reserved area of the display that is written from top to bottom and then scrolling off the top as each new line is written to the bottom of the reserved area.

Use Cases: (Table of Contents)
GUI Building Blocks (tsLibGUI)

- Application Building Blocks (partial listing)
  - Top-Level GUI Objects
    - tsWxFrame
    - tsWxDialog
  - Lower-Level GUI Objects
    - tsWxPanel
    - tsWxStatusBar
  - GUI Controls
    - wxWxButton
    - tsWxCheckbox
    - tsWxGauge
    - tsWxMenuBar
    - tsWxRadioButton
    - tsWxScrollBar
    - tsWxTaskBar
  - GUI Events
    - Keyboard / Mouse / Tmer
  - GUI Sizers
    - tsWxBoxSizer
    - tsWxGridSizer
- Application Configuration
  - tsWxGlobals
  - tsWxGraphicalTextUserInterface
  - tsWxSplashScreen
- Application Diagnostics
  - tsWxLog (Future)
- Application Launchers
  - tsWxApp
  - tsWxPyApp
  - tsWxPyOnDemandOutputWindow
  - tsWxPySimpleApp
  - tsWxMultiFrameEnv

Use Cases: (Table of Contents)
Remote Monitoring / Control

- Once you've logged into your local computer, you may then login to a remote computer using one or more secure shells ("ssh") or non-secure shells ("rsh") provided by the local operating system.
- The Secure Shell ("ssh") is a cryptographic network protocol for secure data communication, remote command-line login, remote command execution, and other secure network services between two networked computers. It connects, via a secure channel over an insecure network, a server and a client running "ssh" server and "ssh" client programs, respectively.
- The remote shell ("rsh") is a command line computer program that can execute shell commands as another user, and on another computer across a computer network. The remote system to which "rsh" connects runs the "rsh" daemon ("rshd").
- The local and remote operating system's command line interfaces provides access to associated terminal interface and the TeamSTARS "tsWxGTUI_PyVx" Toolkit's Python and "nCurses" based character-mode user interfaces which then communicate.
- This enables you to monitor and control one or more remote and local application programs, from the convenience of your local computer terminal, with greater speed and efficiency than possible with the larger communication traffic associated with pixel-mode.

Project (Table of Contents)

- **Objectives**
  - Goals (Capabilities)
  - Non-Goals (Limitations)
- **Plans**
  - Applicable Technology
  - Software Architecture
  - Productivity Tools
  - Administrative Utilities
- **Implementation**
  - Documentation for Training, Engineering, Operation & Troubleshooting
  - Developer Sandboxes
  - Installable Site Packages
  - Release & Publication

Project Objectives:
Goals (Mission Critical Capabilities)

- Provide a foundation of building block libraries, tools and utilities that:
  - lets you work more quickly and integrate your systems more effectively
  - facilitates the rapid prototyping of application software used to monitor, control and coordinate **Mission Critical Equipment**
- The foundation components must be general purpose and re-usable in order to support the following markets:
  - **Commercial** (as in building energy management)
  - **Industrial** (as in power generation)
  - **Medical** (as in cat-scan)
  - **Military** (as in weapon control)
- Foundation components must facilitate the rapid prototyping of application software on diverse assortment of:
  - **General Purpose Desktop & Laptop Computer Systems**
  - **Application-Specific Embedded Computer Systems**:
    - **Automation** (as in robotics)
    - **Communication** (as in network traffic)
    - **Control** (as in supervisory and feedback of equipment and processes)
    - **Diagnostic** (as in hardware and software failure modes and effects)
    - **Instrumentation** (as in sensor data acquisition and analysis)
    - **Simulation** (as in flight control)

Project Objectives:
Goals (Cross-Platform Capabilities)

- Foundation components must be compatible with and portable to a diverse assortment of popular and readily available hardware and software platforms:
  - **Open**
    - HW as in Arm & Intel microprocessor components
    - SW as in GNU’s Unix-like tool components & Linus Torvalds’ Linux operating system kernel components
  - **Proprietary**
    - HW as in Dell, HP, IBM & Lenovo systems
    - SW as in Microsoft’s Windows & Oracle’s Solaris operating system components

Project Objectives:

Goals (Architecture Capabilities)

- Foundation architecture must be modular to support product life-cycle.
  - **Segregation of Python language generations (such as 1x, 2x and 3x)** to facilitate:
    - Future Back-Porting Python 1x from Python 2x.
    - Future Back-Porting Python 2.0.x-2.6.x from Python 2.7.x using "**Six**", a Python 2.x and 3.x compatibility library
    - Porting Python 3.x from Python 2x using Python syntactical converter "**2to3**"
    - Future Porting Python 4x from Python 3x using Python syntactical converter "3**to4**"
  - **Non-Installable Developer Sandbox** to facilitate foundation component development, maintenance and troubleshooting.
  - **Installable Site-Package** to facilitate foundation user's application software development and deployment.

