

Gui mit wxDemo

Einstieg in die Entwicklung
grafischer Benutzeroberflächen
mit Hilfe von wxDemo

Gui mit wxDemo

wxPython: (A Demonstration)

File Demo Options Help

wxPython Demos

- wxPython Overview
 - Recent Additions/Updates
 - Simplebook
 - LEDNumberCtrl
 - DynamicSashWindow
 - TreeListCtrl
 - Frames and Dialogs
 - AUI_DockingWindowMgr
 - AUI_MDI
 - Dialog
 - Frame
 - MDIWindows
 - MiniFrame
 - Wizard
 - Common Dialogs
 - More Dialogs
 - Core Windows/Controls
 - BitmapButton
 - Button
 - CheckBox
 - CheckListBox
 - Choice
 - ComboBox
 - CommandLinkButton
 - DVC_CustomRenderer
 - DVC_DataViewModel
 - DVC_IndexListModel
 - DVC_ListCtrl
 - DVC_TreeCtrl
 - Gauge
 - Grid

Filter Demos:
Search

wxPython Overview Demo Code

wxPython

wxPython is a **GUI toolkit** for the Python programming language. It allows Python programmers to create programs with a robust, highly functional graphical user interface, simply and easily. It is implemented as a Python extension module (native code) that wraps the popular wxWindows cross platform GUI library, which is written in C++.

Like Python and wxWindows, wxPython is **Open Source** which means that it is free for anyone to use and the source code is available for anyone to look at and modify. Or anyone can contribute fixes or enhancements to the project.

wxPython is a **cross-platform** toolkit. This means that the same program will run on multiple platforms without modification. Currently supported platforms are 32-bit Microsoft Windows, most Unix or unix-like systems, and Macintosh OS X. Since the language is Python, wxPython programs are **simple, easy** to write and easy to understand.

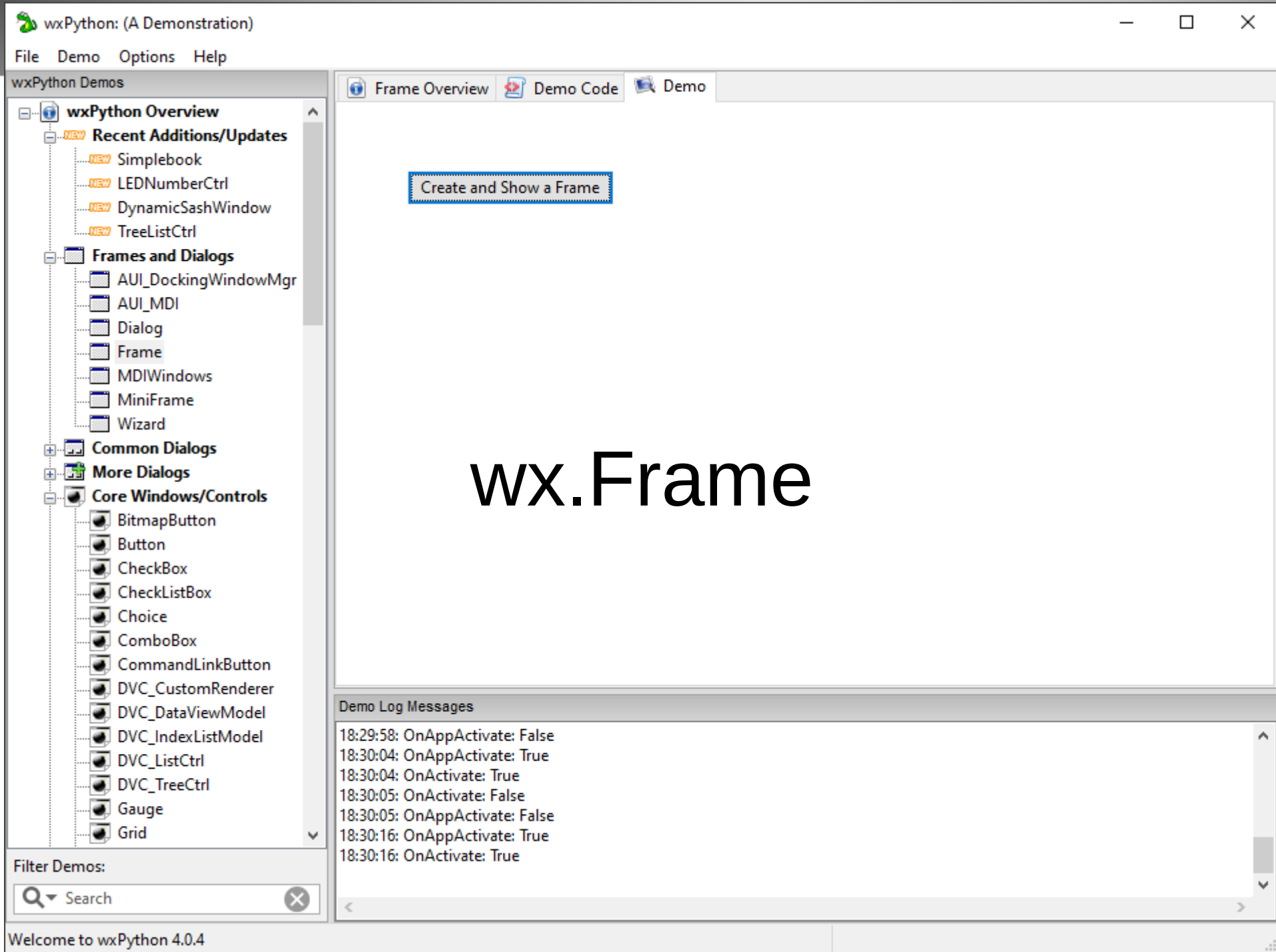
This demo is not only a collection of test cases for wxPython, but is also designed to help you learn about and how to use wxPython. Each sample is listed in the tree control on the left. When a sample is selected in the tree then a module is loaded and run (usually in a tab of this notebook,) and the source code of the module is loaded in another tab for you to browse and learn from.

