

Draft

Draft

# 1 SCREENSHOTS

---

From Wikipedia, the free encyclopedia

"A screen dump, screen capture (or screen-cap), screenshot (or screen shot), screengrab (or screen grab), or print screen[1] is an image taken by the computer user to record the visible items displayed on the monitor, television, or another visual output device. Usually, this is a digital image using the (host) operating system or software running on the computer, but it can also be a capture made by a camera or a device intercepting the video output of the display (such as a DVR). That latent image converted and saved to an image file such as to JPEG or PNG format is also called a screenshot.

Screenshots can be used to demonstrate a program, a particular problem a user might be having, or generally when display output needs to be shown to others or archived. For example, after being emailed a screenshot, a Web page author might be surprised to see how his page looks on a different Web browser and can take corrective action. Likewise with differing email software programs, (particularly such as in a cell phone, tablet, etc.,) a sender might have no idea how his email looks to others until he sees a screenshot from another computer and can (hopefully) tweak his settings appropriately."

---

- ***XTERM with 8-Color / 64-Color Pairs*** (on page 4)
- ***VT-100 with 1-Color / 2-Color Pairs*** (on page 15)

---

## 1.1 XTERM with 8-Color / 64-Color Pairs

From Wikipedia, the free encyclopedia:

"In computing, xterm is the standard terminal emulator for the X Window System. A user can have many different invocations of xterm running at once on the same display, each of which provides independent input/output for the process running in it (normally the process is a Unix shell).

xterm originated prior to the X Window System. It was originally written as a stand-alone terminal emulator for the VAXStation 100 (VS100) by Mark Vandevoorde, a student of Jim Gettys, in the summer of 1984, when work on X started. It rapidly became clear that it would be more useful as part of X than as a standalone program, so it was retargeted to X. As Gettys tells the story, "part of why xterm's internals are so horrifying is that it was originally intended that a single process be able to drive multiple VS100 displays."

After many years as part of the X reference implementation, around 1996 the main line of development then shifted to XFree86 (which itself forked from X11R6.3), and it is presently actively maintained by Thomas Dickey.

Xterm variants are also available.

Most terminal emulators for X started as variations on xterm."

Typically, xterms support keyboard and mouse input and a display whose character cells consist of an array of RED-GREEN-BLUE phosphors per pixel that are independently programmable in intensity so as to produce any of 8-colors (with their associated 64-color pairs) per character:

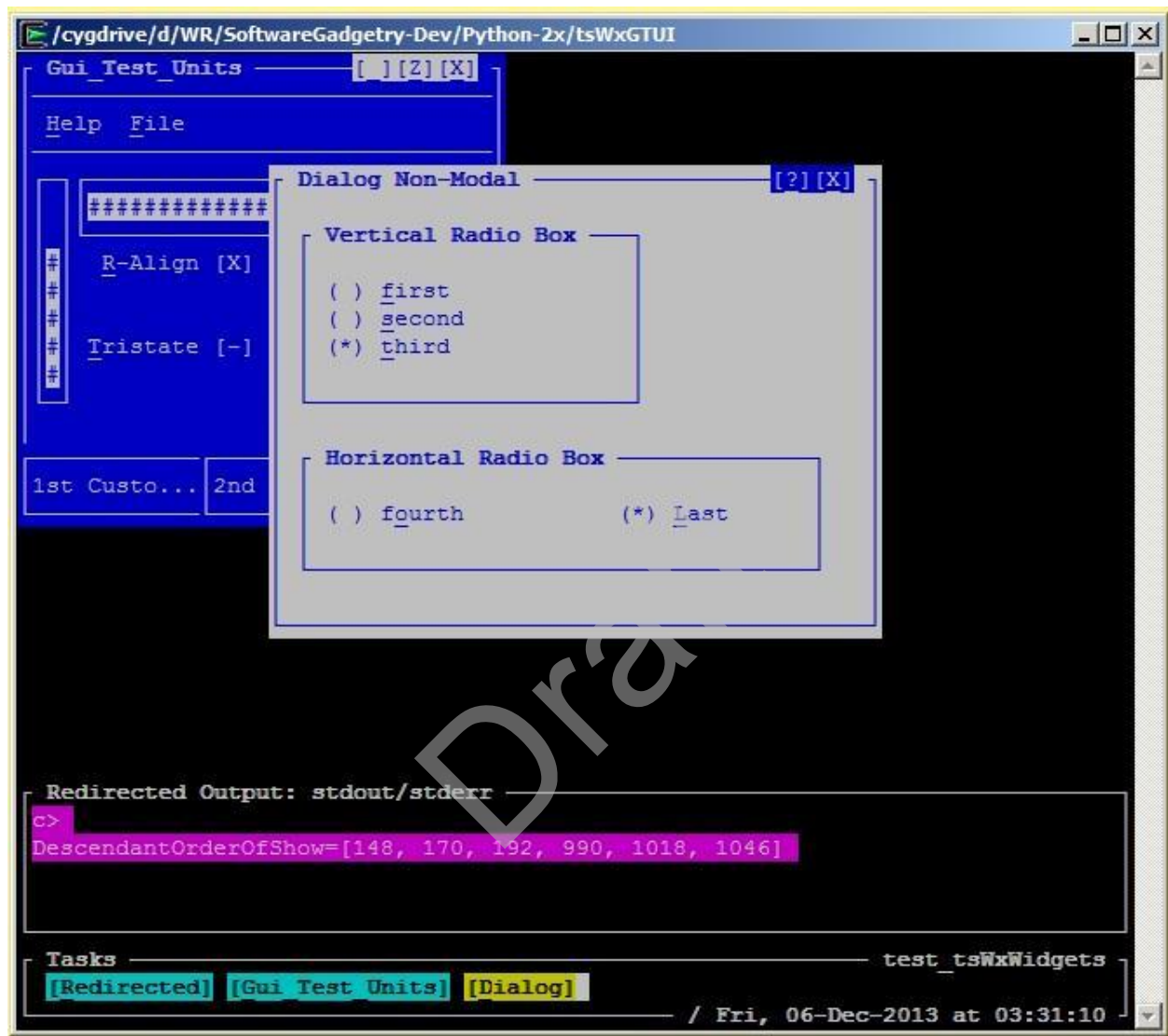
- 1 BLACK
- 2 RED
- 3 GREEN
- 4 YELLOW
- 5 BLUE
- 6 MAGENTA
- 7 CYAN
- 8 WHITE