- Foundation components must be customizable in order to support a diverse assortment of operators & organizations:
  - **Customizable "**txCxGlobals" file of organization-/product-specific usage terms & conditions and operator-specific CLI theme-based feature settings
  - **Customizable "**tsWxGlobals" file of organization-/product-specific GUI theme-based feature settings

Project Objectives:
Goals (Documentation Audience)

- **Prospective & New Users** seek reason to dig deeper or look elsewhere
- **Student Users** seek rationale for unfamiliar computer hardware and software technology usage
- **Intermediate Users** seek rationale for architecture, design and implementation of unfamiliar “Python”, "Curses" and "wxPython" programming techniques
- **Advance Users** seek rationale for unfamiliar project planning and engineering techniques
- **Expert Users** seek rationale for unfamiliar troubleshooting, maintenance, porting and enhancement techniques

Project Objectives:
Goals (Documentation Capabilities)

- Documentation for system administrators & installers, developers & maintainers, operators, troubleshooters and students.
  - Establish a reference library (for Objectives, Plans, Requirements, Architecture, Design, Implementation, Debug, Test & Release)
    - Orients & Trains new contributors & users
    - Focuses or reminds contributors of goals, non-goals, plans and unresolved issues
  - Establish a convenient reference library of third-party material, found on internet, which was relevant but might eventually be subject to change or removal
    - Computer & System Engineering (Definitions, Theory, Technology & Practices)
    - External Goods & Services Resources

Project Objectives:

Goals (Engineering Notebook Capabilities)

- Typical engineering project information with commentaries describing rationale and evolutionary changes (for System, Hardware, Software & Interfaces):
  - Concept, Dictionary, Use Cases & Development Plan
  - Requirement, Design, Test & Qualification Specifications
  - Release Notes & User’s Manual
- Typical engineering project contributor information:
  - Document Authoring & Publishing Tools
  - Software Development Tools
  - Introduction and Training
- In various formats with text, tables and complex graphical images requiring an office suite (such as from “Adobe”, “Microsoft‟, “LibreOffice” etc.) application programs typically found on a general purpose desktop computer system.

Project Objectives:

Goals (User “How-to-Guide” Capabilities)

- Documents
  - Typical install, configure, operate and troubleshoot how-to guides with applicable terms and conditions for usage and redistribution.
  - In a plain text format suitable for embedded systems with only character-mode terminals.
- Manual Pages
  - Typical on-line information about Command Line Interface & Graphical User Interface use and application programming for each building block and tool.
  - In a plain text format suitable for embedded systems with only character-mode terminals.

Project Objectives:
Non-Goals (Host Virtual Machine Limitation)

- Foundation will NOT provide *"Magical"* Host Virtual Machines which:
  - **Runs incompatible Processor & Operating System Specific Applications:**
    - You should **NOT** expect to be able to run application programs designed specifically for one processor make & model and one make & model operating system on a different processor and operating system.
    - You should **ONLY** expect to be able to run applications designed to run on a Python "virtual machine" that itself has been designed (by the Python Software Foundation) for the specific processor & operating system, tested and certified to correctly interpret and execute source code appropriate for the Python language generation (such as 1x, 2x or 3x).

Project Objectives:

Non-Goals (GUI Virtual Machine Limitation)

- Foundation will NOT provide *"Magical"* GUI Virtual Machines which:
  - **Run incompatible GUI Applications:**
    - You should **NOT** expect to be able to run pixel-mode “wxPython”, “wxWidgets”, “Qt” or “Tcl/Tk” applications that can copy graphic images from a file to the display and dynamically construct and output an array of pixels to the display which depicts the desired icons, graphic objects and text images.
    - You should **ONLY** expect to be able to run character-mode “wxPython”-style GUI applications designed to dynamically construct and output to the display an array or sequence Curses-standard alpha-numeric, punctuation and line-drawing characters (with escape sequences to control output to a desired display screen column and row position).

Project Objectives:
Non-Goals (Retrofit Limitations)

- Foundation will **NOT** provide *"Magical"* Python Virtual Machine cross-platforms for use with a diverse assortment of obsolete Hardware and Software platforms.
  - **Open** (HW such as 8-/16-bit Intel & Motorola microprocessors, 32-/64-bit Intel iAPX 432, i860; SW source code such as implemented in assembler, Ada, C/C++, FORTRAN and Python 1.x languages)
  - **Proprietary** (HW as in Digital Equipment Corp. & SGI systems; SW such as VAX/VMS & IRIX operating systems)
- Even if others have or could obtain such long discontinued platforms, this Foundation author neither has nor seeks funding to obtain, reconstruct or simulate such platforms.