Demo Log Messages

```
17:49:51: OnActivate: True
17:50:26: OnActivate: False
17:50:26: OnAppActivate: False
17:50:38: OnAppActivate: True
17:50:38: OnActivate: True
```

Welcome to wxPython 4.0.4

Gui mit wxDemo



Gui mit wxDemo

The screenshot shows the wxPython Demo application window. The title bar reads "wxPython: (A Demonstration)". The menu bar includes "File", "Demo", "Options", and "Help". The main window is divided into several panes:

- Left Pane (Tree View):** Titled "wxPython Demos", it contains a hierarchical tree of widget classes:
 - wxPython Overview**
 - Recent Additions/Updates
 - Simplebook
 - LEDNumberCtrl
 - DynamicSashWindow
 - TreeListCtrl
 - Frames and Dialogs
 - AUI_DockingWindowMgr
 - AUI_MDI
 - Dialog
 - Frame
 - MDIWindows
 - MiniFrame
 - Wizard
 - Common Dialogs
 - More Dialogs
 - Core Windows/Controls
 - BitmapButton
 - Button
 - CheckBox
 - CheckListBox
 - Choice
 - ComboBox
 - CommandLinkButton
 - DVC_CustomRenderer
 - DVC_DataViewModel
 - DVC_IndexListModel
 - DVC_ListCtrl
 - DVC_TreeCtrl
 - Gauge
 - Grid
- Bottom Left:** A "Filter Demos:" section with a search input field containing "Search".
- Bottom:** A "Demo Log Messages" pane showing a list of system events with timestamps and status (e.g., "OnActivate: True", "OnAppActivate: False").

The main content area of the application features a button labeled "Create and Show a Frame". Clicking this button opens a secondary window titled "This is a wx.Frame" which contains a "Close Me" button.

Gui mit wxDemo

The screenshot shows the wxPython Demo application interface. On the left is a tree view titled "wxPython Overview" with categories like "Recent Additions/Updates", "Frames and Dialogs", "Common Dialogs", "More Dialogs", and "Core Windows/Controls". The "Core Windows/Controls" category is expanded, showing various widgets such as Button, CheckBox, and Grid. At the bottom left is a "Filter Demos:" search box. The main area is split into two panes: "Frame Overview" and "Demo Code". The "Demo Code" pane shows the Python code for a class named `MyFrame`, which inherits from `wx.Frame`. The code includes an `__init__` method that sets up a panel and a "Close Me" button, and two event handlers: `OnCloseMe` and `OnCloseWindow`. The "Demo Log Messages" pane at the bottom right displays a log of application events with timestamps.

```
5 # -----
6
7 class MyFrame(wx.Frame):
8     def __init__(
9         self, parent, ID, title, pos=wx.DefaultPosition,
10        size=wx.DefaultSize, style=wx.DEFAULT_FRAME_STYLE
11        ):
12
13        wx.Frame.__init__(self, parent, ID, title, pos, size, style)
14        panel = wx.Panel(self, -1)
15
16        button = wx.Button(panel, 1003, "Close Me")
17        button.SetPosition((15, 15))
18        self.Bind(wx.EVT_BUTTON, self.OnCloseMe, button)
19        self.Bind(wx.EVT_CLOSE, self.OnCloseWindow)
20
21
22    def OnCloseMe(self, event):
23        self.Close(True)
24
25    def OnCloseWindow(self, event):
26        self.Destroy()
27
28 # -----
```

Demo Log Messages

```
18:31:52: OnAppActivate: False
18:31:52: OnAppActivate: True
18:31:52: OnActivate: True
18:31:53: OnActivate: False
18:31:53: OnAppActivate: False
18:32:02: OnAppActivate: True
18:32:02: OnActivate: True
```