---

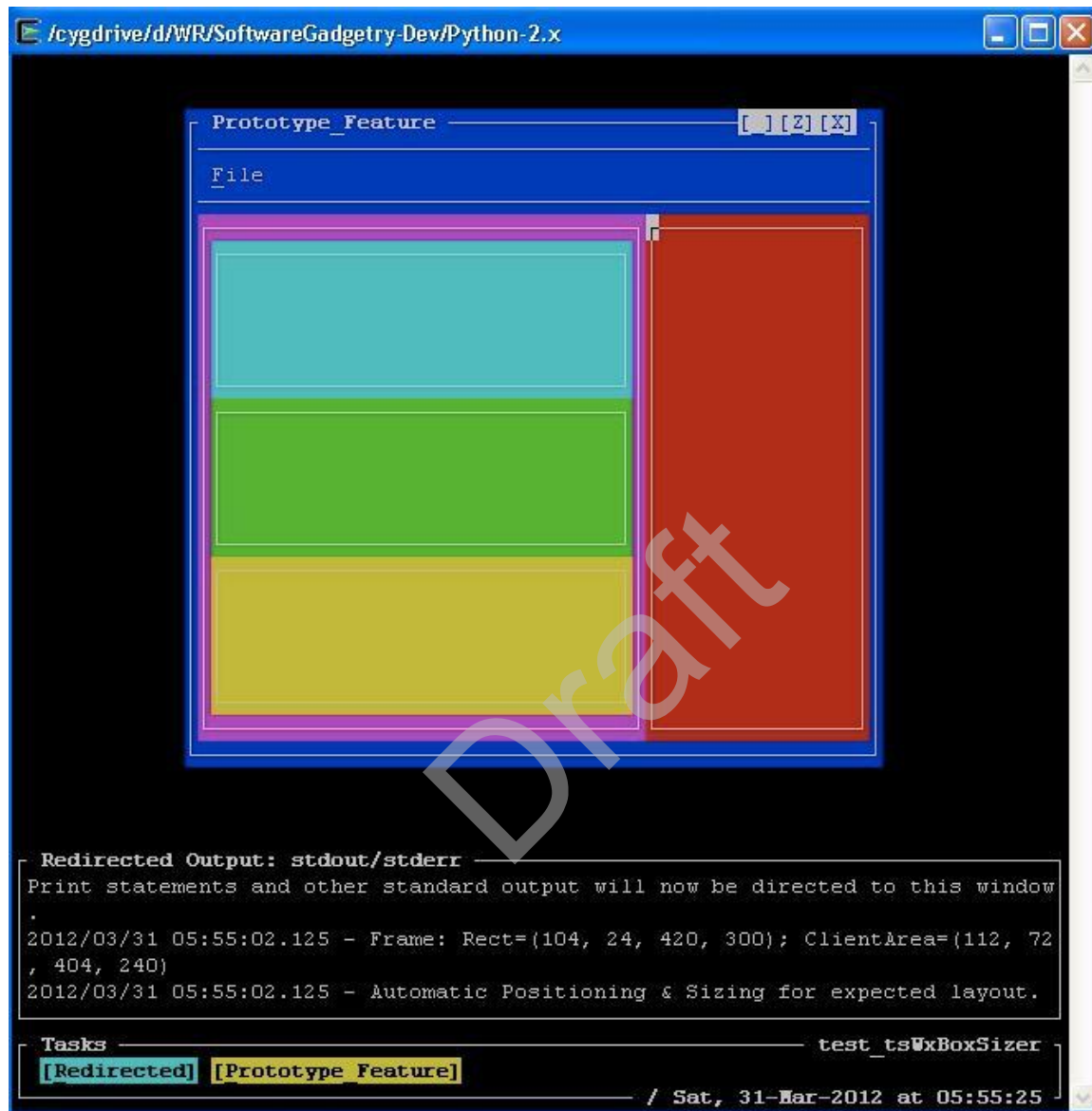
NOTE: The following screen shots demonstrate various widgets on a terminal that reports having colors. It is the same application that runs on terminals that report NOT having colors. The application remains unaware of the presence or absence of colors.

---

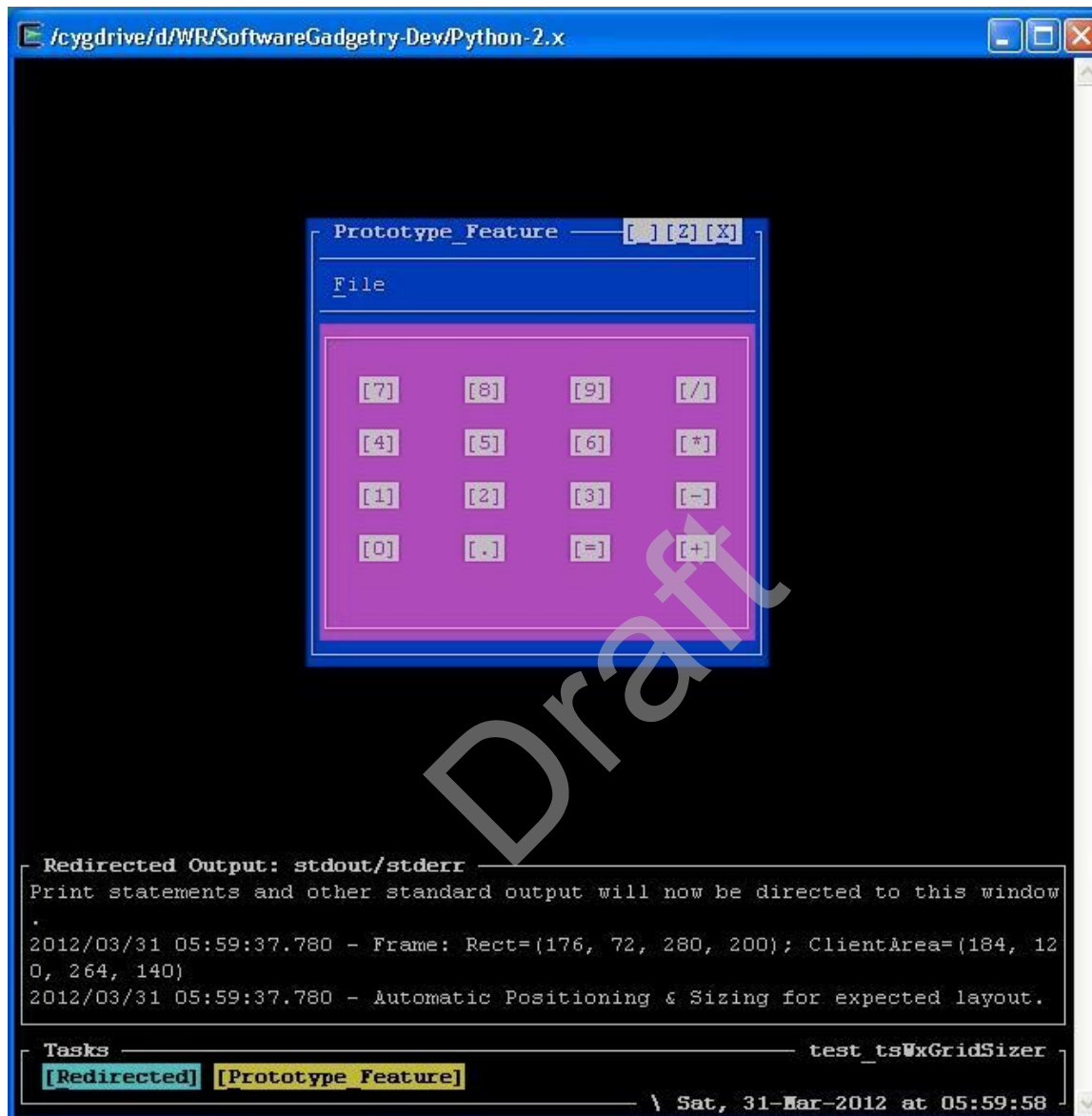
### 1.1.1 test\_tsWxWidgets using xterm (via Cygwin mintty)