Project Plans:
Adopt Cross-Platform Technology

- Apply Hardware & Software Technology that lets you work more quickly and integrate your systems more effectively:
  - Popular, readily available and/or within the project budget
  - Suitable for rapid prototyping
  - Field proven with a long term track record of support
  - Software & Documentation must be free to study, modify, use and redistribute

Project Plans:
Adopt Modular Software Architecture

- Provide a framework for cross-platform software development:
  - **Libraries** of Application and Troubleshooting Building Block components.
  - **Tools** for Facilitating and Tracking developer productivity.
  - **Utilities** for Monitoring and Changing hardware and software configuration.
  - **Tests** (Unit, Integration, System, Regression and Acceptance) for design verification and quality assurance.
  - **Examples** for Algorithms, Coding Style, Programmer Productivity Metrics and System Performance.

Project Plans:

Adopt Cross-Platform Software Environment

- **POSIX**, a Unix-like operating system complying with the Portable Operating System Interface:
  - **Apple Mac OS X** (Darwin- & BSD-based Unix)
  - **GNU/Linux** (combination of free Unix-like GNU tools and free Linux kernel)
  - **Microsoft Windows** (requires **Cygwin**, the free Linux-like environment and command-line interface plug-in from Red Hat)
  - **Unix** (derived directly or indirectly from the original AT&T UNIX)
- **Python**, an interpreted, object-oriented programming language and cross-platform virtual machine:
  - **Python 3x** (**actively evolving & maintained** 3rd generation language)
  - **Python 2x** (**mature & actively maintained** 2nd generation language)
  - **Python 1x** (**obsolete & unmaintained** 1st generation language)
- **wxPython**, a cross-platform GUI Application Programming Interface

Project Plans:
Adopt Python 2x Source Code Plan

- Develop software in mature & actively maintained Python 2x (2nd generation language)
  - Create non-installable Python 2x **Developer Sandbox** to facilitate troubleshooting
    - "__init__.py" defines nested package structure and dependency relationships
    - Modules import from other modules & packages within "try-except" logic to report import errors
  - Create installable Python 2x **Site-Package** to facilitate to use of Toolkit building blocks in same manner as library components registered in Python Global Module Index.
    - Copy Python 2x **Developer Sandbox**
    - Replace "__init__.py" modules with empty ones
    - Replace "try-except" import logic with explicit references to site-package identifier

Project Plans:

Adopt Python 3x Source Code Plan

- Develop software in actively evolving & maintained Python 3x (3rd generation language)
  - Create non-installable Python 3x **Developer Sandbox** to facilitate troubleshooting
    - Copy non-installable Python 2x **Developer Sandbox**
    - Convert syntax of Python 2x to Python 3x (3rd generation language) using Python "**2to3"** utility
    - Debug to identify and resolve remaining issues
  - Create installable Python 3x **Site-Package** to facilitate to use of Toolkit building blocks in same manner as library components registered in Python Global Module Index.
    - Copy Python 3x **Developer Sandbox**
    - Replace "__init__.py" modules with empty ones
    - Replace "try-except" import logic with explicit references to site-package identifier

Project Plans:

Adopt Development, Debug and Test Environment

- Representative & Readily Available
  - Processors (32-/64-bit data register width)
    - Single Core --- A component containing a single processor performing all of the work.
    - Multi-Core or Multi-Processor --- One or more components containing multiple processors each performing their delegated portion of the work.
  - Processor-specific "Host" and optional "Guest" operating systems
    - Host OS --- Primary operating system connected to other computers or terminals to which it provides data or computing services via a hard-wired connection or switched telecommunication network.
    - Guest OS --- Secondary operating system that is either part of a partitioned system or part of a virtual machine (VM) setup.
- Develop, Debug and Test on available Model Platforms

Project Plans:
Adopt Python Virtual Machine (VM) Environment

- A Cross-Platform Environment is created by VMs which are typically:
  - **Implemented** in a platform-independent programming language such as "C/C++"
  - **Compiled** into executable platform-specific VM building block library functions
  - **Executed** upon an operator's shell command (such as "python **DEMO.py** –help")
- VMs typically execute the Python application, upon its launch, via the following process:
  - **Compile** any un-compiled Python application & imported source code into a processor-independent byte-code containing tokens for each standard Python statement or subprogram operation
  - **Interpret** the VM tokens
  - **Execute** VM token-associated executable platform-specific VM building block library functions

Project Plans:
Adopt Python Virtual Machine (VM) Strategy

- Foundation Original Author and Release Adopters
  - Default use of popular Python (2.x.y and 3.x.y) Virtual Machines developed:
    - by the Python Software Foundation (PSF)
    - for popular and readily available Intel processors (x86 & x64) and processor-specific operating systems.
  - Optional use of equivalent Python (2.x.y and 3.x.y) Virtual Machines (or the source code to build them) developed:
    - by the Python Software Foundation
    - for other, less popular, processor types and processor-specific operating systems.

