

Use Cases



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.



Use Cases

Table of Contents *(with slide show [Hyperlinks](#))*

- [Sample Screen Shots](#)
- [Sample Platform Configurations](#)
- [Command Line Interface \(CLI\)](#)
 - [tsLibCLI](#)
 - [tsToolsCLI](#)
- [Graphical User Interface \(GUI\)](#)
 - [tsLibGUI](#)
- [Remote Monitoring / Control](#)
- [Site-Packages](#)
- [Developer-Sandboxes](#)



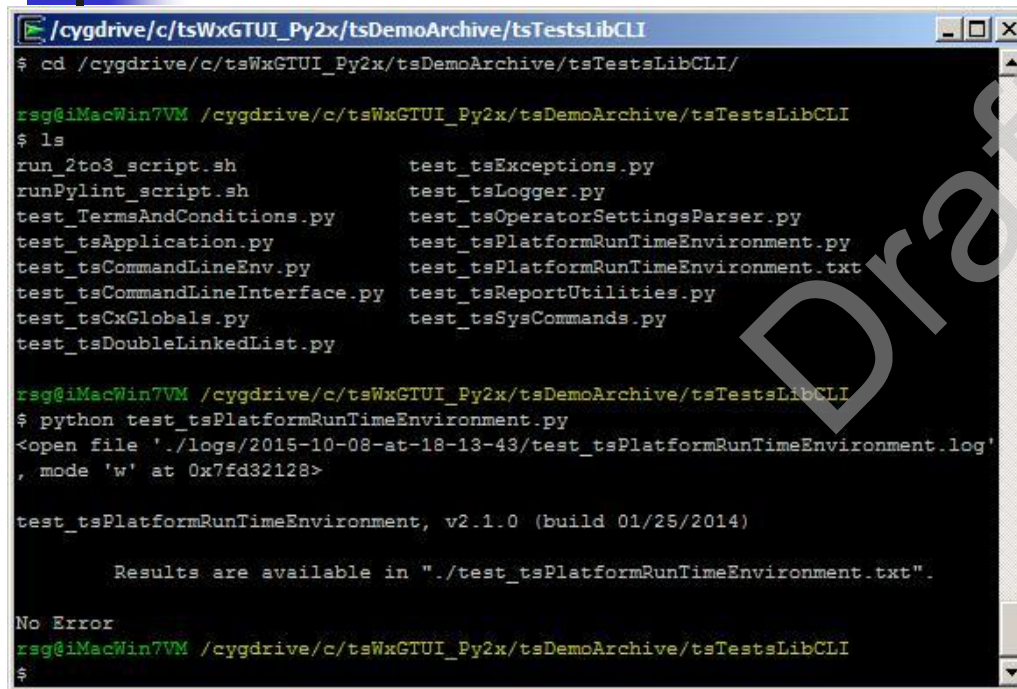
Use Cases: [\(Table of Contents\)](#)

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 \(vt100 Black-on-White\) & \(vt100 White-on-Black\)](#)
- Workstation Desktop for Development & Embedded Systems (HOST)
 - [Sample Mac OS X Desktop with Parallels & VMware Hypervisors & 3 Guest OSs](#)

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)

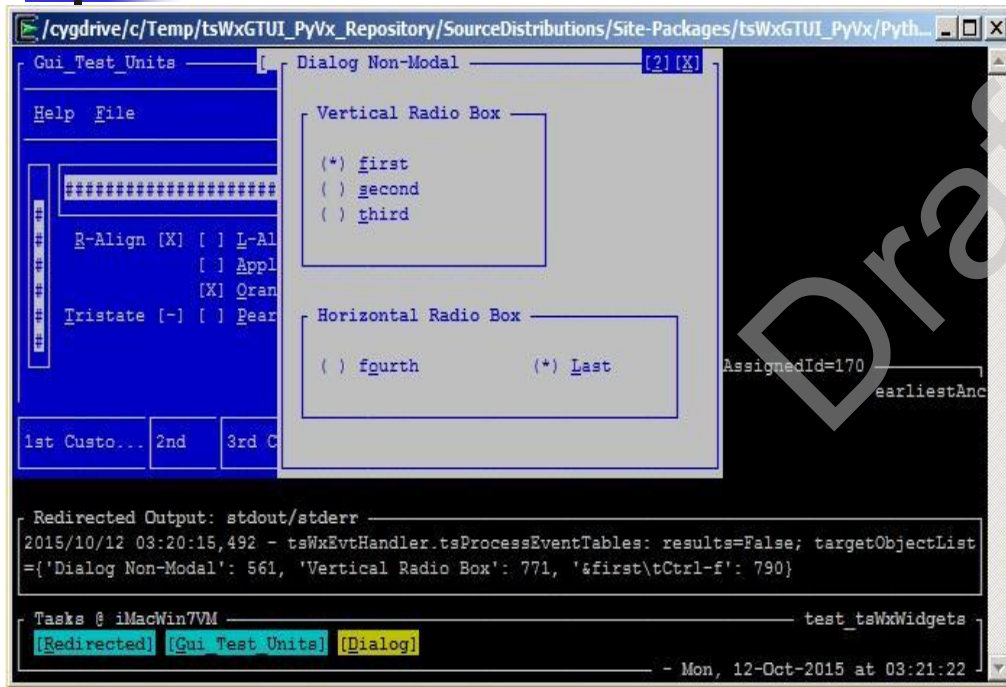
Results are available in './test_tsPlatformRunTimeEnvironment.txt'.

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: [\(Table of Contents\)](#)

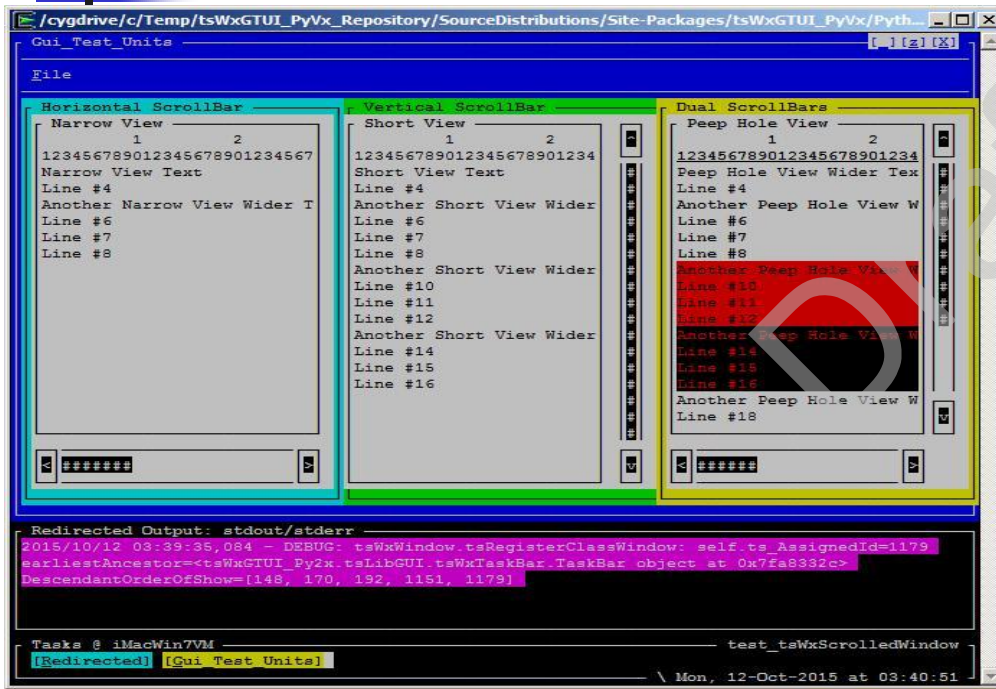
Sample GUI Widgets



- **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)