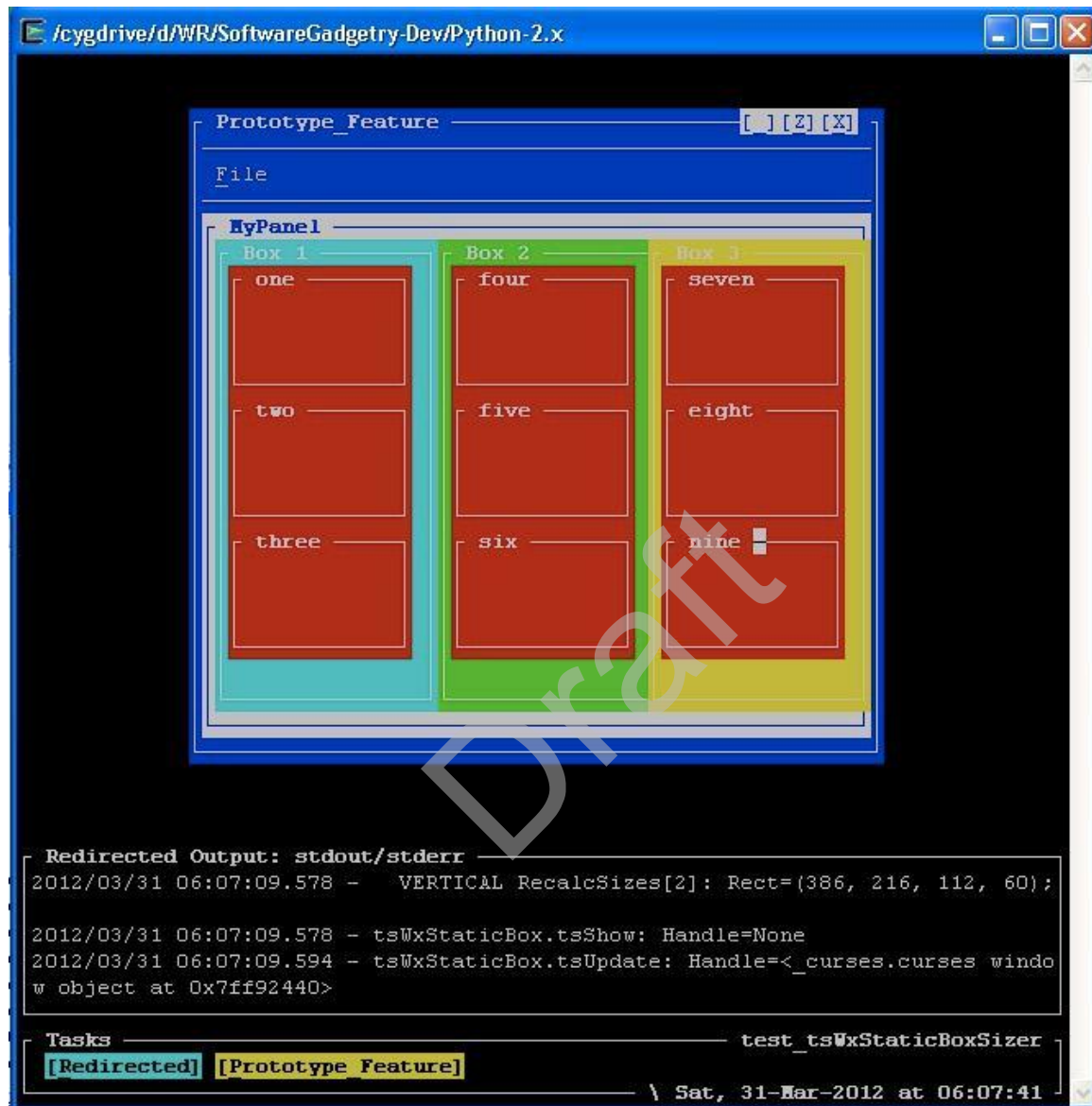
### 1.1.2 test\_tsWxBoxSizer using xterm (via Cygwin mintty)



