



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

- Introduction
- Project
- Platforms
- Release

Draft

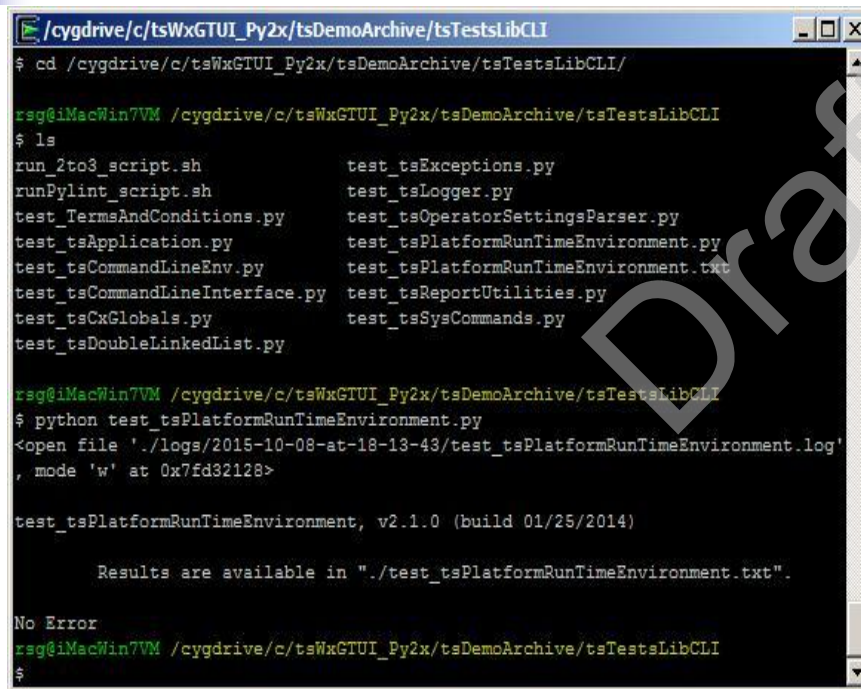


Introduction

- Sample Screen Shots
 - Command Line Interface (CLI)
 - [Sample CLI Display](#)
 - Graphical User Interface (GUI)
 - [Sample GUI Widgets](#)
 - [Sample GUI Scrolled Windows](#)
- Block Diagrams
 - [Toolkit Building Block Diagrams](#)
 - [Non-Networked \(Stand-Alone\) System \(HW-SW\) Block Diagram](#)
 - [Networked \(Stand-Among\) System \(HW-SW\) Block Diagram](#)

Introduction:

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_tsCxxGlobals.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)

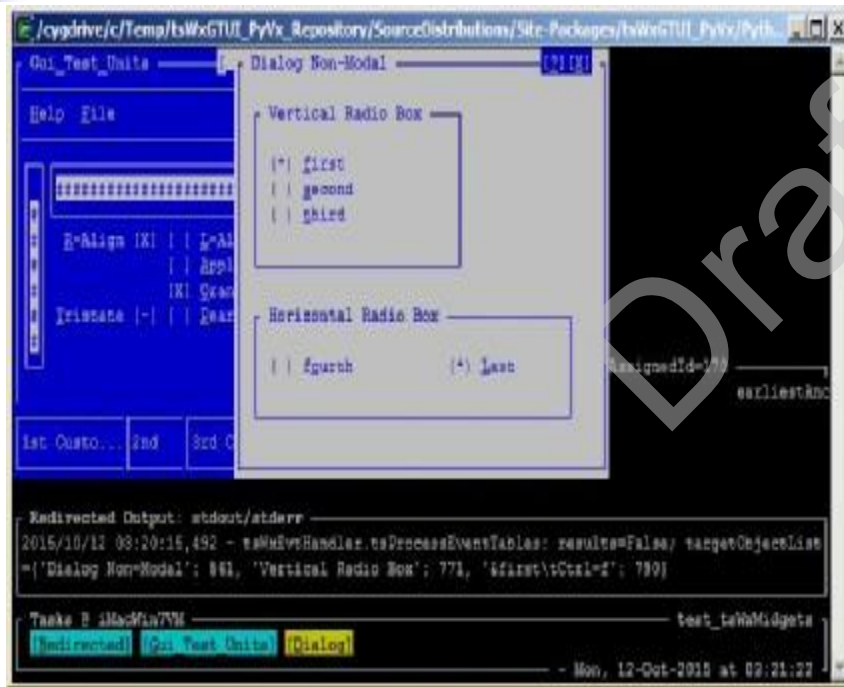
    Results are available in "./test_tsPlatformRunTimeEnvironment.txt".