- **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 colored, 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

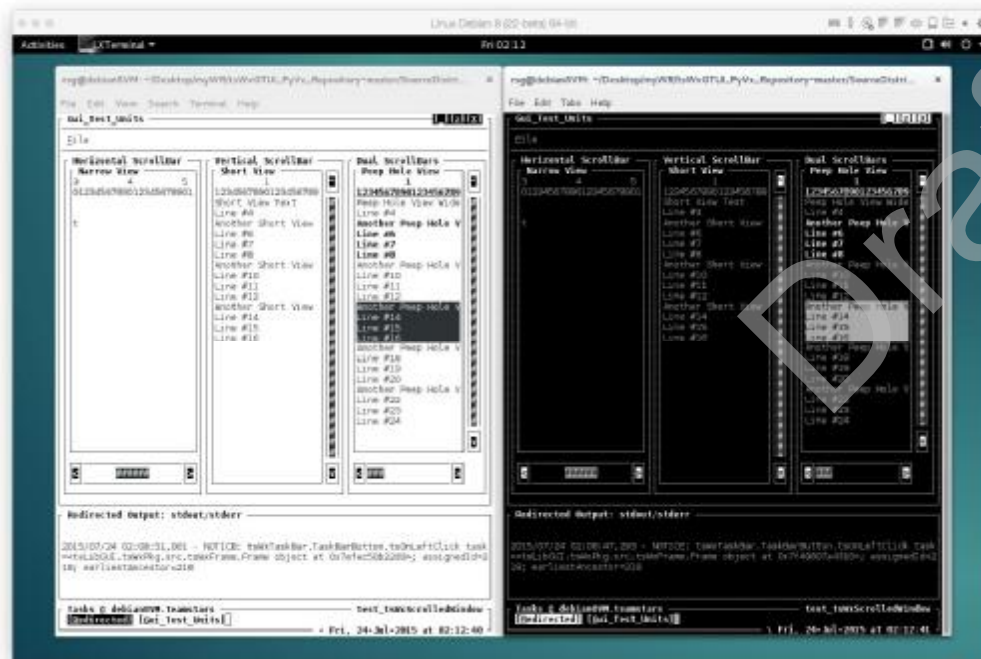
11/7/2015

TeamSTARS "tsWxGTUI_PyVx" Toolkit
prepared & presented by Richard S. Gordon

Use Cases: [\(Table of Contents\)](#)

Sample GUI Scrolled Windows

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



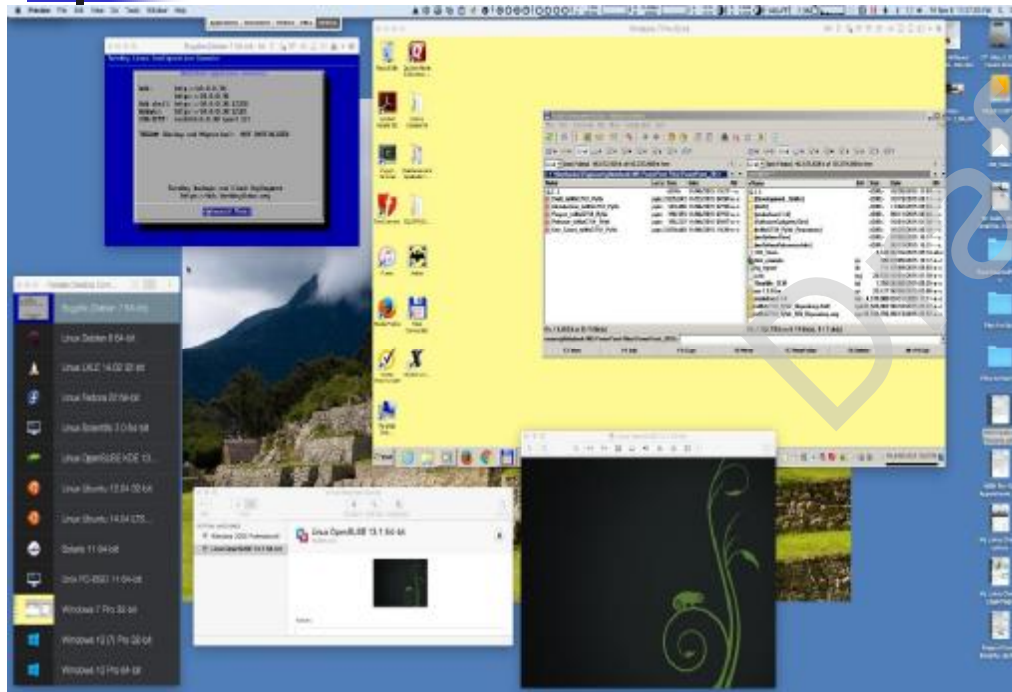
- **Outer-Most Application Frames**
 - With Menu Bar, Window Size & Close Control Buttons and three scrollable panels.
- **Three Scrollable Application Panels (Horizontal, Vertical & Dual)**
 - Each with Multi-Colored & Non-Colored Text, Horizontal and/or Vertical scroll bars, associated clickable arrow buttons and clickable gauge depicting relative size and position or displayed text.
- **Black Redirected Output: stdout/stderr Frame**
 - Output of colored, 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

TeamSTARS "tsWxGTUI_PyVx" Toolkit
prepared & presented by Richard S. Gordon

11/7/2015

Use Cases: [\(Table of Contents\)](#)

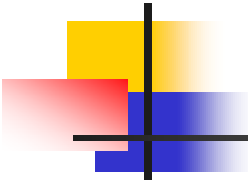
Sample Mac OS X Desktop with: Parallels & VMware Hypervisors & 3 Guest OSs



- Hypervisor #1 (Parallels 11 Guest OS list in bottom left)
 - Bugzilla Database & Apache Server running on Debian Linux (dark blue window in top left)
 - Microsoft Windows 7 (yellow window in top right)
- Hypervisor #2 (VMware Fusion 7 Guest OS list in bottom center)
 - OpenSUSE 13.1 Linux (black window in bottom right)

11/7/2015

TeamSTARS "tsWxGTUI_PyVx" Toolkit
prepared & presented by Richard S. Gordon



Use Cases: [\(Table of Contents\)](#)