Gui mit wxDemo

```
import wx

### -----
class MyFrame(wx.Frame):
    def __init__(
        self, parent, ID, title, pos=wx.DefaultPosition,
        size=wx.DefaultSize, style=wx.DEFAULT_FRAME_STYLE
    ):

        ... (siehe Bild)

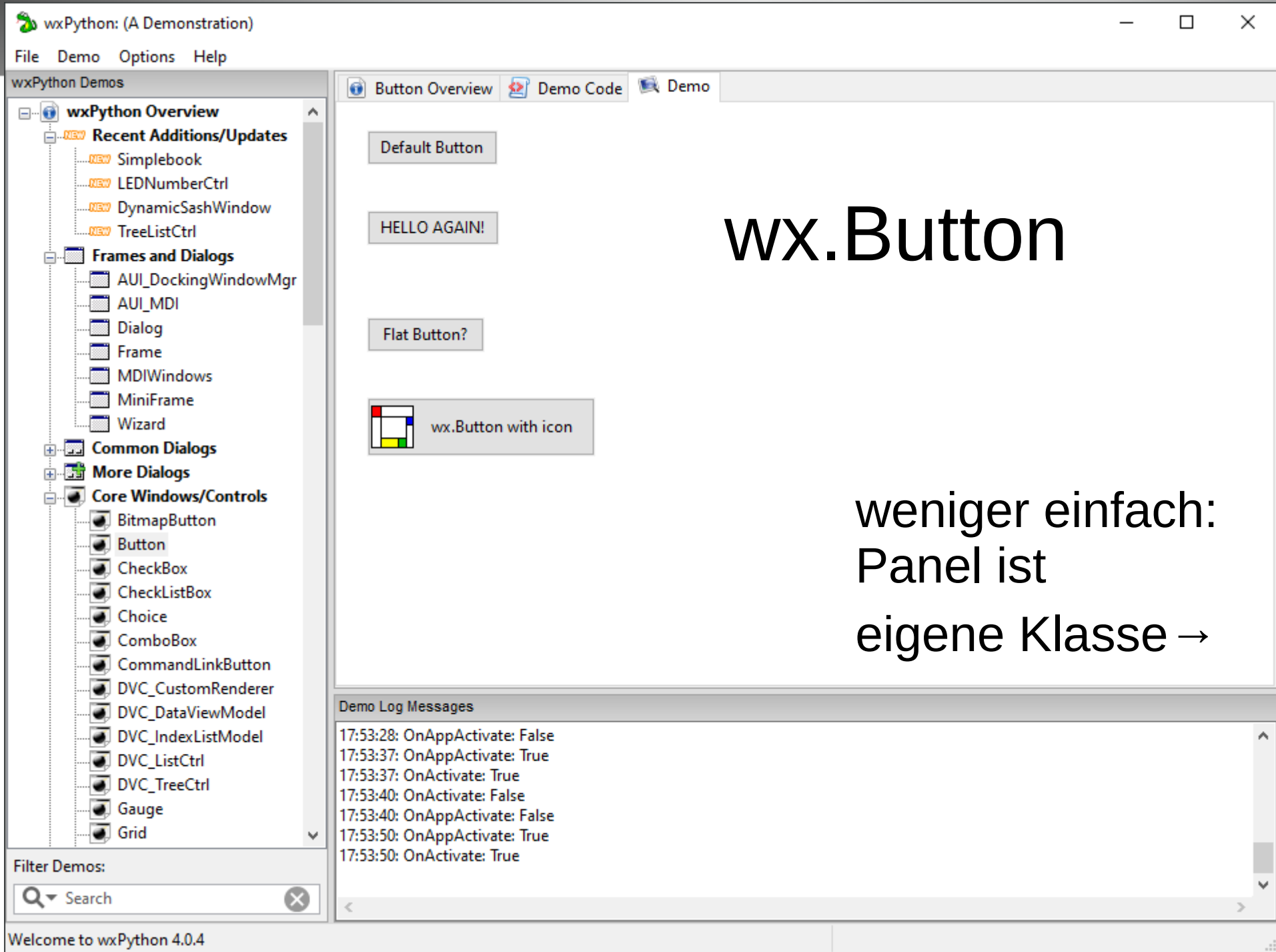
    def OnCloseMe(self, event):
        self.Close(True)

    def OnCloseWindow(self, event):
        self.Destroy()
```

```
### -----
if __name__ == '__main__':
    app=wx.App()
    fenster=MyFrame(None, -1, "Fenster")
    app.SetTopWindow(fenster)
    fenster.Show(True)
    app.MainLoop()
```

Ergänzungen

Gui mit wxDemo



The screenshot shows the wxPython Demo application window. The title bar reads "wxPython: (A Demonstration)". The menu bar includes "File", "Demo", "Options", and "Help". The main window is titled "wxPython Demos" and contains several tabs: "Button Overview", "Demo Code", and "Demo".

The "Button Overview" tab is active, displaying a collection of buttons:

- Default Button
- HELLO AGAIN!
- Flat Button?
- wx.Button with icon (with a small icon)

The left sidebar shows a tree view of the wxPython class hierarchy under "wxPython Overview":

- Recent Additions/Updates
 - Simplebook
 - LEDNumberCtrl
 - DynamicSashWindow
 - TreeListCtrl
- Frames and Dialogs
 - AUI_DockingWindowMgr
 - AUI_MDI
 - Dialog
 - Frame
 - MDIWindows
 - MiniFrame
 - Wizard
- Common Dialogs
- More Dialogs
- Core Windows/Controls
 - BitmapButton
 - Button
 - CheckBox
 - CheckListBox
 - Choice
 - ComboBox
 - CommandLinkButton
 - DVC_CustomRenderer
 - DVC_DataViewModel
 - DVC_IndexListModel
 - DVC_ListCtrl
 - DVC_TreeCtrl
 - Gauge
 - Grid

At the bottom of the sidebar is a "Filter Demos:" section with a search box.