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.

Introduction:

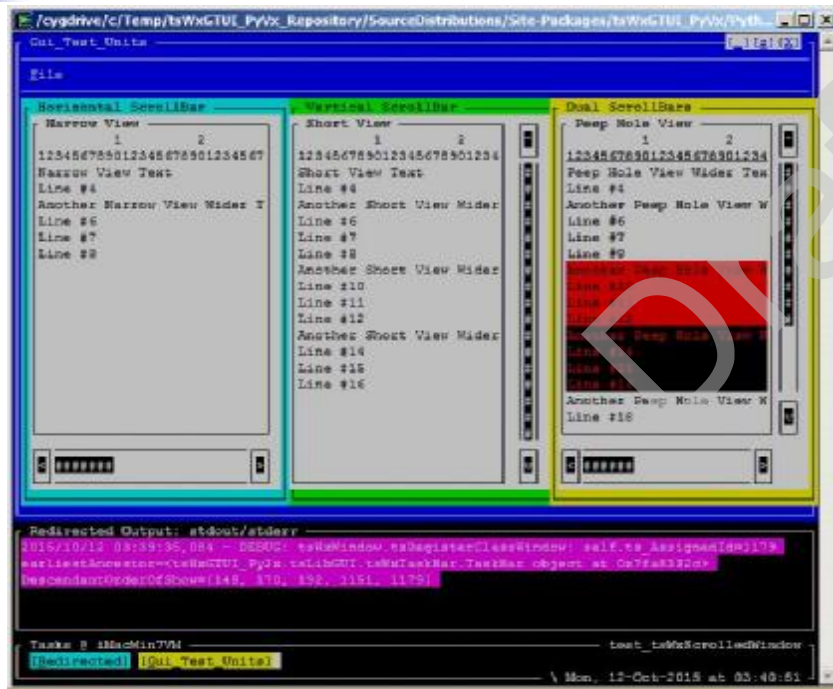
Sample GUI Widgets



- **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

Introduction:

Sample GUI Scrolled Windows



- **Blue Frame**
 - With Menu Bar, Window Size & Close Control Buttons and three scrollable panels.
- **Blue, Green & Yellow Scrollable Panels**
 - 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

TeamSTARS "tsWxGTUI_PyVx" Toolkit
presented by Richard S. Gordon

10/15/2015

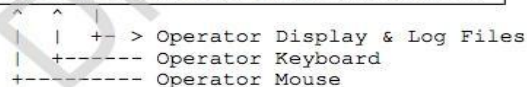
Introduction:

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.	