Block Diagrams

- [Toolkit Building Block Diagrams](#)
- [Non-Networked \(Stand-Alone\) System \(HW-SW\) Block Diagram](#)
 - [Hardware Component Usage Notes](#)
 - [Operating System Software Component Usage Notes](#)
 - [Application Software Usage Notes](#)
- [Networked \(Stand-Among\) System \(HW-SW\) Block Diagram](#)
 - [Local & Remote System Usage Notes](#)

$$\begin{array}{ccc} \wedge & \wedge & | \\ | & | & | \\ | & | & \vee \end{array}$$

```

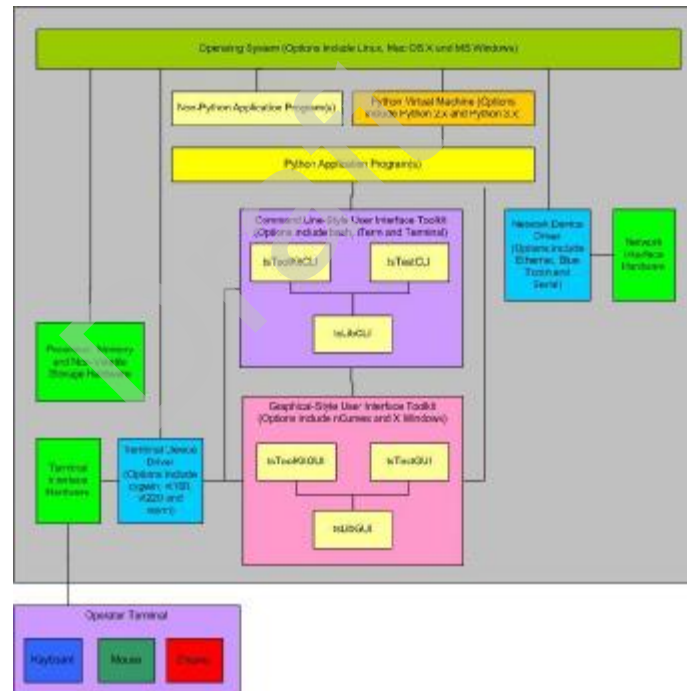
^      ^      |
|      |      +--> Operator Display & Log Files
|      +----- Operator Keyboard
+----- Operator Mouse

```

- | | |
|---|--|
| <p>Building</p> | <p>interface application programs for regression testing and tutorial demos.</p> |
| <p>style user interface building blocks that establishes the high-level, pixel-mode, "wxPython" GUI Toolkit via the User Interface Toolkit.</p> | |
| <p>^ ^ </p> <p> </p> <p> v</p> | |
| <p>and</p> | <p>The "tsTestsCLI" is a set of command line interface application programs and scripts for regression testing and tutorial demos.</p> |
| <p>the building blocks that establishes the POSIX-style, run the files, launching application programs, handling (and event severity annotations) and configuring console</p> | |
| <p>system configuration, diagnostic, installation, presents for various host hardware and software platforms.</p> | |

Use Cases: [\(Table of Contents\)](#)

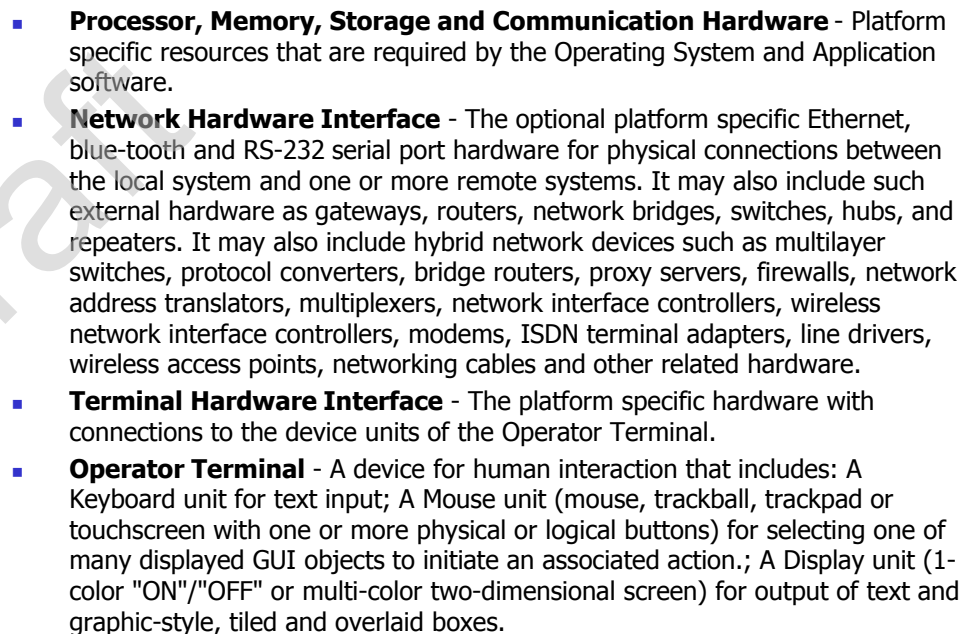
Non-Networked (Stand-Alone) Mode System (HW-SW) Block Diagram



11/7/2015

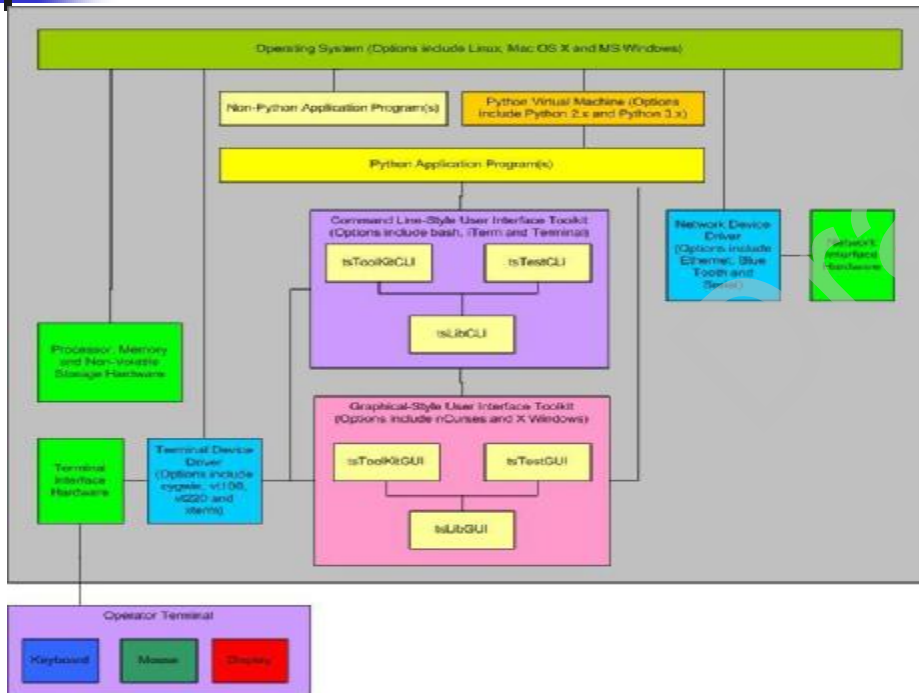
TeamSTARS "tsWxGTUI_PyVx" Toolkit
prepared & presented by Richard S. Gordon

Hardware Component Usage Notes



Use Cases: [\(Table of Contents\)](#)