### 1.1.3 test\_tsWxGridSizer using xterm (via Cygwin mintty)

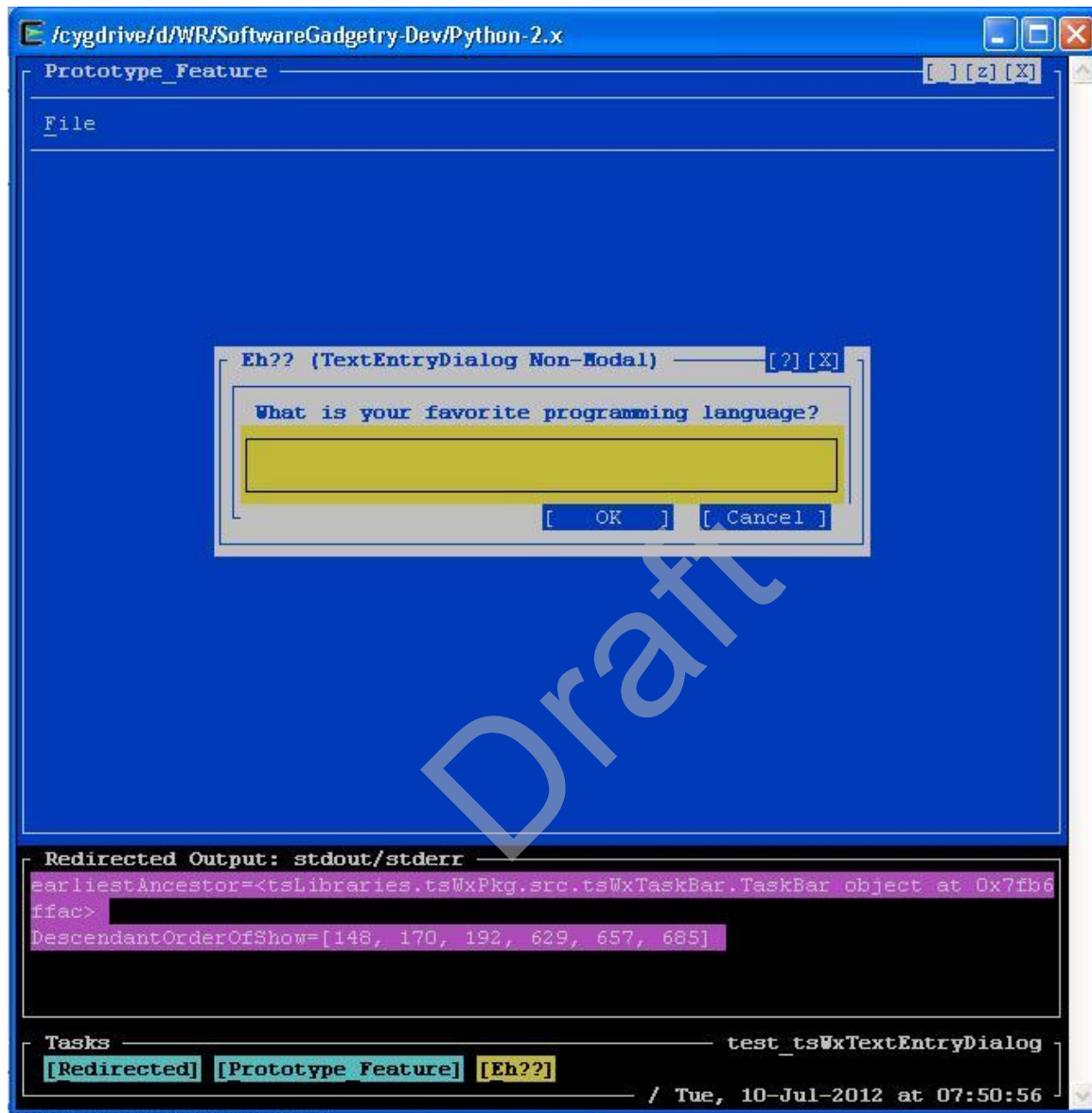


### 1.1.4 test\_tsWxStaticBoxSizer using xterm (via Cygwin mintty)

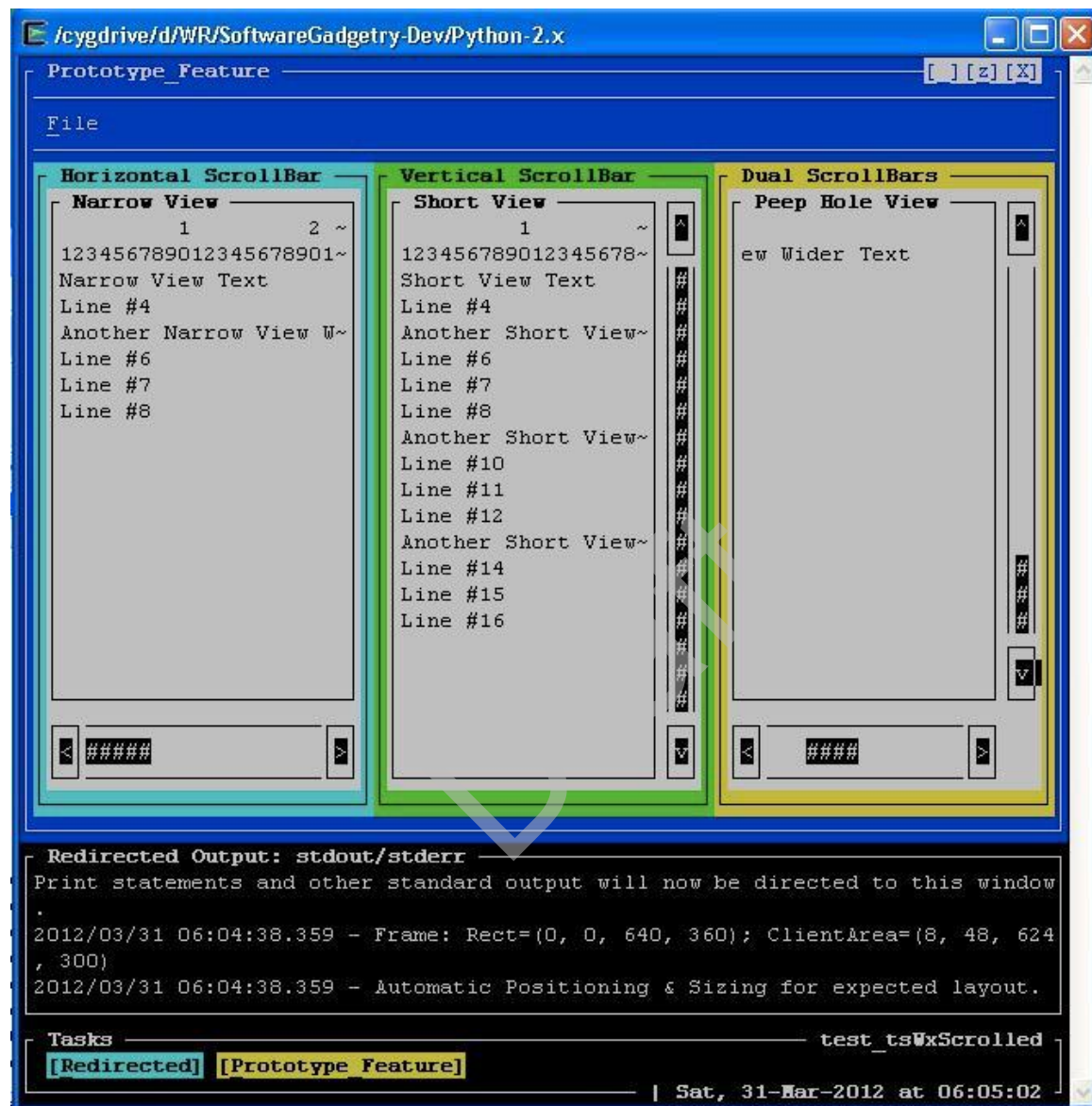




### 1.1.5 test\_tsWxTextEntryDialog using xterm (via Cygwin mintty)



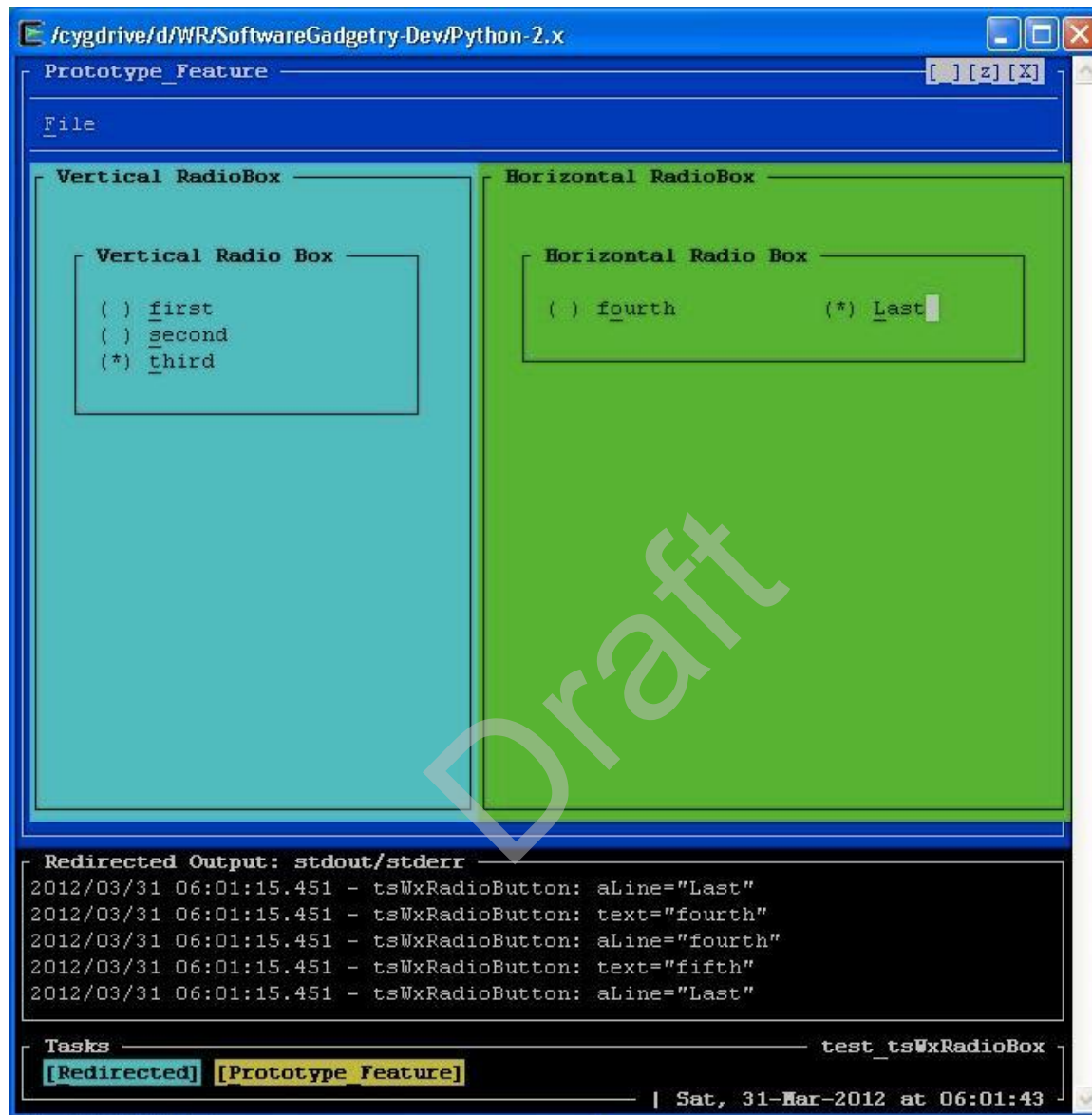
## 1.1.6 test\_tsWxScrolled using xterm (via Cygwin mintty)