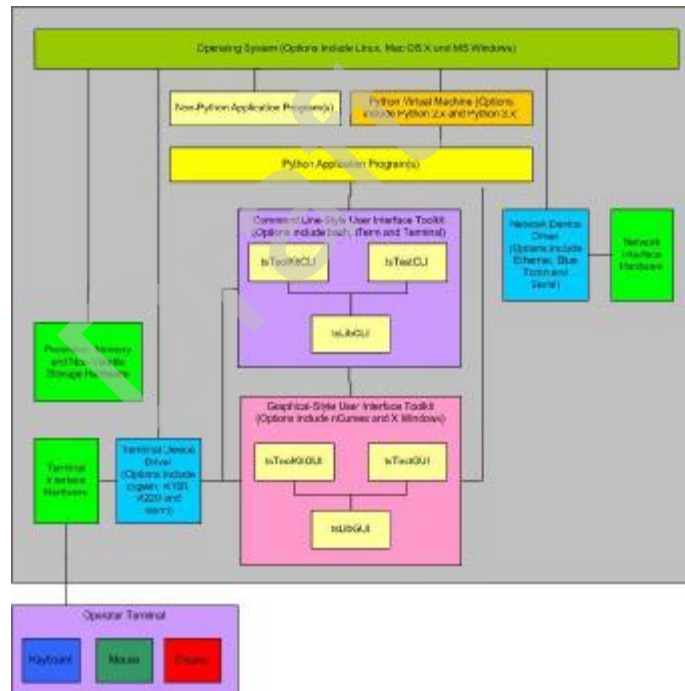
Command Line Interface (tsToolkitCLI)	
The "tsToolsCLI" is a set of command line interface application programs and utility scripts for: checking source code syntax and style via "plint"; generating Unix-style "man page" documentation from source code comments via "pydoc"; and installing, modifying for 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 performance tool components for various host hardware and software platforms.	



- 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

Introduction:

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

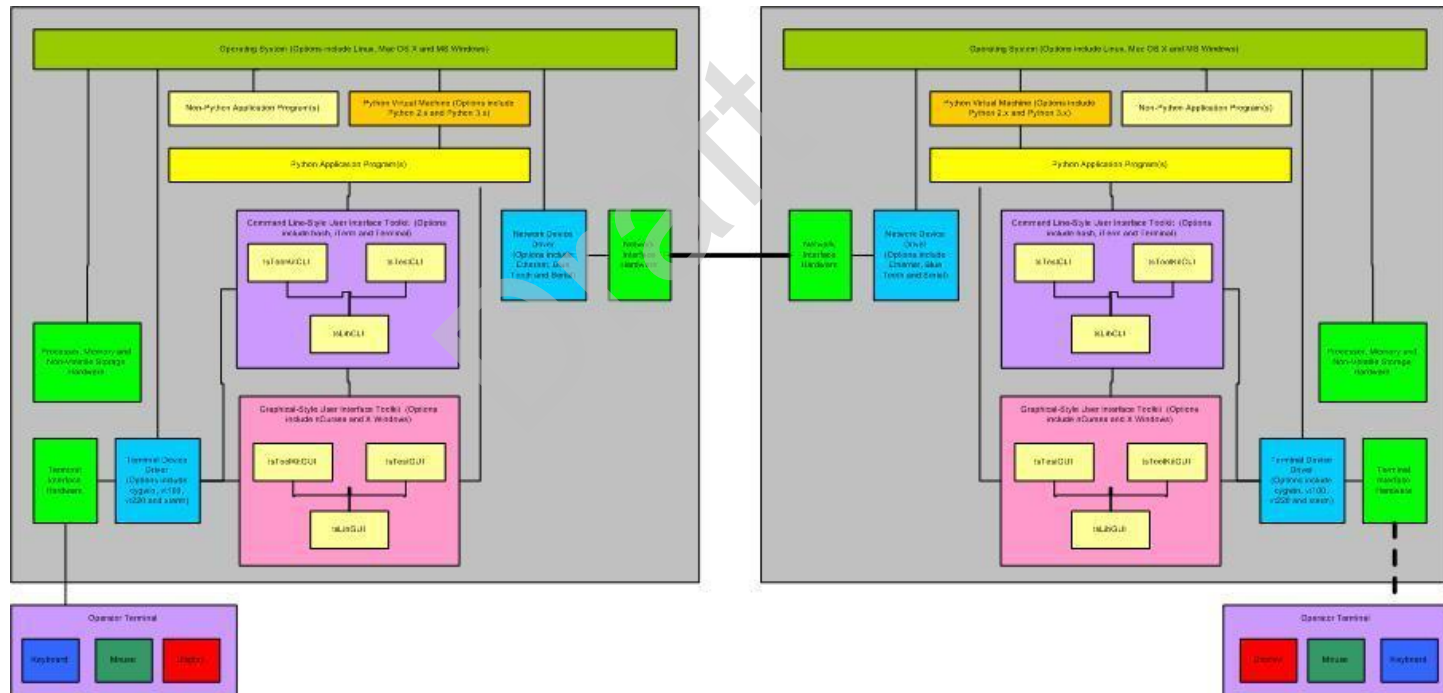


10/15/2015

TeamSTARS "tsWxGTUI_PyVx" Toolkit
presented by Richard S. Gordon

Introduction:

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



TeamSTARS "tsWxGTUI_PyVx" Toolkit
presented by Richard S. Gordon

10/15/2015



Project

- **Objectives**

- Goals (Capabilities)
- Non-Goals (Limitations)

- **Plan**

- 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 to facilitate 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)



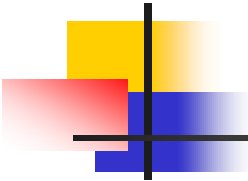
Project Objectives: Goals (Architecture Capabilities)

- 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 "**3to4**"
 - **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 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 (Cross-Platform Limitations)

- Foundation will NOT provide "*Magical*" cross-platforms which:
 - **Run Host Platform Native Applications (Processor & Operating System Specific):**
 - 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).
 - **Run Pixel-mode "wxPython" (wrapper for Python programmers to "wxWidgets" implemented in C++) GUI Applications**
 - You should **NOT** expect to be able to run 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 applications designed to dynamically construct and output to the display an array or sequence of alpha-numeric, punctuation and Curses-standard line-drawing characters.



Project Objectives: Non-Goals (Retrofit Limitations)

- Foundation will **NOT** provide "*Magical*" 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, rebuild or simulate such platforms.



Project Plan: Adopt Cross-Platform Technology

- Apply Hardware & Software Technology that is:
 - 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 Plan:

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 Plan: Adopt Cross-Platform Software Environment

- **POSIX**, a Unix-like operating system complying with the Portable Operating System Interface:
 - **Apple Mac OS X** (Darwin- and 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 2x** (mature 2nd generation language)
 - **Python 3x** (evolving 3rd generation language)



Project Plan:

Adopt Python 2x Source Code Plan

- Develop software in mature 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 imports with explicit ones referencing site-package identifier



Project Plan:

Adopt Python 3x Source Code Plan

- Develop software in evolving 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 imports with explicit ones referencing site-package identifier



Project Plan:

Adopt Debug and Test Environment

- Debug and test on available hardware and software platforms:
 - Accessorized "Development" Desktop and Guest "Embedded" System
 - Accessorized "Development" Laptop and Guest "Embedded" System
 - Basic "Development" Laptop and Pseudo "Embedded" System
 - Representative 32-/64-bit processors (Intel x86 or x64)
- Debug and test on available hardware and software platforms:
 - Representative 32-/64-bit host and/or guest operating systems:
 - GNU/Linux --- Debian-based (Debian, LXLE & Ubuntu) & RPM-based (Fedora, CentOS, OpenSUSE, Red Hat & Scientific)
 - Mac OS X --- Darwin & BSD Unix-based (Lion / 10.7 – El Capitan / 10.11)
 - Microsoft Windows --- NT-based (XP, 7, 8, 8.1 & 10) with Cygwin, the free GNU/Linux-like Plug-in from Red Hat
 - Unix --- AT&T/BSD-based (OpenIndiana 151a8, OpenSolaris 11, FreeBSD 10 & PCBSD 11)

10/15/2015

TeamSTARS "tsWxGTUI_PyVx" Toolkit
presented by Richard S. Gordon



Project Plan:

Adopt Python Virtual Machine (VM) Environment

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

Project Plan:

Adopt Python Virtual Machine (VM) Strategy

- Foundation Original Author and Release Adopters
 - Default use 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.
 - Optionally use 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 processor types and processor-specific operating systems.