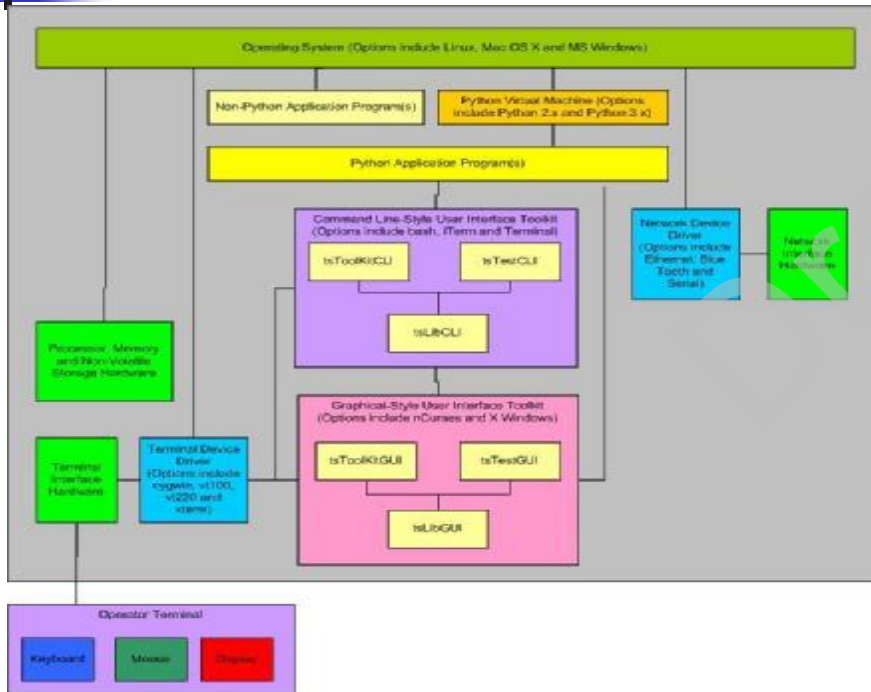
Operating System Software Component Usage Notes



- **Operating System** - The platform specific software (such as Linux, Mac OS X, Microsoft Windows and Unix) that coordinates and manages the time-shared use of a platform's processor, memory, storage and input/output hardware resources by multiple application programs and their associated users/operators.
- **Network Device Driver Interface** - The optional platform specific software whose layered protocol suite (such as TCP/IP) enables the concurrent sharing of the physical connection between the local system and one or more remote systems.
- **Terminal Device Driver** - The platform specific software for transforming data (such as single button scan codes, multi-button flags and pointer position) to and from the platform independent formats (such as upper and lower case text, display screen column and row and displayed colors, fonts and special effects) used by the Command Line Interface and Graphical User Interface software.

Use Cases: [\(Table of Contents\)](#)

Application Software Usage Notes



- **Non-Python Application Program** - The application specific program that performs its service when its pre-compiled, platform specific machine code is executed. Typically, these services are used to analyze, edit, view, copy, move or delete those data and log files which are of interest or no longer needed.
- **Python Application Program** - The application specific program that performs its service when executed by the Python Virtual Machine.
 - **Command Line-Style Interface ("tsLibCLI")** - The platform specific keywords arguments, positional arguments and their associated values and syntax of text used to request services from the Operating System and various Application Programs.
 - **Graphical-Style User Interface ("tsLibGUI")** - The platform specific tiled, overlaid and click-to-select Frames, Dialogs, Pull-down Menus, Buttons, CheckBoxes, Radio Buttons, Scrollbars and associated keywords, values and syntax of text used to request services from the Operating System and various Application Programs.
- **Python Virtual Machine** - The platform specific program that loads, compiles Python language application program source code into platform independent tokenized byte-code and then interprets and executes the byte-code using a processor and operating system specific run time library.

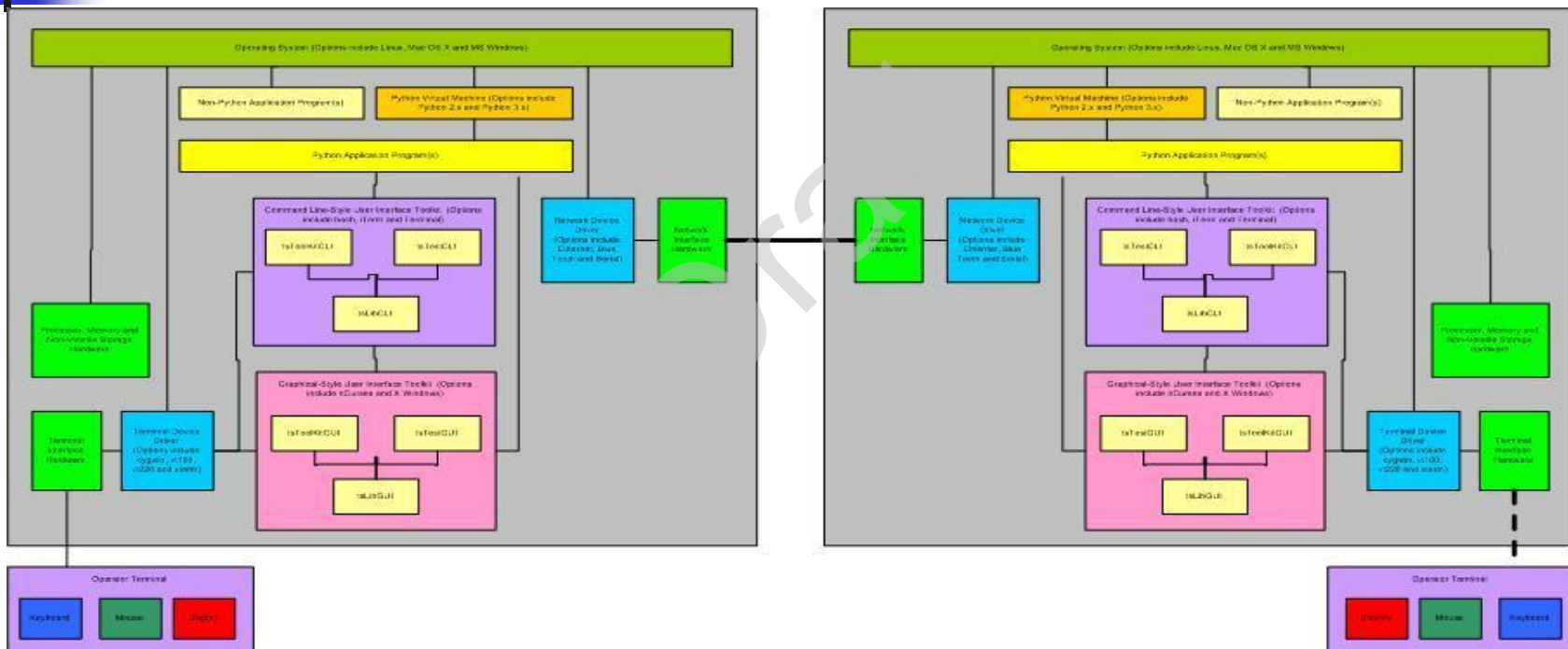
11/7/2015

TeamSTARS "tsWxGTUI_PyVx" Toolkit
prepared & presented by Richard S. Gordon

14

Use Cases: [\(Table of Contents\)](#)