### 1.1.7 test\_tsWxScrollBar using xterm (via Cygwin mintty)



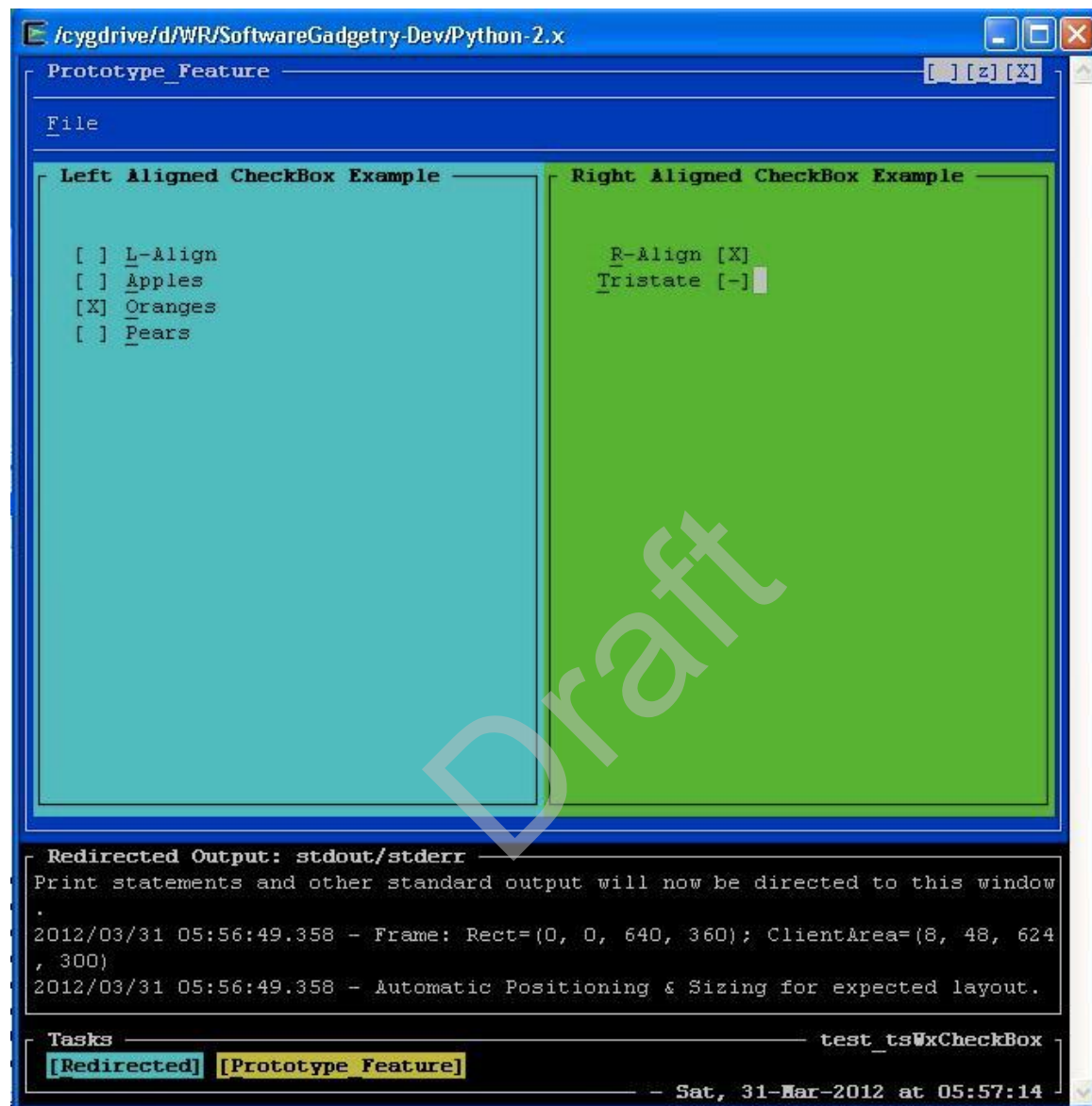
### 1.1.8 test\_tsWxRadioBox using xterm (via Cygwin mintty)



### 1.1.9 test\_tsWxGauge using xterm (via Cygwin mintty)



### 1.1.10 test\_tsWxCheckBox using xterm (via Cygwin mintty)



---

## 1.2 VT-100 with 1-Color / 2-Color Pairs

From Wikipedia, the free encyclopedia:

"The VT100 is a video terminal that was made by Digital Equipment Corporation (DEC). Its detailed attributes became the de facto standard for terminal emulators to emulate.

It was introduced in August 1978, following its predecessor, the VT52, and communicated with its host system over serial lines using the ASCII character set and control sequences (a.k.a. escape sequences) standardized by ANSI. The VT100 was also the first Digital mass-market terminal to incorporate "graphic renditions" (blinking, bolding, reverse video, and underlining) as well as a selectable 80 or 132 column display. All setup of the VT100 was accomplished using interactive displays presented on the screen; the setup data was stored in non-volatile memory within the terminal. The VT100 also introduced an additional character set that allowed the drawing of on-screen forms.

The control sequences used by the VT100 family are based on the ANSI X3.64 standard, also known as ECMA-48 and ISO/IEC 6429. These are sometimes referred to as ANSI escape codes. The VT100 was not the first terminal to be based on X3.64—The Heath Company had a microprocessor-based video terminal, the Heathkit H-19 (H19), that implemented a subset of the standard proposed by ANSI in X3.64. In addition, the VT100 provided backwards compatibility for VT52 users, with support for the VT52 control sequences.

In 1983, the VT100 was replaced by the more-powerful VT200 series terminals such as the VT220.

In August 1995 the terminal business of Digital was sold to Boundless Technologies."

Typically, vt100/vt220 terminals support keyboard (without mouse) input and a display whose character cells consist of a single WHITE, GREEN or ORANGE color phosphor per pixel that are independently programmable in intensity so as to produce any of 2-colors (with their associated 2-color pairs) per character:

- 1** BLACK (phosphor pixel "OFF")
- 2** WHITE, GREEN or ORANGE (phosphor pixel "ON")

---

NOTE: The following screen shots demonstrate various widgets on a terminal that reports NOT having colors. It is the same application that runs on terminals that report having colors. The application remains unaware of the presence or absence of colors.

