KORISNIČKA SUČELJA

Aleksandar Maksimović
IRB

uvod u wxPython

- wxPython
  - OS
    - MS Windows (Windows 98 i novije)
    - Unix, Linux + gtk (Gnome Toolkit)
    - Mac OS X 10.2.3
  - Python ver 2.3
  - wxPython toolkit - više verzija
  - tekst editor

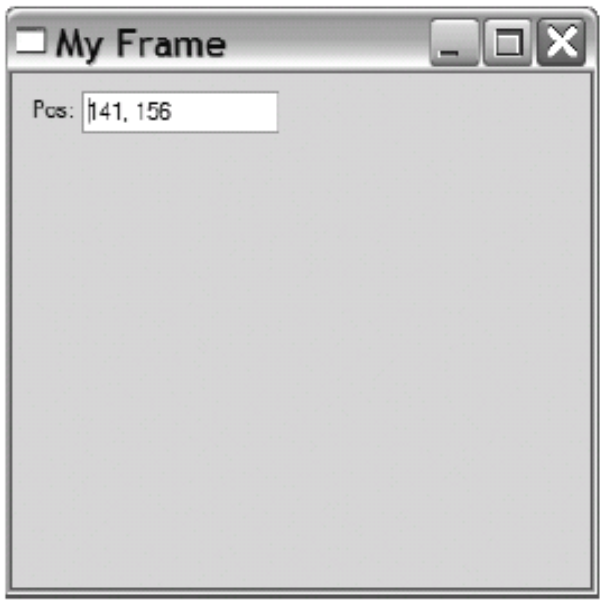
uvod wxpython

sample.py

vjezbe7

```
#!/bin/env python
import wx
class MyFrame(wx.Frame):

    def __init__(self):
        wx.Frame.__init__(self, None, -1, "My Frame", size=(300, 300))
        panel = wx.Panel(self, -1)
        panel.Bind(wx.EVT_MOTION,  self.OnMove)
        wx.StaticText(panel, -1, "Pos:", pos=(10, 12))
        self.posCtrl = wx.TextCtrl(panel, -1, "", pos=(40, 10))

    def OnMove(self, event):
        pos = event.GetPosition()
        self.posCtrl.SetValue("%s, %s" % (pos.x, pos.y))

if __name__ == '__main__':
    app = wx.PySimpleApp()
    frame = MyFrame()
    frame.Show(True)
    app.MainLoop()
```

uvod wxpython



sample.py

Label - StaticText

Entry - TextCtrl
Tkinter ⟶ wxPython

wx.Frame.__init__ - wx konstruktor

wx.Panel

wx.EVT_MOTION - događaj

uvod

Figure 1.2
Running `hello.py`
on Windows

Figure 1.3
Running `hello.py`
on Linux

minimalni wxpy program

bare.py

```
import wx

class App(wx.App):

    def OnInit(self):
        frame = wx.Frame(parent=None, title='Bare')
        frame.Show()
        return True

app = App()
app.MainLoop()
```

Provjerimo da li wxpython radi.

Program napravi prazan frame (okvir) i prikaže ga.

minimalni wxpy program

Sve linije koda u primjeru su neophodne.

Ilustrira 5 koraka potrebnih za svaki wxPython program:

1. Import wxPython paket
2. Naslijedi wx.App klasu (subklasa)
3. Definiraj konstruktor (__init__)
4. Kreiraj instancu klase (App())
5. Napravi glavnu petlju (Mainloop())

1. uvod

1. Importiranje wxPythona (modul wx)

   import wx

2. Koristimo wx klase, funkcije ili konstante stavljanjem imena wx kao prefiks

   class App(wx.App):

3. wx moramo importirati prije svih ostalih klasa, funkcija iz wxPythona

1. uvod

- Stari stil (NE KORISTIMO)
  - from wxPython import wx # DEPRECATED
  - from wxPython.wx import * # DON'T DO THIS ANY MORE

Ako ne importiramo prvo wx neke klase neće dobro raditi, npr. xrc

```
import wx             # Always import wx before
from wx import xrc    # any other wxPython packages,
from wx import html   # just to be on the safe side.
```

1. uvod

- Ostale pakete i dalje importiramo kako želimo

```
import sys
import wx
import os
from wx import xrc
import urllib
```

2. uvod

- wxPython program mora imati
  - 1 objekt aplikacije (wx.App), mora biti instanca wx.App ili subklasa (nasljedi wx.App) koja definira metodu OnInit(). Metodu OnInit() koristi wx.App prilikom kreiranja objekta.
  - 1 ili više frame objekata wx.Frame