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



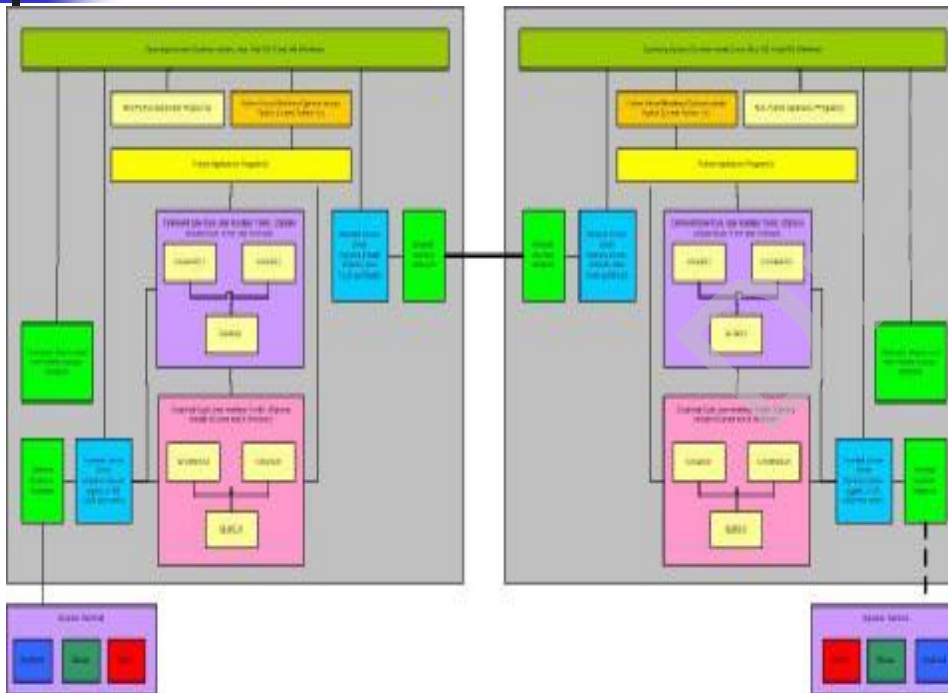
TeamSTARS "tsWxGTUI_PyVx" Toolkit
prepared & presented by Richard S. Gordon

11/7/2015

15

Use Cases: [\(Table of Contents\)](#)

Local & Remote System Usage Notes



- Launch the Local (Left) Command Line Interface shell session (bash).
- Set Local Working Directory
- Login to Remote (Right) System, via SSH (secure shell) or SFTP (secure file transfer protocol), as user with/without administrative privileges.
- Set Remote Working Directory
- Transfer any missing application programs from Local to Remote System via SFTP
- Launch one or more Remote Application(s)
- Create archive of Remote Application logs directory(s)
- Transfer archive(s) of Remote Application logs directory from Remote to Local System via SFTP
- Logout of Remote System
- Logout of Local System

11/7/2015

TeamSTARS "tsWxGTUI_PyVx" Toolkit
prepared & presented by Richard S. Gordon

16



Use Cases: [\(Table of Contents\)](#)

Sample Platform Configurations

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

- Hypervisor Virtual Machines
 - Third-party add-ons for host computers that are used to concurrently run one or more guest operating systems (and their applications).
- 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 concurrently support host, hypervisor and at least one 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 concurrently support host, hypervisor and multiple guest operating systems with both command line and graphical user interfaces.



Use Cases: [\(Sample Platform Configurations\)](#)

Hypervisor Virtual Machines

- A **hypervisor** or **virtual machine monitor (VMM)** is a piece of computer software, firmware or hardware that creates and runs virtual machines.
- A computer on which a hypervisor is running one or more virtual machines is defined as a host machine.
- Each virtual machine is called a guest machine.
- The hypervisor presents the guest operating systems with a virtual operating platform and manages the execution of the guest operating systems.
- Multiple instances of a variety of operating systems may share the virtualized hardware resources.
- **Parallels Desktop for Mac**, by Parallels, and **VMware Fusion**, by VMware are hypervisors that provides hardware virtualization only for Macintosh host computers with Intel x86 or x64 processors that are running Mac OS X. Each virtual machine can execute its own operating system, including versions of Microsoft Windows, Linux, BSD Unix, and MS-DOS.
- **VMware Workstation**, by VMware, is a hypervisor that runs on x64 host computers (an x86 version of earlier releases was available) running Linux or Microsoft Windows. Each virtual machine can execute its own operating system, including versions of Microsoft Windows, Linux, BSD Unix, and MS-DOS.

Use Cases: [\(Sample Platform Configurations\)](#)

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 Configuration**
 - Cygwin 1.7 (includes various GNU, Linux & Python components)
 - Office 2002
 - XEmacs
 - Python 2x & 3x
 - **Ubuntu Linux 12.04 LTS Configuration**
 - GNOME Desktop
 - LibreOffice
 - XEmacs
 - Python 2x & 3x

Use Cases: [\(Sample Platform Configurations\)](#)

Accessorized "Development" Laptop and Guest "Embedded" System

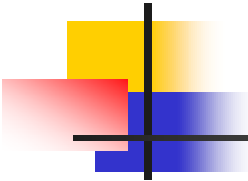
- **2007 Apple MacBook Pro Hardware**
 - 2.33 GHz Intel Core 2 Duo processor
 - 4 GB RAM
 - 17" 1920x1200 pixel LCD display
 - 160 GB (5400 RPM) SATA 1.5 Gb/s internal hard drive
 - 1.5 TB (7200 RPM) SATA 3 Gb/s external hard drive
 - Ethernet Network Adapter
 - WiFi Wireless Network Adapter
- **Development / 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 Guest OS:
 - 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, LibreOffice 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 LibreOffice and Xemacs
 - **VMware Fusion 7** Hypervisor for running Guest OS:
 - Linux (OpenSuSE 13.1)
 - Microsoft Windows (3.1)



Use Cases: [\(Sample Platform Configurations\)](#)

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 Guest OS:
 - Linux (Centos 7, Debian 8, Fedora 22, OpenSuSE 13.2, Scientific 7 & Ubuntu 14.04 LTS & 15.04) with Wing IDE 5, LibreOffice 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 LibreOffice and Xemacs
 - **VMware Fusion** 7 Hypervisor for running Guest OS:
 - 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: [\(Table of Contents\)](#)

tsLibCLI

- This library of building blocks is organized, by the functional scope of each component, into a collection of Python "modules".
- When appropriate, tsLibCLI modules may import and use the services of:
 - any other tsLibCLI module
 - any module listed in the Python Global Module Index



Use Cases: [\(Table of Contents\)](#)

tsApplication.py

- Module enables the application program launched by an operator via a Command Line Interface (CLI) to also initialize, configure and use the same character-mode terminal with a Graphical-style User Interface (GUI).
 - Module registers and validates:
 - operator application settings inputs from command line keyword-value pairs and positional arguments.
 - instantiation settings input from applications via caller parameter list.
- Module is the base class, for **tsCommandLineEnv** which itself is the base class for **tsWxMultiFrameEnv**.



Use Cases: [\(Table of Contents\)](#)

tsCommandLineEnv.py

- Class to initialize and configure the application program launched by an operator.
 - Class delivers those keyword-value pair options and positional arguments specified by the application, in its invocation parameter list.
 - Class wraps the Command Line Interface application with exception handlers to control exit codes and messages that may be used to coordinate other application programs.



Use Cases: [\(Table of Contents\)](#)

tsCommandLineInterface.py

- Class establishes methods that:
 - prompt or re-prompt the operator for input
 - validate that the operator has supplied the expected number of inputs and that each is of the expected type.\
 - does NOT validate the input value



Use Cases: ([Table of Contents](#))

tsCxGlobals.py

- Module to establish configuration constants and macro-type functions for the Command Line Interface mode of the "tsWxGTUI" Toolkit.
 - Module provides a theme-based mechanism for modifying/restoring configuration constants as appropriate for various user roles and activities.