---



## 1.2.1 test\_tsWxWidgets using vt100 (via Cygwin mintty)

```

/cygdrive/d/WR/SoftwareGadgetry-Dev/Python-2.x
widgets_communicate [?] [Z] [X]
Help File
#####
# R-Align [X]
#
# Tristate [-]
#
1st Custo... 2nd

Dialog Non-Modal [?] [X]
  Vertical Radio Box
    ( ) first
    ( ) second
    (*) third

  Horizontal Radio Box
    ( ) fourth    (*) Last

Redirected Output: stdout/stderr
2012/03/30 03:41:43.108 - tsWxRadioButton: aLine="Last"
2012/03/30 03:41:43.124 - tsWxRadioButton: text="fourth"
2012/03/30 03:41:43.124 - tsWxRadioButton: aLine="fourth"
2012/03/30 03:41:43.124 - tsWxRadioButton: text="fifth"
2012/03/30 03:41:43.124 - tsWxRadioButton: aLine="Last"

Tasks test_tsWxWidgets
[Redirected] [Widgets_Communicate] [Dialog]
/ Fri, 30-Mar-2012 at 03:42:26

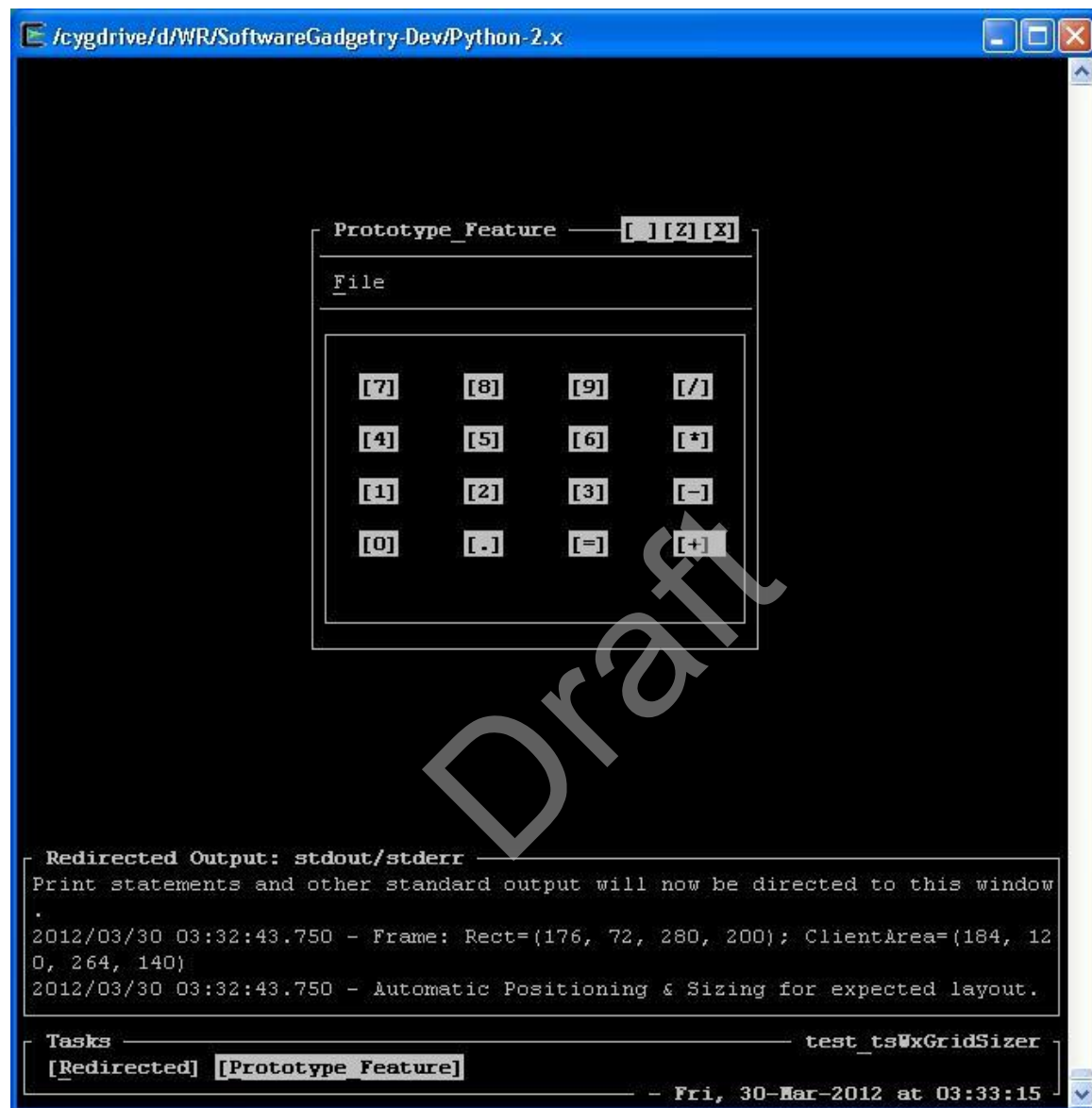
```



## 1.2.2 test\_tsWxBoxSizer using vt100 (via Cygwin mintty)



### 1.2.3 test\_tsWxGridSizer using vt100 (via Cygwin mintty)-



```
/cygdrive/d/WR/SoftwareGadgetry-Dev/Python-2.x

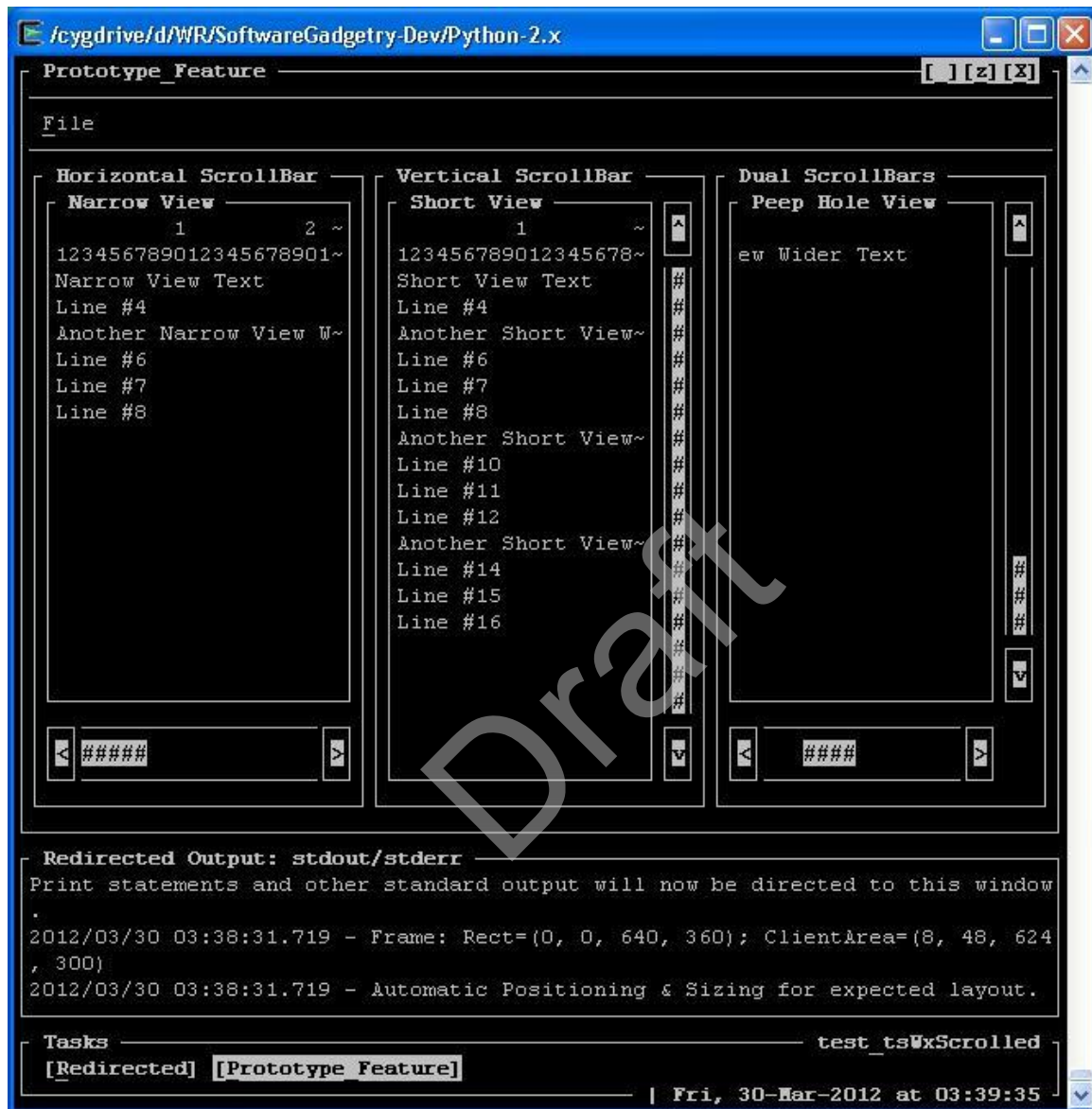
Prototype_Feature [ ] [Z] [X]
File
[7] [8] [9] [/]
[4] [5] [6] [*]
[1] [2] [3] [-]
[0] [.] [=] [+]

Draft

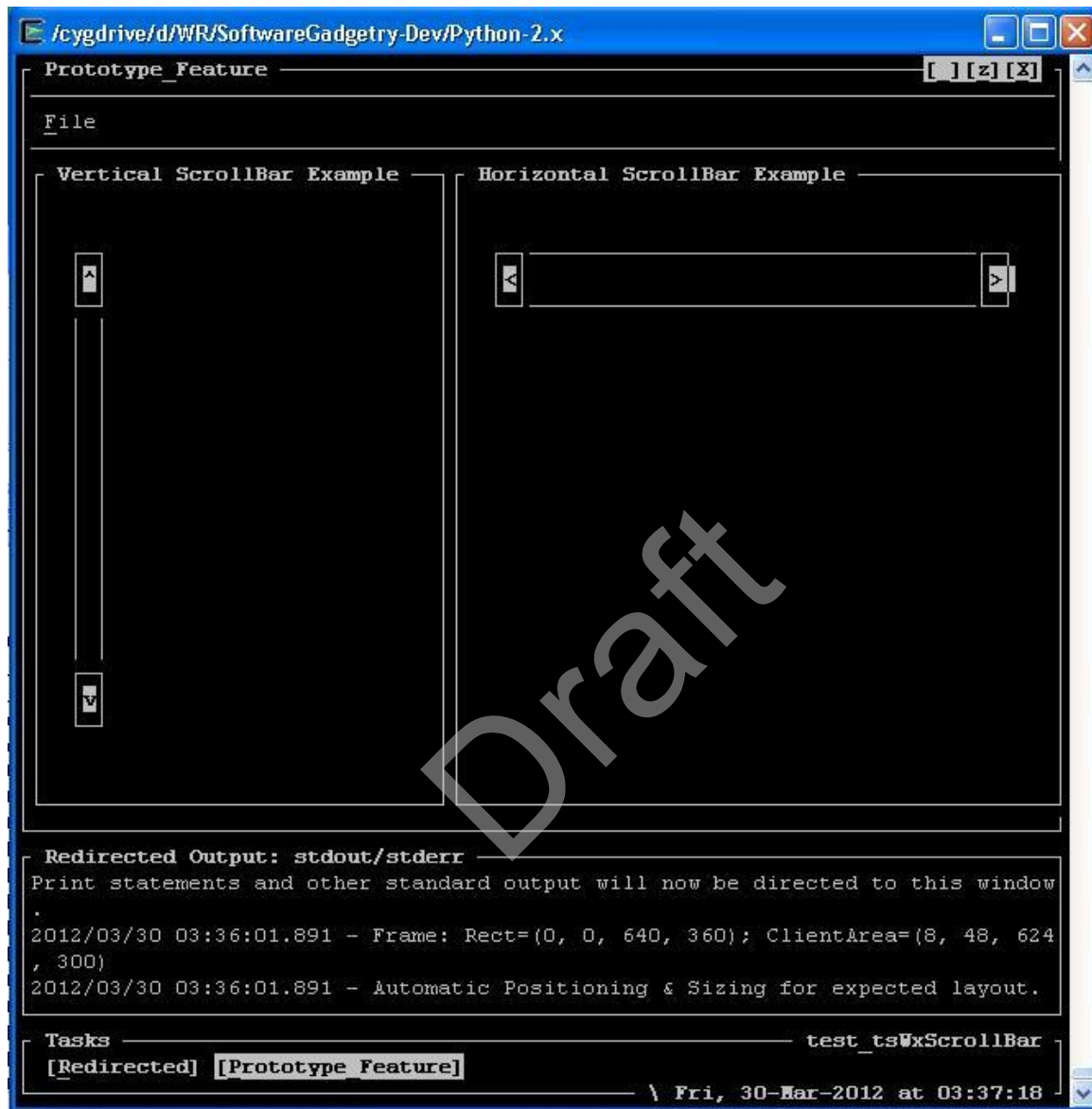
Redirected Output: stdout/stderr
Print statements and other standard output will now be directed to this window
.
2012/03/30 03:32:43.750 - Frame: Rect=(176, 72, 280, 200); ClientArea=(184, 120, 264, 140)
2012/03/30 03:32:43.750 - Automatic Positioning & Sizing for expected layout.

Tasks test_tsWxGridSizer
[Redirected] [Prototype_Feature]
- Fri, 30-Mar-2012 at 03:33:15
```