The bottom of the window features a "Demo Log Messages" window with the following log entries:

```
17:53:28: OnAppActivate: False
17:53:37: OnAppActivate: True
17:53:37: OnActivate: True
17:53:40: OnActivate: False
17:53:40: OnAppActivate: False
17:53:50: OnAppActivate: True
17:53:50: OnActivate: True
```

The status bar at the very bottom reads "Welcome to wxPython 4.0.4".

wx.Button

weniger einfach:
Panel ist
eigene Klasse →

Gui mit wxDemo

The screenshot shows the wxPython Demo application interface. On the left is the 'wxPython Overview' tree, which is categorized into 'Recent Additions/Updates', 'Frames and Dialogs', 'Common Dialogs', 'More Dialogs', and 'Core Windows/Controls'. The 'Core Windows/Controls' section is expanded, showing various widget types like Button, CheckBox, and ComboBox. At the bottom left, there is a 'Filter Demos' search box.

The main area is the 'Demo Code' editor, showing the Python code for the 'TestPanel' class. The code defines a class that inherits from 'wx.Panel' and includes three buttons with different styles and tool tips.

```
8 class TestPanel(wx.Panel):
9     def __init__(self, parent, log):
10        wx.Panel.__init__(self, parent, -1,
11                          style=wx.NO_FULL_REPAINT_ON_RESIZE)
12        self.log = log
13
14        b = wx.Button(self, 10, "Default Button", (20, 20))
15        self.Bind(wx.EVT_BUTTON, self.OnClick, b)
16        b.SetDefault()
17        b.SetSize(b.GetBestSize())
18
19        b = wx.Button(self, 20, "HELLO AGAIN!", (20, 80))
20        self.Bind(wx.EVT_BUTTON, self.OnClick, b)
21        b.SetToolTip("This is a Hello button...")
22
23        b = wx.Button(self, 40, "Flat Button?", (20,160), style=wx.NO_BORDER)
24        b.SetToolTip("This button has a style flag of wx.NO_BORDER.\n"
25                    "On some platforms that will give it a flattened lo
26        self.Bind(wx.EVT_BUTTON, self.OnClick, b)
27
28        b = wx.Button(self, 50, "wx.Button with icon", (20, 220))
29        b.SetToolTip("wx.Button can now have an icon on the left, right,\n"
30                    "above or below the label.")
31        self.Bind(wx.EVT_BUTTON, self.OnClick, b)
```

At the bottom right, the 'Demo Log Messages' window displays a series of log entries, including 'OnAppActivate' and 'OnActivate' events with their respective boolean values.

Gui mit wxDemo

wxPython: (A Demonstration)

File Demo Options Help

wxPython Demos

- PageSetupDialog
- PrintDialog
- ProgressDialog
- SingleChoiceDialog
- TextEntryDialog
- RearrangeDialog
- RichMessageDialog
- More Dialogs
 - ImageBrowser
 - ScrolledMessageDialog
- Core Windows/Controls
 - BitmapButton
 - Button
 - CheckBox
 - CheckListBox
 - Choice
 - ComboBox
 - CommandLinkButton
 - DVC_CustomRenderer
 - DVC_DataViewModel
 - DVC_IndexListModel
 - DVC_ListCtrl
 - DVC_TreeCtrl
 - Gauge

Filter Demos:

Button Overview Demo Code Demo

Active Version: Original Modified Save Changes Delete Modified

```
7
8 class TestPanel(wx.Panel):
9     def __init__(self, parent, log):
10        wx.Panel.__init__(self, parent, -1,
11                          style=wx.NO_FULL_REPAINT_ON_RESIZE)
12        self.log = log
13
14        b = wx.Button(self, 10, "Default Button", (20, 20))
15        self.Bind(wx.EVT_BUTTON, self.OnClick, b)
16        b.SetDefault()
17        b.SetSize(b.GetBestSize())
18
19        b = wx.Button(self, 20, "HELLO AGAIN!", (20, 80))
20        self.Bind(wx.EVT_BUTTON, self.OnClick, b)
21        b.SetToolTip("This is a Hello button...")
22
23        b = wx.Button(self, 40, "Flat Button?", (20,160), style=wx.NO_BORI
24        b.SetToolTip("This button has a style flag of wx.NO_BORDER.\n"
25                    "On some platforms that will give it a flattened
26        self.Bind(wx.EVT_BUTTON, self.OnClick, b)
27
```

Demo Log Messages

```
11:14:38: OnAppActivate: False
11:14:46: OnActivate: True
11:14:46: OnAppActivate: True
11:14:46: OnActivate: False
11:14:46: OnAppActivate: False
11:14:46: OnActivate: True
11:14:46: OnAppActivate: True
```