Use Cases: [\(Table of Contents\)](#)

tsDoubleLinkedList.py

- Class to establish a general purpose representation of a linked list with forward and backward pointers
 - Class provides methods to append, insert, delete and access the ordered list of entries.



Use Cases: [\(Table of Contents\)](#)

tsException.py

- Class to define and handle error exception events.
 - Class maps run time exception types into 8-bit exit codes
 - Class prints associated exception diagnostic message and trace back information



Use Cases: [\(Table of Contents\)](#)

tsGistGetTerminalSize.py

- Third-Party Module, derived from "terminalsize.py" by Justin T. Riley:
 - Module acquires the character size of the Python console window as a Python tuple (width, height) on host operating systems (such as Linux, Mac OS X, Microsoft Windows and Unix).



Use Cases: [\(Table of Contents\)](#)

tsLogger.py

- Class emulates a subset of Python logging API.
 - Class defines and handles prioritized, time and date stamped event message formatting and output to files and devices:
 - Log files are organized in a date and time stamped directory named for the launched application.
 - Log devices include the Unix-type syslog, stderr, stdout and stdscr (the Curses display screen).
- Class also supports "wxPython"-style logging of assert and check case results.



Use Cases: [\(Table of Contents\)](#)

tsOperatorSettingsParser.py

- Class to parse the command line entered by the operator of an application program:
 - Platform-independent parsing algorithm extracts and returns the Keyword-Value pair Options and Positional Arguments that will configure and control the application during its execution.
 - Platform-specific parsing algorithm applies the standard Python library module ("argparse", "optparse" or "getopt") appropriate for the active Python version.



Use Cases: [\(Table of Contents\)](#)

tsPlatformRunTimeEnvironment.py

- Class to capture current hardware, software and network information about the run time environment for the user process.
 - Host processor hardware support includes various releases of Arm, x86, PowerPC, SPARC and others.
 - Host operating system software support includes various releases of Cygwin, Linux, Mac OS X, Unix, Windows and others.
 - Host virtual machine software support includes various releases of Java and Python.
 - Network identification support includes host name, aliases and ip-address list.
 - Environment Variable support includes user, session, shell, path and time zone
- It makes this information available via a file (default is `"./PlatformRunTimeEnvironment.txt"`).



Use Cases: [\(Table of Contents\)](#)

tsReportUtility.py

- Class defining methods used to format information for the operator's display and log files:
 - Convert file size from numeric and string format with optional kilo-, mega-, giga-, tera-, peta-, exa-, zeta- and yotta-byte units
 - Convert time between string and seconds formats.
 - Construct a formatted log of multi-level nested Python dictionary contents
 - Construct a string of title and white space to separate one section of text from another.
 - Construct a string of white space appropriate for indenting level.
 - Construct the path to the next log file.
 - Create test summary after elapsed time with statistics details on the
 - Number of (total test runs, passing test run subtotal and failing test run subtotal)
 - Timestamp (startup, shutdown and elapsed)



Use Cases: [\(Table of Contents\)](#)

tsSysCommands.py

- Class definition and methods for:
 - Issuing shell commands to the host operating system
 - Receiving responses from the host operating system
 - Wrapping and using appropriate Python sub-process module methods.



Use Cases: [\(Table of Contents\)](#)

tsToolsCLI

- Developer Tools
 - [tsStripComments](#)
 - [tsStripLineNumbers](#)
 - [tsTreeCopy](#)
 - [tsTreeTrimLines.py](#)
- Troubleshooter Tools
 - [tsPlatformQuery](#)
- Project Tools
 - [tsLinesOfCodeProjectMetrics](#)



Use Cases: [\(Table of Contents\)](#)

tsStripComments

- Python application program, launched via a Command Line Interface (CLI) with options for the operator to designate input and output directories.
 - The application transforms an annotated, development version of a directory containing subdirectories and Python source files into an unannotated copy.
 - The unannotated copy is intended to conserve storage space when installed in an embedded system.
 - The transformation involves stripping comments and “docstrings” by detokenizing a tokenized version of each Python source file.
 - Non-Python files are trimmed of trailing whitespace.



Use Cases: [\(Table of Contents\)](#)

tsStripLineNumbers

- Python application program, launched via a Command Line Interface (CLI) with options for the operator to designate input and output directories.
 - Application strips line numbers from source code (such as from annotated documentation listings) that would not be required as reference points for conditional branching.



Use Cases: [\(Table of Contents\)](#)

tsTreeCopy

- Python application program, launched via a Command Line Interface (CLI) with options for the operator to designate input and output directories.
 - Application copies the files and directories contained in an input source directory to an output target directory.



Use Cases: [\(Table of Contents\)](#)

tsTreeTrimLines.py

- Python application program, launched via a Command Line Interface (CLI) with options for the operator to designate input and output directories.
 - Application copies the files and directories contained in an input source directory to an output target directory after stripping superfluous white space (blanks) from end of each line.



Use Cases: [\(Table of Contents\)](#)

tsPlatformQuery

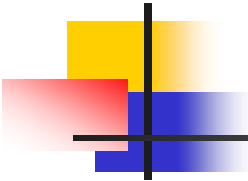
- Python application program, launched via a Command Line Interface (CLI) with options for the operator to designate input and output directories.
 - Application uses [tsPlatformRunTimeEnvironment](#) module to capture and report current hardware and software information about the run time environment available to computer programs.



Use Cases: [\(Table of Contents\)](#)

tsLinesOfCodeProjectMetrics

- Python application program that:
 - Launches via Command Line Interface (CLI) with options designating input directory and output file
 - Scans an operator designated file directory tree containing the source files, in one or more programming language specific formats (such as Ada, Assembler, C/C++, Cobol, Fortran, PL/M, Python, Text, and various command line shells).
 - Accumulates and reports the number of code lines, blank/comment lines, words and characters for individual files, language subtotals and project totals.
 - Generates reports of software project progress and metrics for the software development project (such as labor, cost or contributed value, schedule and lines of code per day productivity).