Project Plans:
Adopt Engineering Notebook Strategy

- **Engineering Notebooks (master in repository on development system)**
  - Collection of commentaries that express opinions or offerings of explanations about events or situations that might be useful to installers, developers, operators, troubleshooters and distributors of the toolkit framework.
  - Formats include text, tables and complex graphical images requiring an office suite application programs (such as from “Adobe”, “Microsoft‟, “LibreOffice” etc.) typically found on a general purpose workstation, desktop or laptop computer systems.
- **Project (master in repository on development system; copy in site-package on embedded system)**
  - Excerpts from the Engineering Project Notebook that is in plain text format and suitable for embedded systems with only character-mode terminals.

Project Plans:
Adopt Operator, System Administrator & Field Service Strategy

- **Documents (master in repository on development systems with copies in developer sandboxes; copies also in site-packages on embedded systems)**
  - Typical install, configure, operate and troubleshoot how-to guides with applicable terms and conditions for usage and redistribution. In a plain text format suitable for embedded systems with only character-mode terminals.
- **Manual Pages (master in repository on development systems with copies in developer sandboxes; copies also in site-packages on embedded systems)**
  - Typical on-line information about command line use and application programming for each building block and tool. Each is generated, by a Python source code processing tool, in a plain text format suitable for embedded systems with only character-mode terminals.

Project Implementation:
Adopt Document Focus Strategy

- **System Administrators** and **Field Service** will need to know or learn:
  - Hardware and Software Requirements for System and Applications
  - Available Product & Support Resources
  - How to Install, Configure, Operate and Troubleshoot the Hardware and Software for System and Applications
- **Operators** will need to know or learn:
  - Available System and Application Hardware and Software features and how to use them
- **Developers** will need to know or learn:
  - Hardware and Software Architecture
  - Hardware and Software Interfaces
  - Software Design and Qualification Requirements
  - How to Install, Configure, Operate and Troubleshoot the Hardware and Software for local and remote Systems and Applications

Release (Table of Contents)

Release:
Command Line Interface (CLI) Capabilities

- Launch application programs with or without:
  - operator key word-value pair options
  - positional arguments
- Create platform configuration and event log files for date and time stamped messages notifying operator and field service of situations requiring:
  - immediate action
  - later troubleshooting
- Generate and return Unix-type exit code to coordinate operation of multiple collaborating scripts.

Release:
Graphical User Interface (GUI) Capabilities

- Launch character-mode wxPython-like application programs (mimicking the look and feel of native GUI applications on Microsoft Windows) with or without configurable options for:
  - Splash Screen containing:
    - Trademark, Copyright, License and/or Notice depending on screen size
  - Redirected Output containing:
    - Event log files for date and time stamped messages notifying operator of situations requiring immediate action
  - Taskbar containing:
    - A row of buttons that represent open programs, among which the user can switch back and forth by clicking on the appropriate button
- Generate and return Unix-type exit code to coordinate operation of multiple collaborating scripts.

Release:
Local Monitoring / Control Capabilities

- Once you've logged into your local computer, you can launch one or more local shells (such as "sh" and "bash") associated with the local operating system's command line interface.
- The local operating system's command line interface provides access to associated terminal interface and the TeamSTARS "tsWxGTUI_PyVx" Toolkit's Python and "nCurses" based character-mode user interfaces which enable you to monitor and control one or more local application programs.

Release:
Pre-Alpha Stage Limitations

- Designates a program or application that is not:
  - Feature complete
  - Independently tested
  - Usually be released to the public
- Developers are still deciding on what features the source code and documentation should have and are seeking input from third-parties willing to try it and provide constructive feedback.
  - Current source code and documentation is subject to change without notice.
  - Previous versions of the source code and documentation are archived and remain available in the GitHub repository.

Release (Table of Contents)
Issues

- Engineering & User Issues and associated resolutions are documented within the:
  - Associated “GitHub” Public Repository
    - “./Documents”
      - “./Bugs.txt” file
      - “./Changes.txt” file
    - “./SourceDistributions”
      - “./Developer-Sandboxes” source code file(s)
      - “./Site-Packages” source code file(s)
  - Associated “GitHub” Private Database:
    - Author/Contributor Issues

Release (Table of Contents)
Where to get further information?

https://github.com/rigordo959/tsWxGTUI_PyVx_Repository