Welcome to wxPython 4.0.7.post2

Gui mit wxDemo

The screenshot shows the wxPython Demo application window. The title bar reads "wxPython: (A Demonstration)". The menu bar includes "File", "Demo", "Options", and "Help". The main window is divided into several panes:

- Left Pane:** A tree view titled "wxPython Overview" showing a hierarchy of controls. The "Core Windows/Controls" section is expanded, listing various widgets like Button, CheckBox, Choice, etc.
- Top Pane:** A tabbed interface with "Button Overview", "Demo Code", and "Demo" tabs. The "Demo Code" tab is active, showing Python code for a button.
- Code Editor:** Displays Python code for a button. A red oval highlights the `OnClick` event handler function:

```
def OnClick(self, event):  
    self.log.write("Click! (%d)\n" % event.GetId())
```
- Bottom Pane:** A "Demo Log Messages" pane showing a list of log entries with timestamps and messages like "OnAppActivate: True".

At the bottom of the window, a status bar reads "Welcome to wxPython 4.0.4".

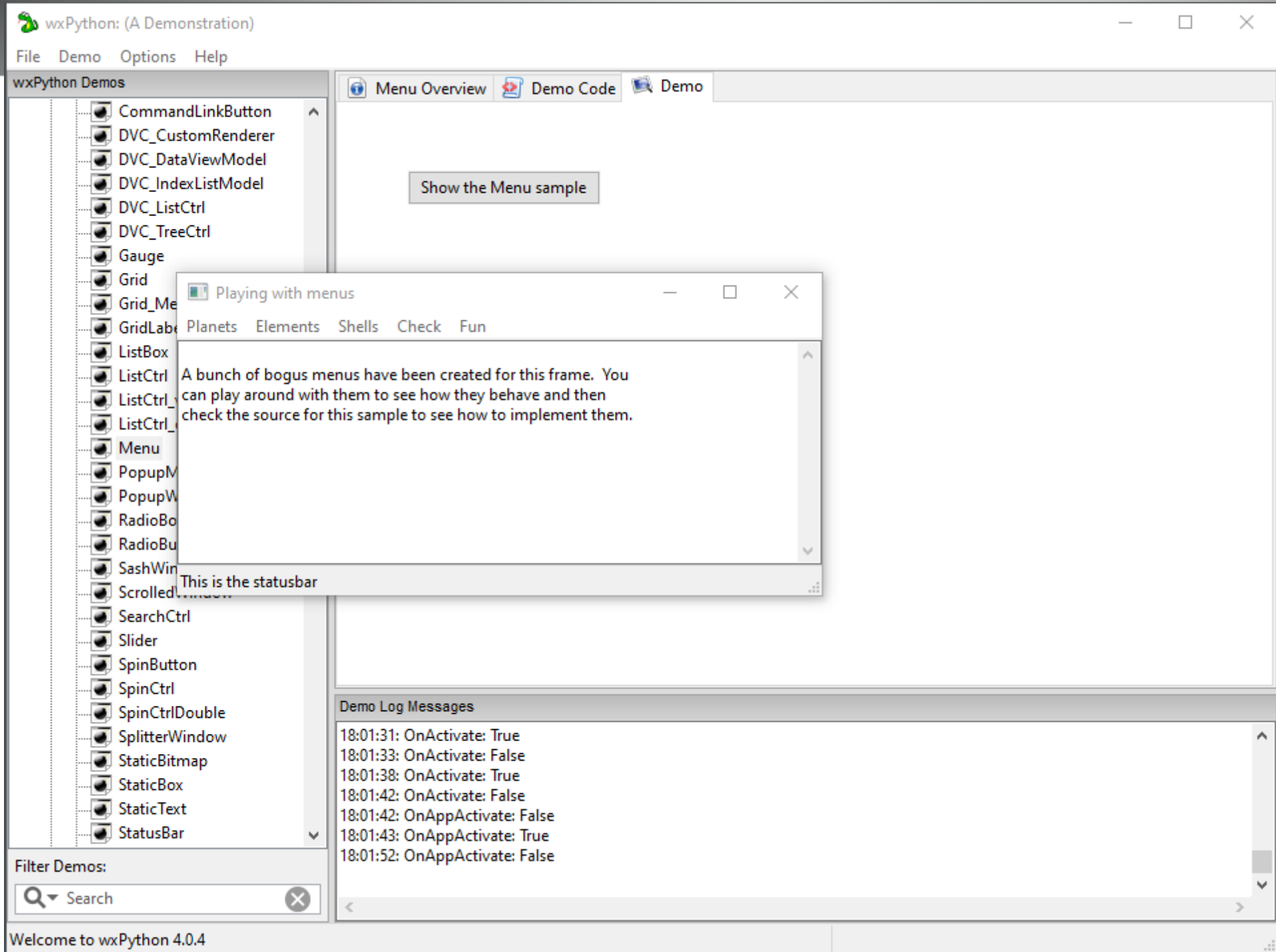
Gui mit wxDemo

The screenshot shows the wxPython GUI with the following components:

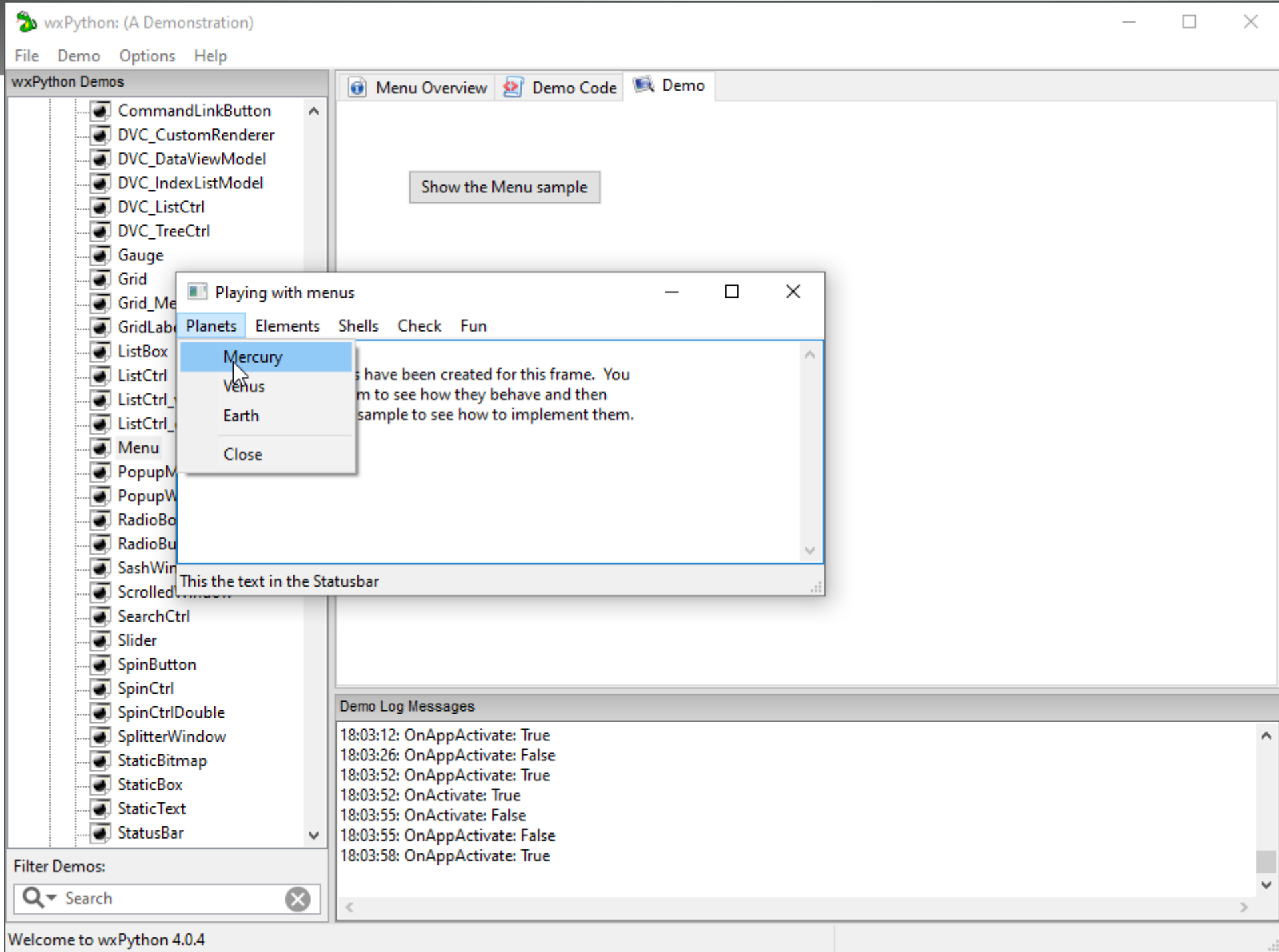
- Window Title:** wxPython: (A Demonstration)
- Menu Bar:** File Demo Options Help
- Tabbed Interface:** Menu Overview (selected), Demo Code, Demo
- Left Panel (wxPython Demos):** A list of widget classes with checkboxes. The **Menu** class is selected and highlighted in blue. Other classes include CommandLinkButton, DVC_CustomRenderer, DVC_DataViewModel, DVC_IndexListModel, DVC_ListCtrl, DVC_TreeCtrl, Gauge, Grid, Grid_MegaExample, GridLabelRenderer, ListBox, ListCtrl, ListCtrl_virtual, ListCtrl_edit, PopupMenu, PopupWindow, RadioBox, RadioButton, SashWindow, ScrolledWindow, SearchCtrl, Slider, SpinButton, SpinCtrl, SpinCtrlDouble, SplitterWindow, StaticBitmap, StaticBox, StaticText, and StatusBar.
- Filter Demos:** A search box with a magnifying glass icon and a close button (X).
- Main Content Area:** A button labeled "Show the Menu sample" and the text "wx.Menu" displayed in a large font.
- Demo Log Messages:** A scrollable log area showing the following messages:

```
18:00:25: OnItemCollapsed: Window Layout
18:00:53: Loading demo Menu.py...
18:00:53: Running demo module...
18:00:59: OnActivate: False
18:00:59: OnAppActivate: False
18:01:13: OnAppActivate: True
18:01:13: OnActivate: True
```
- Status Bar:** Welcome to wxPython 4.0.4

Gui mit wxDemo



Gui mit wxDemo



Gui mit wxDemo

The screenshot shows the wxPython Demo application window. The title bar reads "wxPython: (A Demonstration)". The menu bar includes "File", "Demo", "Options", and "Help".