Subklasa:

```
class MyApp(wx.App):

    def OnInit(self):
        frame = wx.Frame(parent=None, id=-1, title="Bare")
        frame.Show()
        return True
```

Show - prikazuje ili skriva Frame (prozor)

3. Konstruktor

- Nismo definirali konstruktor
  - kad __init__ metoda nije definirana Python automatski zove konstruktor od klase iznad (roditelja) wx.App.__init__()
  - ako definiramo konstruktor, moramo zvati konstruktor od wx.App klase

```
class App(wx.App):

    def __init__(self):
        # Call the base class constructor.
        wx.App.__init__(self)
        # Do something here...
```

4 i 5 Aplikacija i petlja

- Konačni korak je stvaranje instance aplikacije i pozivanje metode MainLoop()

```
app = App()
app.MainLoop()
```

wxPython preuzima kontrolu i odgovara na događaje.

spare.py

```
#!/usr/bin/env python    ❶

"""Spare.py is a starting point for a wxPython program."""    ❷

import wx

class Frame(wx.Frame):    ❸
    pass

class App(wx.App):

    def OnInit(self):
        self.frame = Frame(parent=None, title='Spare')    ❹
        self.frame.Show()
        self.SetTopWindow(self.frame)    ❺
        return True

if __name__ == '__main__':    ❻
    app = App()
    app.MainLoop()
```

uvod

1. unix OS poziva interpreter, inače je komentar
2. docstring - opisuje program

```
>>> import spare
>>> print spare.__doc__
Spare.py is a starting point for simple wxPython programs.
>>>
```

3. promjenili smo stvaranje Frame objekta, sada je Frame subklasa od wx.Frame klase
4. varijabla .frame sadrži instancu Frame

uvod

5. SetTopWindow() metoda postavlja self.frame kao "glavni" prozor. Metoda nasljeđena iz wx.App klase.

6. Ako je modul "glavni" program izvrši linije

```
if __name__ == '__main__':
    app = App()
    app.MainLoop()
```

hello.py

```
#!/usr/bin/env python
```
❶ Shebang

```
"""Hello, wxPython! program."""
```
⟵ Docstring describes the code

```
import wx
```
⟵ Import the wxPackage

❷ wx.Frame subclass

```
class Frame(wx.Frame):
    """Frame class that displays an image."""

    def __init__(self, image, parent=None, id=-1,
                 pos=wx.DefaultPosition,
                 title='Hello, wxPython!'):
        """Create a Frame instance and display image."""
        temp = image.ConvertToBitmap()
        size = temp.GetWidth(), temp.GetHeight()
        wx.Frame.__init__(self, parent, id, title, pos, size)
        self.bmp = wx.StaticBitmap(parent=self, bitmap=temp)
```
❸ Image parameter

❹ Displaying the image

hello.py

```
class App(wx.App):
    """Application class."""

    def OnInit(self):
        image = wx.Image('wxPython.jpg', wx.BITMAP_TYPE_JPEG)
        self.frame = Frame(image)

        self.frame.Show()
        self.SetTopWindow(self.frame)
        return True

def main():
    app = App()
    app.MainLoop()

if __name__ == '__main__':
    main()
```