Project Plan:

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 Toolkit installers, developers, operators, troubleshooters and distributors. 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 office desktop computer system.
- **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 Plan:

Adopt Operator, System Administrator & Field Service Strategy

- **Documents (master in repository on development system; copy in site-package on embedded system)**
 - 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 system; copy in site-package on embedded system)**
 - 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 Documentation Audience Strategy

- **New Users** seeking reason to dig deeper or just pass it over
- **Student Users** seeking strategy explanation for computer hardware and software technology usage
- **Intermediate Users** seeking strategy explanation for and useful "Python", "Curses" and "wxPython" programming techniques
- **Advance Users** seeking strategy explanation for project planning and engineering techniques
- **Expert Users** seeking strategy explanation for and useful troubleshooting, maintenance, porting and enhancement techniques



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



Project Release

- Capabilities
 - Command Line Interface (CLI)
 - Graphical User Interface (GUI)
- Limitations
- Issues

Draft



Project

Release: Command Line Interface 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.



Project

Release: Graphical User Interface 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.



Project

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.



Project

Release: Limitations

- Pre-Alpha stage development version:
 - Designates a program or application that is not feature complete, not independently tested and would not usually be released to the public.
 - Developers are still deciding on what features the program should have and are seeking input from third-parties willing to try it and provide constructive feedback.



Project

Release: System Technology Requirements

- Hardware and Software must be:
 - Popular and readily available
 - Suitable for cross-platform rapid prototyping
 - Field proven with a long term track record of support
 - Third-party software must be open source to be re-distributable
 - Third-party documentation must be re-distributable
 - Available or within the Toolkit author's procurement means



Project

Release: System Hardware Requirements

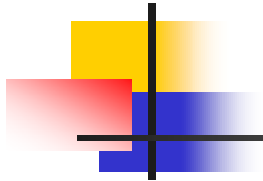
- **Processor** (single or multiple core 32-/64-bit)
- **Random Access Memory** (for transient data and program storage during execution)
- **Non-volatile Memory** (hard disk drive or solid state (flash memory) drive for data and program storage during power outages)
- **Input/Output** ports (plug-in connections for optional peripheral equipment)
- **Network Communication** (Local and/or Wide Area Network using Ethernet and/or WiFi)
- **Keyboard** (a typewriter-like panel of alpha-numeric, punctuation, control and function keys)
- **Display** (a television-like screen used to display the output of a computer to the user)
 - Development System
 - Pixel-mode with 640 x 480 pixel resolution or higher
 - Optional character-mode with at least 80 columns/line and 40 lines/page
 - Embedded System
 - Character-mode with at least 35 columns/line and 16 lines/page
 - Optional pixel-mode with 280 x 200 pixel resolution or higher
- **Mouse** (a hand-operated device that controls the coordinates of a cursor on the display as it is moved)
- **Printer** (an optional device which allows a user to print items on paper)



Project

Release: System Software Requirements

- A multi-user (for local and remote access), multi-process (for interacting with multiple applications) and multi-threaded (for sharing platform resources) 32-/64-bit operating system such as:
 - Apple Mac OS X on Apple-IBM-Motorola alliance (PowerPC) and Intel (x86 and x64) architectures
 - GNU/Linux on Intel (x86 and x64) and other architectures
 - Microsoft Windows on Intel (x86 and x64) architectures with "Cygwin", the free GNU/Linux-like plug-in from Red Hat.
 - Unix on Apple-IBM-Motorola alliance (PowerPC), HP-UX (PA-RISC), IBM-AIX (RS/6000), Intel (x86 and x64), IRIX (SGI/MIPS), Solaris (Sun/SPARC) and other architectures
- Curses-like Terminal Control Library and terminal emulators
 - xterm, xterm-16color, vt100, vt220
- Python 2x and/or Python 3x Interpreter and Virtual Machine