Use Cases: [\(Table of Contents\)](#)

Graphical User Interface (GUI)

- Output to the user of text string characters 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 trigger a function button, radio button, checkbox or keyboard input operation.
- 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)

- “wxPython” API Mode
 - Application Building Blocks (partial listing)
 - Top-Level GUI Objects
 - Lower-Level GUI Objects
 - GUI Controls
 - GUI Events
 - GUI Sizers
 - Application Configuration
 - Application Diagnostics
 - Application Launchers
- “Curses” API Mode
 - GraphicalTextUserInterface



Use Cases: [\(Table of Contents\)](#)

Application Building Blocks *partial listing* (tsLibGUI)

- Top-Level GUI Objects

- tsWxFrame
- tsWxDialog
- tsWxPyOnDemandOutputWindow (Frame for RedirectedOutput)
- tsWxTaskBar (Frame)

- Lower-Level GUI Objects

- tsWxPanel
- tsWxStatusBar

- GUI Controls

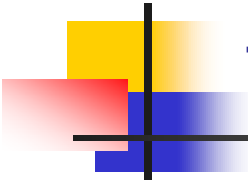
- wxWxButton
- tsWxCheckbox
- tsWxGauge
- tsWxMenuBar
- tsWxRadioButton
- tsWxScrollBar (Buttons & Gauge)
- tsWxTaskBar (Buttons)

- GUI Events

- Keyboard / Mouse / Timer

- GUI Sizers

- tsWxBoxSizer
- tsWxGridSizer



Use Cases: [\(Table of Contents\)](#)

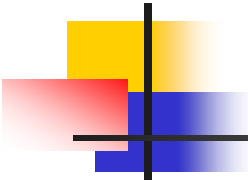
Top-Level GUI Objects (tsLibGUI)

■ **tsWxFrame**

- A window whose size and position can (usually) be changed by the user. It usually has thick borders and a title bar, and can optionally contain a menu bar, toolbar and status bar. A frame can contain any window that is not a Frame or Dialog. It is one of the most fundamental of the wxWindows components.
- A Frame that has a status bar and toolbar created via the CreateStatusBar / CreateToolBar functions manages these windows, and adjusts the value returned by GetClientSize to reflect the remaining size available to application windows.
- By itself, a Frame is not too useful, but with the addition of Panels and other child objects, it user interfaces are constructed.

■ **tsWxDialog**

- A window with a title bar and sometimes a system menu, which can be moved around the screen. It can contain controls and other windows and is often used to allow the user to make some choice or to answer a question.
- Dialogs can be made scrollable, automatically, for computers with low resolution screens.
- Dialogs usually contains either a single button allowing to close the dialog or two buttons, one accepting the changes and the other one discarding them (such button, if present, is automatically activated if the user presses the "Esc" key).



Use Cases: [\(Table of Contents\)](#)

Lower-Level GUI Objects (tsLibGUI)

- **tsWxPanel**

- A window on which controls are placed. It is usually placed within a frame.
- Its main feature over its parent class wxWindow is code for handling child windows and TAB traversal.

- **tsWxStatusBar**

- A narrow window that can be placed along the bottom of a frame to give small amounts of status information.
- It can contain one or more fields, one or more of which can be variable length according to the size of the window.



Use Cases: [\(Table of Contents\)](#)

GUI Controls (tsLibGUI)

- **wxWxButton**
 - A control that contains a text string. It is one of the most common elements of a GUI. It may be placed on a dialog box or panel, or indeed almost any other window.
- **tsWxCheckbox**
 - A labelled box that by default is either on (the checkmark is visible) or off (no checkmark). Optionally (When the wx.CHK_3STATE style flag is set) it can have a third state, called the mixed or undetermined state. Often this is used as a "Does Not Apply" state.
- **tsWxRadioButton**
 - A labelled box which by default is either on (the checkmark is visible) or off (no checkmark). When the operator turns one RadioButton on, all other Radio Buttons within the same RadioBox group are automatically turned off.
- **tsWxMenuBar**
 - A series of menus accessible from the top of a frame.
- **tsWxGauge**
 - A horizontal or vertical bar which shows a quantity (often time). wxGauge supports two working modes: determinate and indeterminate progress. The first is the usual working mode (see SetValue() and SetRange()) while the second can be used when the program is doing some processing but you do not know how much progress is being done. In this case, you can periodically call the Pulse() function to make the progress bar switch to indeterminate mode (graphically it is usually a set of blocks which move or bounce in the bar control).
- **tsWxScrollBar**
 - A control that represents a horizontal or vertical scrollbar. It is distinct from the two scrollbars that some windows provide automatically, but the two types of scrollbar share the way events are received.
- **tsWxTaskBar (buttons)**
 - A narrow window with a taskbar icon (button) for each top level window (frame, dialog etc.). A taskbar icon appears in the "system tray" and responds to mouse clicks, optionally with a tooltip above it to help provide information.



Use Cases: [\(Table of Contents\)](#)

GUI Events (tsLibGUI)

- **Keyboard**

- An event class that contains:
 - key identifier
 - key pressed/released state

- **Mouse**

- An event class that contains:
 - mouse identifier
 - button identifier
 - button state (pressed, released, single-clicked, double-clicked, triple-clicked)
 - current cursor position (display screen character cell row/column coordinates)

- **Timer**

- An event class to allow you to execute code at specified intervals.
- Its precision is platform-dependent, but in general will not be better than 1 millisecond nor worse than 1 second.



Use Cases: [\(Table of Contents\)](#)

GUI Sizers (tsLibGUI)

■ **tsWxBoxSizer**

- The basic idea behind a box sizer is that windows will most often be laid out in rather simple basic geometry, typically in a row or a column or nested hierarchies of either.
- A wx.BoxSizer will lay out its items in a simple row or column, depending on the orientation parameter passed to the constructor.

■ **tsWxGridSizer**

- A grid sizer is a sizer which lays out its children in a two-dimensional table with all cells having the same size. In other words, the width of each cell within the grid is the width of the widest item added to the sizer and the height of each grid cell is the height of the tallest item.
- An optional vertical and/or horizontal gap between items can also be specified (in pixels.)
- Items are placed in the cells of the grid in the order they are added, in row-major order. In other words, the first row is filled first, then the second, and so on until all items have been added. (If necessary, additional rows will be added as items are added.) If you need to have greater control over the cells that items are placed in then use the wx.GridBagSizer.



Use Cases: [\(Table of Contents\)](#)

Application Configuration (tsLibGUI)

■ **tsWxGraphicalTextUserInterface**

- Class uses the Standard Python Curses API to initialize, manage and shutdown input (from a keyboard and mouse) and output (to a two-dimensional display screen).
- Uses built-in or builds monochrome or "wxPython" 68-color palette for operator designated terminal or terminal emulator (vt100-family or xterm-family).
- When appropriate, it substitutes available "Curses" 8-/16-colors for those "wxPython" 68-colors which are NOT available.
- When appropriate, it synthesizes a single click "Curses" xterm-family mouse button input from a sequence of available press-release vt100-family mouse button input.
- Builds platform-specific splash screen from configuration data extracted from txCxGlobals.

■ **tsWxSplashScreen**

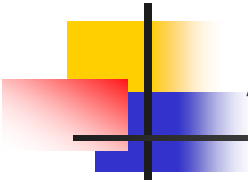
- Class to show a window with a thin border, displaying a "bitmap" (text) describing your application. The splash screen is shown during application initialization. The application then either explicitly destroys it or lets it time-out.
- Became deprecated when tsWxGraphicalTextUserInterface module began building splash screen from configuration data extracted from txCxGlobals.

■ **tsWx**

- Module to load all symbols that should appear within the wxPython wx emulation namespace.
- Includes various classes, constants, functions and methods available for use by applications built with components from the "wxPython" emulation infrastructure.
- The tsWx module is intended to be imported by each Toolkit user applications and Toolkit Test.

■ **tsWxGlobals**

- The tsWxGTUI_PyVx Toolkit emulates the definitions of and references to constant and variable names used by the C++ based "wxWidgets" API and the wxPython interface wrapper used by Python programmers.
- The tsWxGlobals module and emulated tsWxPython classes and methods are then imported by the tsWx module to emulate the "wxPython" wx module.