## 1.2.4 test\_tsWxScrolled using vt100 (via Cygwin mintty)



### 1.2.5 test\_tsWxScrollBar using vt100 (via Cygwin mintty)



## 1.2.6 test\_tsWxRadioBox using vt100 (via Cygwin mintty)

```
/cygdrive/d/WR/SoftwareGadgetry-Dev/Python-2.x

Prototype_Feature [ ] [z] [X]

File

Vertical RadioBox Horizontal RadioBox

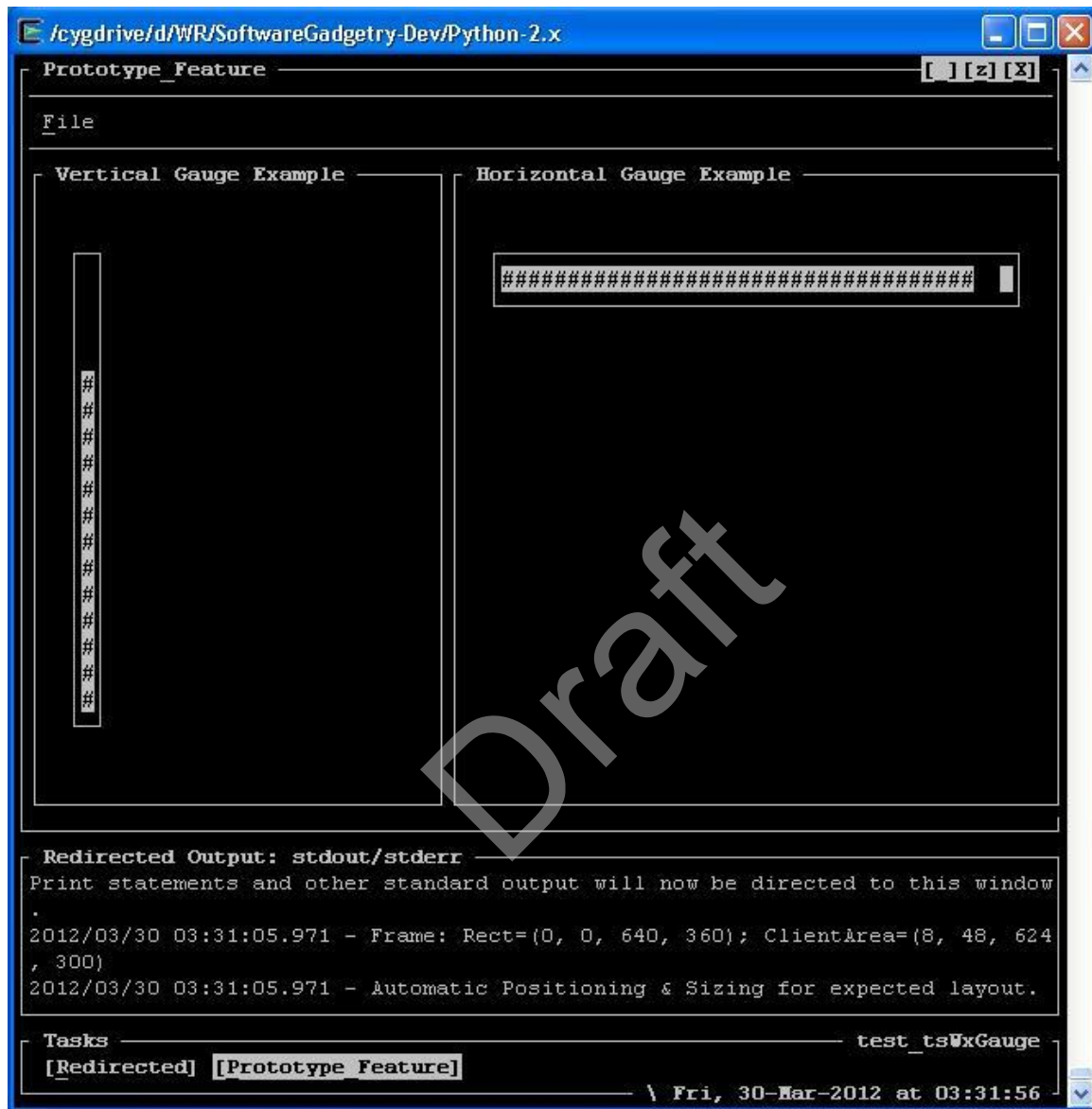
Vertical Radio Box Horizontal Radio Box

( ) first ( ) fourth (*) Last
( ) second
(*) third

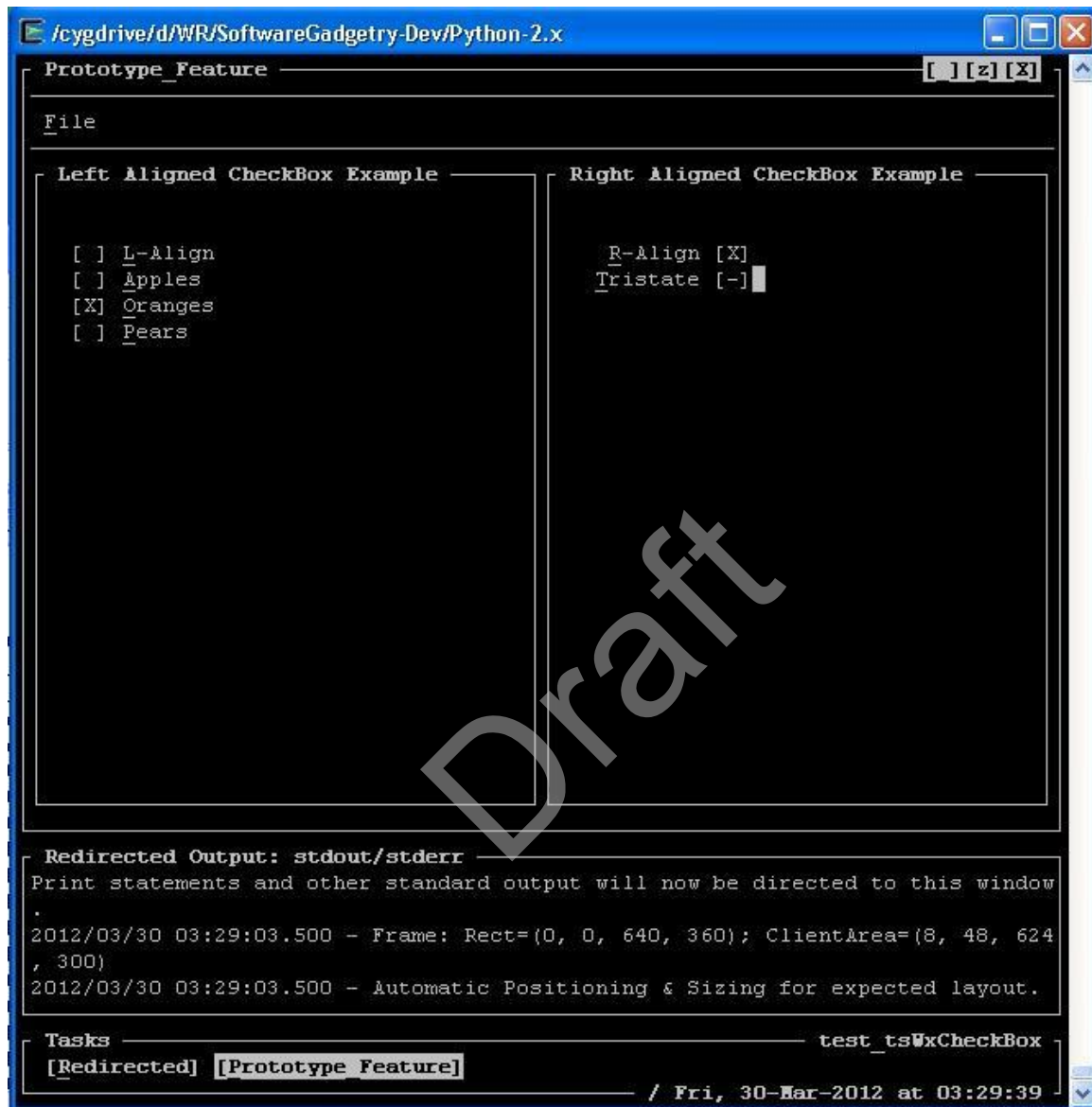
Redirected Output: stdout/stderr
2012/03/30 03:34:13.016 - tsWxRadioButton: aLine="Last"
2012/03/30 03:34:13.016 - tsWxRadioButton: text="fourth"
2012/03/30 03:34:13.016 - tsWxRadioButton: aLine="fourth"
2012/03/30 03:34:13.016 - tsWxRadioButton: text="fifth"
2012/03/30 03:34:13.016 - tsWxRadioButton: aLine="Last"

Tasks test_tsWxRadioBox
[Redirected] [Prototype Feature]
- Fri, 30-Mar-2012 at 03:34:50
```

## 1.2.7 test\_tsWxGauge using vt100 (via Cygwin mintty)



## 1.2.8 test\_tsWxCheckBox using vt100 (via Cygwin mintty)



---

## 1.3 Multi-Session Desktop

From Wikipedia, the free encyclopedia

"In computer science, in particular networking, a session is a semi-permanent interactive information interchange, also known as a dialogue, a conversation or a meeting, between two or more communicating devices, or between a computer and user (see Login session). A session is set up or established at a certain point in time, and then torn down at some later point. An established communication session may involve more than one message in each direction. A session is typically, but not always, stateful, meaning that at least one of the communicating parts needs to save information about the session history in order to be able to communicate, as opposed to stateless communication, where the communication consists of independent requests with responses.

An established session is the basic requirement to perform a connection-oriented communication. A session also is the basic step to transmit in connectionless communication modes. However any unidirectional transmission does not define a session.[1]

Communication sessions may be implemented as part of protocols and services at the application layer, at the session layer or at the transport layer in the OSI model.

- Application layer examples:
  - HTTP sessions, which allow associating information with individual visitors
  - A telnet remote login session
- Session layer example:
  - A Session Initiation Protocol (SIP) based Internet phone call
- Transport layer example:
  - A TCP session, which is synonymous to a TCP virtual circuit, a TCP connection, or an established TCP socket.



In the case of transport protocols that do not implement a formal session layer (e.g., UDP) or where sessions at the application layer are generally very short-lived (e.g., HTTP), sessions are maintained by a higher level program using a method defined in the data being exchanged. For example, an HTTP exchange between a browser and a remote host may include an HTTP cookie which identifies state, such as a unique session ID, information about the user's preferences or authorization level.

HTTP/1.0 was thought to only allow a single request and response during one Web/HTTP Session. However a workaround was created by David Hostettler Wain in 1996 such that it was possible to use session IDs to allow multiple phase Web Transaction Processing (TP) Systems (in ICL[disambiguation needed] nomenclature), with the first implementation being called Deity. Protocol version HTTP/1.1 further improved by completing the Common Gateway Interface (CGI) making it easier to maintain the Web Session and supporting HTTP cookies and file uploads.

Most client-server sessions are maintained by the transport layer - a single connection for a single session. However each transaction phase of a Web/HTTP session creates a separate connection. Maintaining session continuity between phases required a session ID. The session ID is embedded within the <A HREF> or <FORM> links of dynamic web pages so that it is passed back to the CGI. CGI then uses the session ID to ensure session continuity between transaction phases. One advantage of one connection-per-phase is that it works well over low bandwidth (modem) connections. Deity used a sessionID, screenID and actionID to simplify the design of multiple phase sessions."