The left sidebar, titled "wxPython Demos", contains a list of controls with checkboxes. The controls listed are: CommandLinkButton, DVC_CustomRenderer, DVC_DataViewModel, DVC_IndexListModel, DVC_ListCtrl, DVC_TreeCtrl, Gauge, Grid, Grid_MegaExample, GridLabelRenderer, ListBox, ListCtrl, ListCtrl_virtual, ListCtrl_edit, Menu, PopupMenu, PopupWindow, RadioBox, RadioButton, SashWindow, ScrolledWindow, SearchCtrl, Slider, SpinButton, SpinCtrl, SpinCtrlDouble, SplitterWindow, StaticBitmap, StaticBox, StaticText, and StatusBar.

The main area is a code editor with tabs for "Menu Overview", "Demo Code", and "Demo". The "Demo Code" tab is active, showing Python code for creating a menu bar. The code includes comments and function calls for creating menus and submenus. The code is as follows:

```
22 check the source for this sample to see how to implement them.
23 """ , style=wx.TE_READONLY|wx.TE_MULTILINE)
24
25     # Prepare the menu bar
26     menuBar = wx.MenuBar()
27
28     # 1st menu from left
29     menu1 = wx.Menu()
30     menu1.Append(101, "&Mercury", "This the text in the StatusBar")
31     menu1.Append(102, "&Venus", "")
32     menu1.Append(103, "&Earth", "You may select Earth too")
33     menu1.AppendSeparator()
34     menu1.Append(104, "&Close", "Close this frame")
35     # Add menu to the menu bar
36     menuBar.Append(menu1, "&Planets")
37
38     # 2nd menu from left
39     menu2 = wx.Menu()
40     menu2.Append(201, "Hydrogen")
41     menu2.Append(202, "Helium")
42     # a submenu in the 2nd menu
43     submenu = wx.Menu()
44     submenu.Append(2031, "Lanthanium")
45     submenu.Append(2032, "Cerium")
```

Below the code editor is a "Demo Log Messages" window showing the following log entries:

```
18:04:08: OnAppActivate: False
18:04:37: OnAppActivate: True
18:04:40: OnActivate: True
18:04:52: OnActivate: False
18:04:52: OnAppActivate: False
18:05:01: OnAppActivate: True
18:05:01: OnActivate: True
```

At the bottom of the window, there is a "Filter Demos:" section with a search box containing the text "Search". The status bar at the very bottom reads "Welcome to wxPython 4.0.4".

Gui mit wxDemo

Eine einfache Lösung
(keine eigene Panel-Klasse)

Gui mit wxDemo

... (siehe erste Code-Folie)

```
panel = wx.Panel(self, -1)
```

```
### Menu-Abschnitt
```

```
menuBar = wx.MenuBar()    # Prepare the menu bar
```

```
menu1 = wx.Menu()        # 1st menu from left
```

```
menu1.Append(101, "&Mercury", "This the text in the Statusbar")
```

```
menu1.Append(102, "&Venus", "")
```

```
menu1.Append(103, "&Earth", "You may select Earth too")
```

```
menu1.AppendSeparator()
```

```
menu1.Append(104, "&Close", "Close this frame")
```

```
menuBar.Append(menu1, "&Planets")    # Add menu to the menu bar
```

```
self.SetMenuBar(menuBar)
```

```
# Menu events
```

```
## self.Bind(wx.EVT_MENU_HIGHLIGHT_ALL, self.OnMenuHighlight) ## raus!
```

```
self.Bind(wx.EVT_MENU, self.Menu101, id=101)
```

```
self.Bind(wx.EVT_MENU, self.Menu102, id=102)
```

```
self.Bind(wx.EVT_MENU, self.Menu103, id=103)
```

```
self.Bind(wx.EVT_MENU, self.OnCloseWindow, id=104) ## Korrektur
```

```
### Button-Abschnitt
```

... (siehe nächste Code-Folie)

Gui mit wxDemo

... (siehe vorige Code-Folie)

```
### Button-Abschnitt
button = wx.Button(panel, 1003, "Close Me")
button.SetPosition((15, 15))
self.Bind(wx.EVT_BUTTON, self.OnCloseMe, button)
self.Bind(wx.EVT_CLOSE, self.OnCloseWindow)

### Ereignisbehandlung Menu-Abschnitt
def Menu101(self, event):
    ## self.log.write('Welcome to Mercury\n') # anpassen:
    print('Welcome to Mercury\n')

def Menu102(self, event):
    print('Welcome to Venus\n')      ## s.o.

def Menu103(self, event):
    print('Welcome to the Earth\n')  ## s.o.

### Ereignisbehandlung Button-Abschnitt
def OnCloseMe(self, event):
    self.Close(True)

def OnCloseWindow(self, event):
    self.Destroy()
```