tsWxGTUI_PyVx

PLATFORMS

Draft

10/15/2015

TeamSTARS "tsWxGTUI_PyVx" Toolkit
presented by Richard S. Gordon

39



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, 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 GuestOS:
 - Linux (OpenSuSE 13.1)
 - Microsoft Windows (2000)



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, 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 5** Hypervisor for running GuestOS:
 - Linux (OpenSuSE 13.1)
 - Microsoft Windows (3.1)



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
 - LibreOffice
 - Xemacs
 - Python 2x & 3x



Development Systems

- General purpose commercial systems (laptop, desktop, mainframe or super computer) with suitable and/or readily upgradeable components:
 - Processing, memory, communication, input/output.
 - File storage (hard disk drive or solid state drive) resources.
 - A keyboard and mouse (trackball, touchpad or touchscreen)
 - A 12" to 60+" pixel-mode multi-font graphics display that supports at least 80 col x 40 row (640 x 480 pixels) and 16,777,216 colors.
 - Industry standard terminal emulators.
- Usage:
 - Installing, configuring, maintaining, troubleshooting and running the operating system and application software.
 - Viewing system messages, status, manual pages and log files.



Embedded Systems

- Computer systems customized and optimized with suitably limited but not readily upgradeable application-specific components:
 - Processing, memory, communication, input/output.
 - File storage (hard disk drive or solid state drive) resources.
 - A keyboard and mouse (trackball, touchpad or touchscreen)
 - A character-mode single font text display (whose size could be somewhere between that of a 3" smart phone and a 12" tablet) that supports the application (or its splash screen).
 - Industry standard terminal emulators.
- Usage:
 - Installing, configuring, maintaining, troubleshooting and running the operating system and application software;
 - Viewing system messages, status, manual pages and log files.



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.



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



CLI Tools (tsToolsCLI) 1 of 2

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



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.



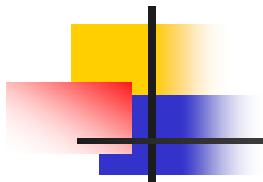
Graphical User Interface (GUI)

- 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.



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 / Timer
 - GUI Sizers
 - tsWxBoxSizer
 - tsWxGridSizer
- Application Configuration
 - tsWxGlobals
 - tsWxGraphicalTextUserInterface
 - tsWxSplashScreen
- Application Diagnostics
 - tsWxLog (Future)
- Application Launchers
 - tsWxApp
 - tsWxPyApp
 - tsWxPyOnDemandOutputWindow
 - tsWxPySimpleApp
 - tsWxMultiFrameEnv



tsWxGTUI_PyVx

RELEASE

Draft

10/15/2015

TeamSTARS "tsWxGTUI_PyVx" Toolkit
presented by Richard S. Gordon

51



Issues

Draft



Application Programs

- Applications for automation, communication, control, diagnostic, instrumentation and simulation.
- Applications typically require an "operator-friendly" Command Line Interface (CLI) or a Graphical-style User Interface (GUI) that can be controlled locally and/or remotely.



Capabilities: 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.



Limitations: Not Magical Cross-platform

- Host Specific Native Applications
 - You will NOT be able to run native Linux, Mac OS X, Microsoft Windows and Unix applications on each other's development and embedded system platforms.
- Pixel-mode "wxPython" / "wxWidgets" Applications
 - You will NOT be able to run pixel-mode applications on platforms with character-mode terminals or terminal emulators (such as vt100, vt220, Cygwin, linux, xterm, xterm-color, xterm-16color, xterm-88color and xterm-256color).