Use Cases: [\(Table of Contents\)](#)

Application Diagnostics (tsLibGUI)

■ **tsWxGraphicalTextUserInterface**

- Logs the following platform configuration information:
 - "Curses" GUI platform configuration constants
 - Keyboard, Mouse, Display
 - "Curses" GUI operator-designated run time configuration
 - Terminal Name / Type
 - Terminal Has Colors
 - Number of Colors
 - Number of Color-Pairs
 - Built-in Color Palette
 - Generated "wxPython"-style Color Palette
 - "wxPython"-style GUI Object configuration (including parent-child relationship(s), type, position, size, style, color-pair, title, label, border and curses GUI object handle etc.)
- Logs the following run time event notification information:
 - Equipment Operator event notifications
 - Maintenance / Field Service event notifications
 - Developer Debug event notifications

■ **tsWxLog (Future)**



Use Cases: [\(Table of Contents\)](#)

Application Launchers (tsLibGUI)

- **tsWxPyApp**

- PyApp class, based on tsWxEvtHandler, to represent the application and is used to:
 - Bootstrap the "wxPython"-style system and initialize the underlying GUI toolkit
 - Set and get application-wide properties
 - Implement the windowing system main message or event loop, and to dispatch events to window instances.

- **tsWxApp**

- App class, based on tsWxPyApp, to represent the application and is used to do the same things.

- **tsWxPySimpleApp**

- PySimple is a simple application class, based on tsWxApp.
- You can just create one of these and then make your top level windows later, and not have to worry about OnInit.

- **tsWxPyOnDemandOutputWindow**

- PyOnDemandOutputWindow class that can be used for redirecting Python stdout and stderr streams.
- It will do nothing until something is written to the stream at which point it will create a Frame with a text area and write the text there.

- **tsWxMultiFrameEnv**

- MultiFrameEnv class to enable an application using a Command Line Interface (CLI) to launch and use the same character-mode terminal with a Graphical-style User Interface (GUI).
- It uses application specified configuration keyword-value pair options to initialize any application specific logger(s)
- It wraps the CLI, underlying the GUI, and the GUI with exception handlers to control the exit codes and messages used to coordinate other application programs.



Use Cases: [\(Table of Contents\)](#)

GraphicalTextUserInterface *partial listing* (tsLibGUI)

- Curses Supervisor
 - tsWxGraphicalTextUserInterface.py
 - tsWxMultiFrameEnv.py
 - tsWxScrollBarButton.py
 - tsWxScrollBarGauge.py
 - tsWxSplashScreen.py
 - tsWxTaskBar.py
- Curses Event Handler
 - tsWxEvtHandler.py
- Curses Keyboard
 - tsWxCursesKeyCodesDataBase.py
- Curses Mouse
 - tsWxCursesMouseButtonCodesDataBase.py
- Curses Utilities
 - tsWxPythonPrivateLogger.py
 - tsWxCursesServices.py
 - tsWxWindowCurses.py
- Curses Configuration
 - tsWx
 - tsWxGlobals.py
- Curses Display
 - tsWxDisplay.py
 - tsWxScreen.py
 - tsWxColorDatabase.py
 - tsWxPythonColor16DataBase.py
 - tsWxPythonColor16SubstitutionMap.py
 - tsWxPythonColor256DataBase.py
 - tsWxPythonColor88DataBase.py
 - tsWxPythonColor8DataBase.py
 - tsWxPythonColor8SubstitutionMap.py
 - tsWxPythonColorDataBaseRGB.py
 - tsWxPythonColorNames.py
 - tsWxPythonColorRGBNames.py
 - tsWxPythonColorRGBValues.py
 - tsWxPythonMonochromeDataBase.py



Use Cases: [\(Table of Contents\)](#)

Curses Supervisor (tsLibGUI)

- tsWxGraphicalTextUserInterface.py
- tsWxMultiFrameEnv.py
- tsWxScrollBarButton.py
- tsWxScrollBarGauge.py
- tsWxSplashScreen.py
- tsWxTaskBar.py

Draft



Use Cases: [\(Table of Contents\)](#)

Curses Event Handler (tsLibGUI)

- tsWxEvtHandler.py

Draft



Use Cases: [\(Table of Contents\)](#)

Curses Keyboard (tsLibGUI)

- tsWxCursesKeyCodesDataBase.py

Draft



Use Cases: [\(Table of Contents\)](#)

Curses Mouse (tsLibGUI)

- `tsWxCursesMouseButtonCodesDataBase.py`

Draft

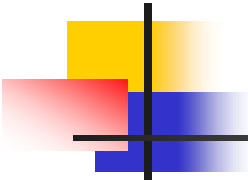


Use Cases: [\(Table of Contents\)](#)

Curses Utilities (tsLibGUI)

- tsWxPythonPrivateLogger.py
- tsWxCursesServices.py
- tsWxWindowCurses.py

Draft



Use Cases: [\(Table of Contents\)](#)

Curses Configuration (tsLibGUI)

- tsWx
- tsWxGlobals.py

Draft



Use Cases: [\(Table of Contents\)](#)

Curses Display (tsLibGUI)

- tsWxDisplay.py
- tsWxScreen.py
- tsWxColorDatabase.py
- tsWxPythonColor16DataBase.py
- tsWxPythonColor16SubstitutionMap.py
- tsWxPythonColor256DataBase.py
- tsWxPythonColor88DataBase.py
- tsWxPythonColor8DataBase.py
- tsWxPythonColor8SubstitutionMap.py
- tsWxPythonColorDataBaseRGB.py
- tsWxPythonColorNames.py
- tsWxPythonColorRGBNames.py
- tsWxPythonColorRGBValues.py
- tsWxPythonMonochromeDataBase.py

11/7/2015

TeamSTARS "tsWxGTUI_PyVx" Toolkit
prepared & presented by Richard S. Gordon



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.



Use Cases: [\(Table of Contents\)](#)

Site-Packages

- Site-Packages are the location where third-party packages are installed (i.e., those not part of the core Python distribution).
 - A Site-Package can only be used with those Python versions for which it has been explicitly installed.
 - For Linux, Mac OS X and Unix operating systems, one must have “root” or administrator privileges to write to the install location.
- To facilitate application software development and deployment, Site-Packages:
 - Organize source code into a single layer set of subdirectories within the Site-Package directory.
 - Import modules, via static Site-Package directory relative path specifications.
- **Default Python Example:** A Site-Package, installed for use with the platform’s default Python must be installed via
 - “python setup.py install”
- **Optional Python 2x Example:** A Site-Package, installed for use with the platform’s add-on Python 2.7.10 must be installed via
 - “python2.7.10 setup.py install”
- **Optional Python 3x Example:** A Site-Package, installed for use with the platform’s add-on Python 3.5.0 must be installed via
 - “python3.5.0 setup.py install”



Use Cases: [\(Table of Contents\)](#)

Developer-Sandboxes

- A Developer-Sandbox is a testing environment that isolates untested code changes and outright experimentation from the production (Site-Package) environment or repository.
 - There should be one Developer-Sandbox for any or all Python 2x (second generation language) releases.
 - There should be one Developer-Sandbox for any or all Python 3x (third generation language) releases.
- To facilitate software development and troubleshooting, Developer-Sandboxes:
 - Organize source code into a set of multi-level nested directories within the Developer-Sandbox directory.
 - Import modules, within try-except blocks, via dynamic multi-level nested path specifications.