... (siehe erste Code-Folie)

Gui mit wxDemo

The screenshot shows the wxPython Demo application window. The title bar reads "wxPython: (A Demonstration)". The menu bar includes "File", "Demo", "Options", and "Help". The main window is divided into several panes:

- Left Pane (wxPython Demos):** A list of various wxPython controls. "TextCtrl" is selected and highlighted in blue.
- Top Pane (Tabs):** Three tabs are visible: "TextCtrl Overview", "Demo Code", and "Demo".
- Right Pane (Demo Area):** This pane displays the functionality of the selected control. It includes:
 - wx.TextCtrl:** A single-line text box containing the text "Test it out and see".
 - Password:** An empty password input field.
 - Multi-line:** A multi-line text area containing two paragraphs of text. To its right are three buttons: "Test Replace", "Test GetSelection", and "Test WriteText".
 - Rich Text:** A rich text control showing text with color (the word "red" is in red) and font style (the word "different" is in a blue, italicized font).
 - Test Positions:** A text area showing the same text "0123456789" repeated five times, likely demonstrating character positions.
- Bottom Pane (Demo Log Messages):** A log window showing the following messages:

```
18:11:49: OnActivate: True
18:12:13: Loading demo TextCtrl.py...
18:12:13: Running demo module...
18:12:18: OnActivate: False
18:12:18: OnAppActivate: False
18:12:31: OnAppActivate: True
18:12:31: OnActivate: True
```
- Bottom Left:** A "Filter Demos:" section with a search input field.
- Bottom Bar:** A status bar that says "Welcome to wxPython 4.0.4".

wx.TextCtrl

Gui mit wxDemo

The screenshot shows the wxPython Demo application window. The title bar reads "wxPython: (A Demonstration)". The menu bar includes "File", "Demo", "Options", and "Help".

wxPython Demos

- ListBox
- ListCtrl
- ListCtrl_virtual
- ListCtrl_edit
- Menu
- PopupMenu
- PopupWindow
- RadioBox
- RadioButton
- SashWindow
- ScrolledWindow
- SearchCtrl
- Slider
- SpinButton
- SpinCtrl
- SpinCtrlDouble
- SplitterWindow
- StaticBitmap
- StaticBox
- StaticText
- StatusBar
- StockButtons
- TextCtrl
- ToggleButton
- ToolBar
- TreeCtrl
- Validator
- "Book" Controls
- Custom Controls
- Advanced Generic Widgets
- More Windows/Controls

TextCtrl Overview | **Demo Code** | **Demo**

Active Version: Original Modified

```
13     # print("OnKillFocus")
14     # evt.Skip()
15     # def OnWindowDestroy(self, evt):
16     #     print("OnWindowDestroy")
17     #     evt.Skip()
18
19
20     def __init__(self, parent, log):
21         wx.Panel.__init__(self, parent, -1)
22         self.log = log
23
24         ll = wx.StaticText(self, -1, "wx.TextCtrl")
25         t1 = wx.TextCtrl(self, -1, "Test it out and see", size=(125, -1))
26         wx.CallAfter(t1.SetInsertionPoint, 0)
27         self.tcl = t1
28
29
30         self.Bind(wx.EVT_TEXT, self.EvtText, t1)
31         t1.Bind(wx.EVT_CHAR, self.EvtChar)
32         # t1.Bind(wx.EVT_SET_FOCUS, self.OnSetFocus)
33         # t1.Bind(wx.EVT_KILL_FOCUS, self.OnKillFocus)
34         # t1.Bind(wx.EVT_WINDOW_DESTROY, self.OnWindowDestroy)
35
36         12 = wx.StaticText(self, -1, "Password")
37         t2 = wx.TextCtrl(self, -1, "", size=(125, -1), style=wx.TE_PASSWORD)
```

Demo Log Messages

```
18:12:34: OnAppActivate: False
18:13:09: OnAppActivate: True
18:13:09: OnActivate: True
18:13:41: OnActivate: False
18:13:41: OnAppActivate: False
18:13:50: OnAppActivate: True
18:13:50: OnActivate: True
```

Filter Demos:

Welcome to wxPython 4.0.4

Gui mit wxDemo

... (zusätzlich im Konstruktor)

```
### Label (StaticText) und TextCtrl
self.label1 = wx.StaticText(self, -1, "Ein/Ausgabe")
self.label1.SetPosition((155, 15))
self.textCtrl1 = wx.TextCtrl(self, -1, "noch ohne Inhalt", size=(205, -1))
self.textCtrl1.SetPosition((155, 45))
self.Bind(wx.EVT_TEXT, self.EvtText, self.textCtrl1)    # sinnvoll?
```

... (zusätzlich in Ereignisbehandlung)

```
### Ereignisbehandlung
### TextCtrl
def EvtText(self, event):                ## (so noch nicht sinnvoll)
    print('EvtText')
```

... (siehe vorige Code-Folie)