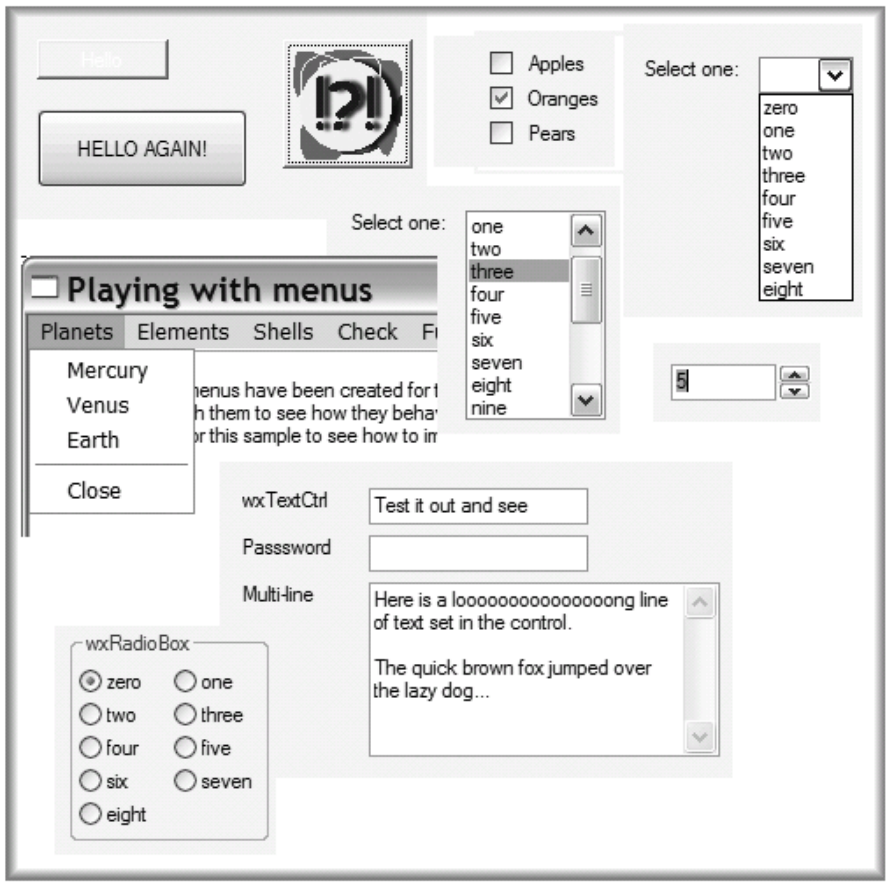
5 wx.App subclass

Image handling 6

7 main() function

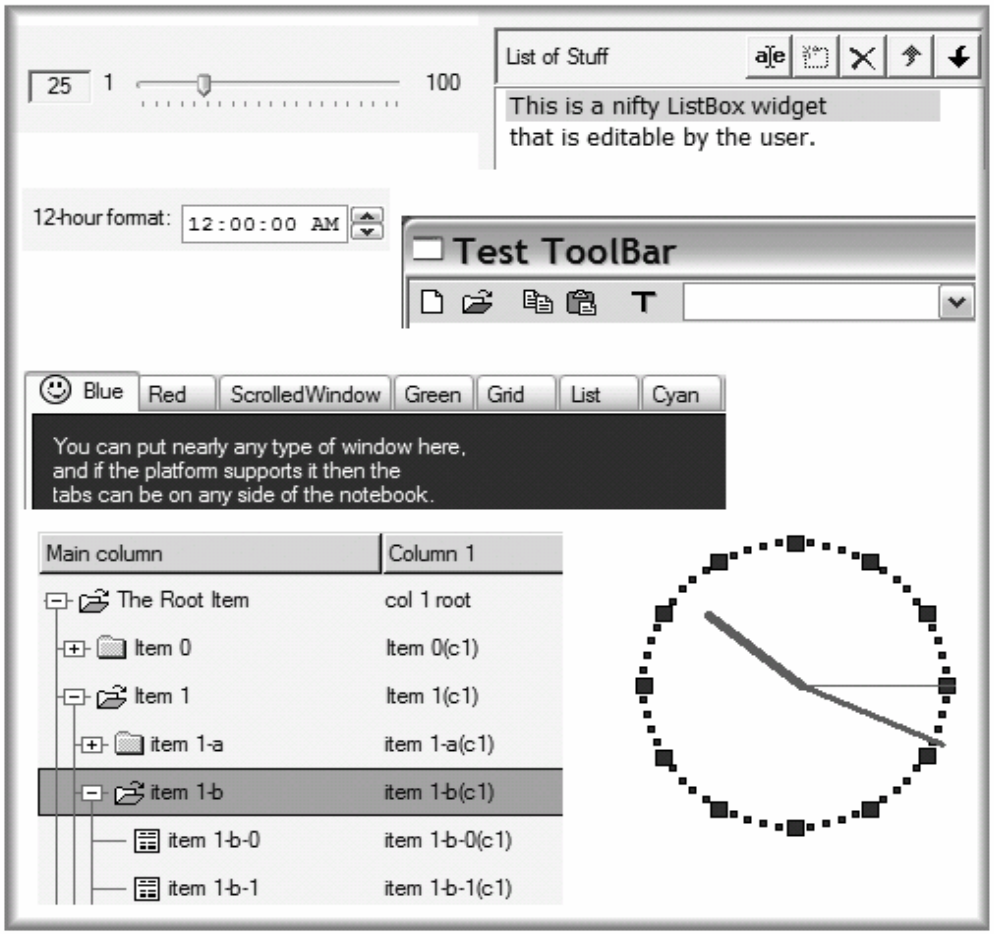
8 Import vs. execute

Mogućnosti wxPythona



osnovni widgeti

Mogućnosti wxPythona

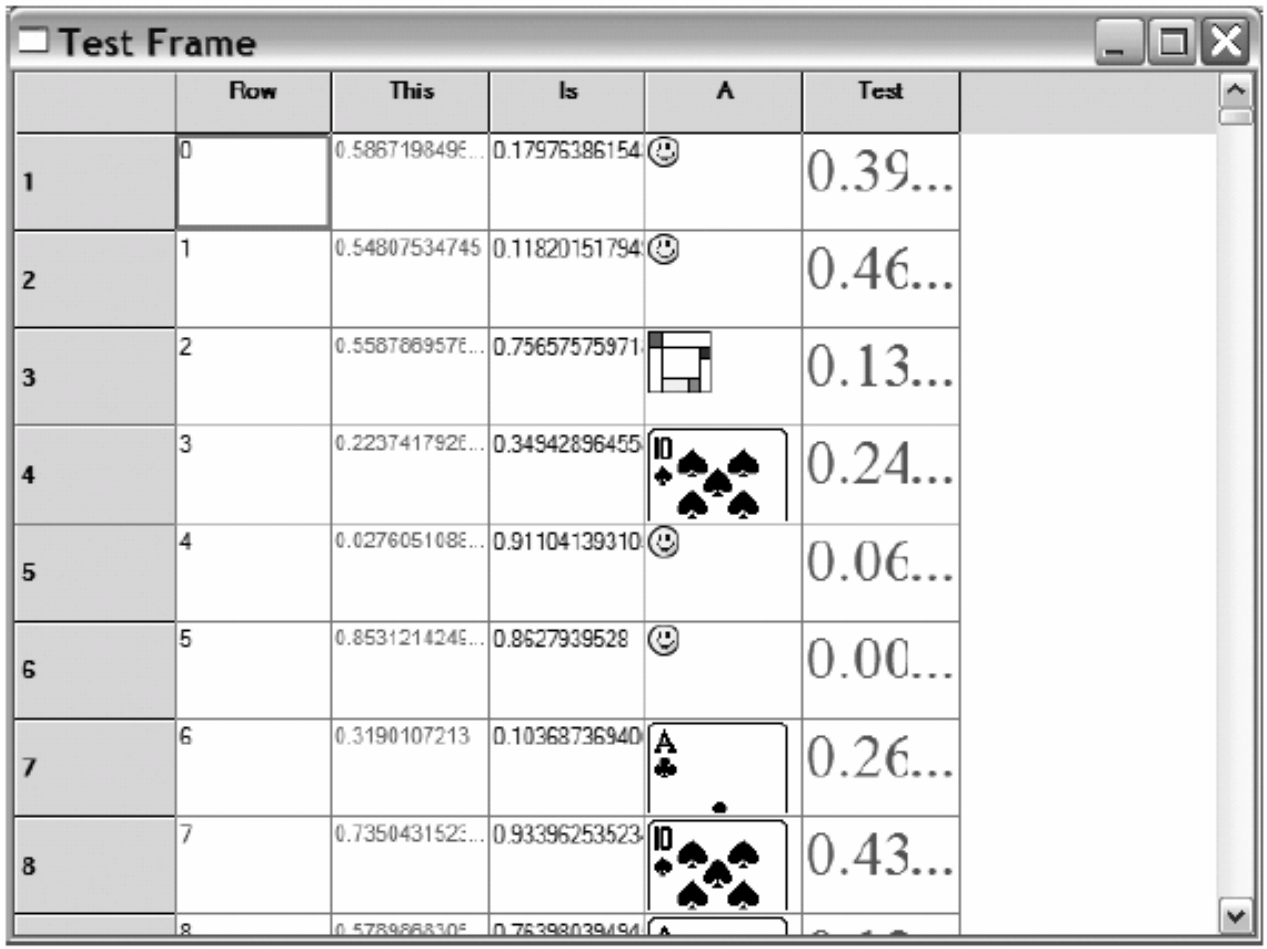


naprednije kontrole

tree list

analogni sat

Mogućnosti wxPythona

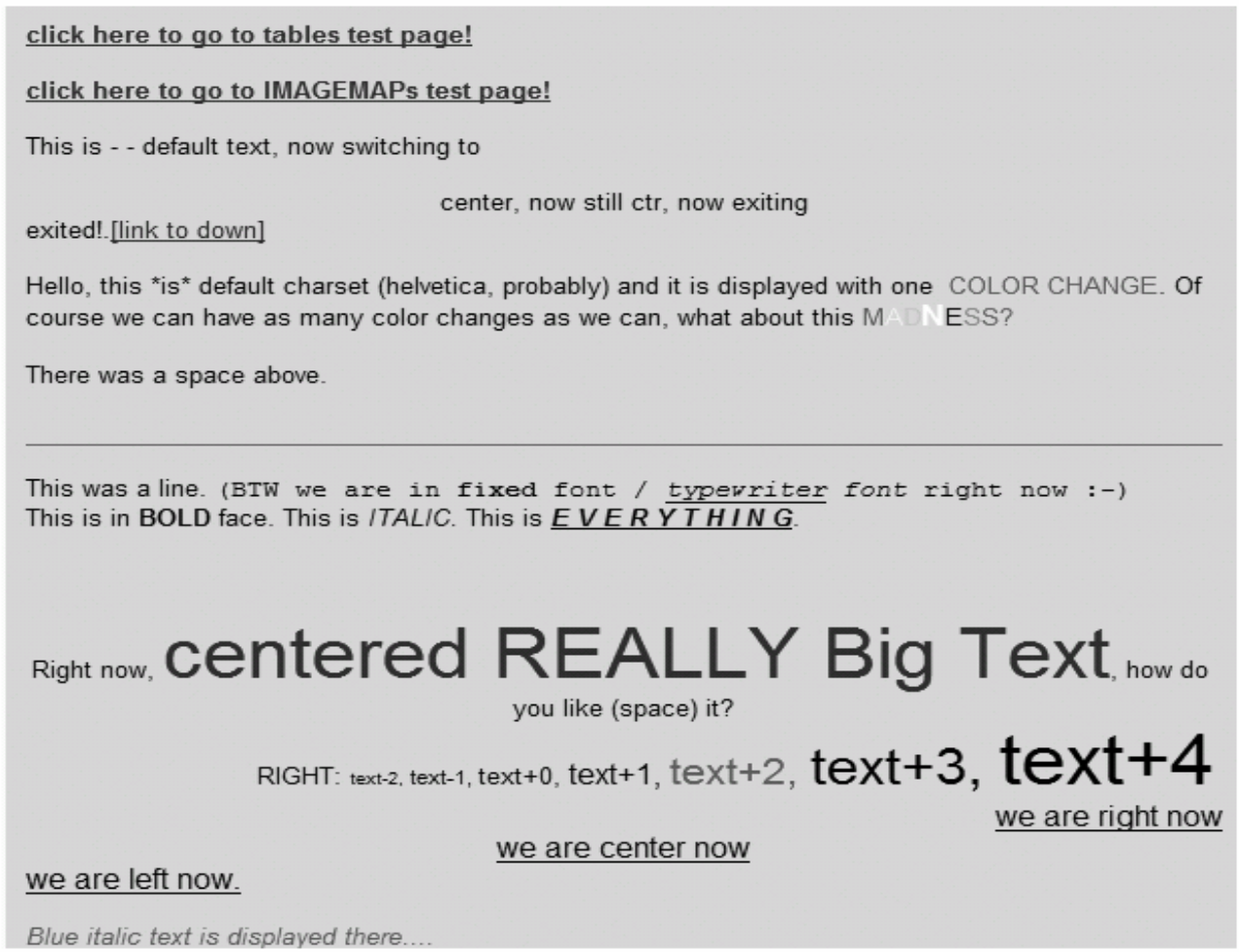


grid

prikazivanje ćelija

s odabranim pozadinama

Mogućnosti wxPythona



HTML mogućnosti

wx.HTMLwindow

Hello world program

```
import wx

class MyApp(wx.App):

    def OnInit(self):
        frame = MyFrame("Hello World", (50, 60), (450, 340))
        frame.Show()
        self.SetTopWindow(frame)
        return True

class MyFrame(wx.Frame):

    def __init__(self, title, pos, size):
        wx.Frame.__init__(self, None, -1, title, pos, size)
        menuFile = wx.Menu()
        menuFile.Append(1, "&About...")
        menuFile.AppendSeparator()
        menuFile.Append(2, "E&xit")
        menuBar = wx.MenuBar()
        menuBar.Append(menuFile, "&File")
        self.SetMenuBar(menuBar)
```

FRAME

IZBORNIK

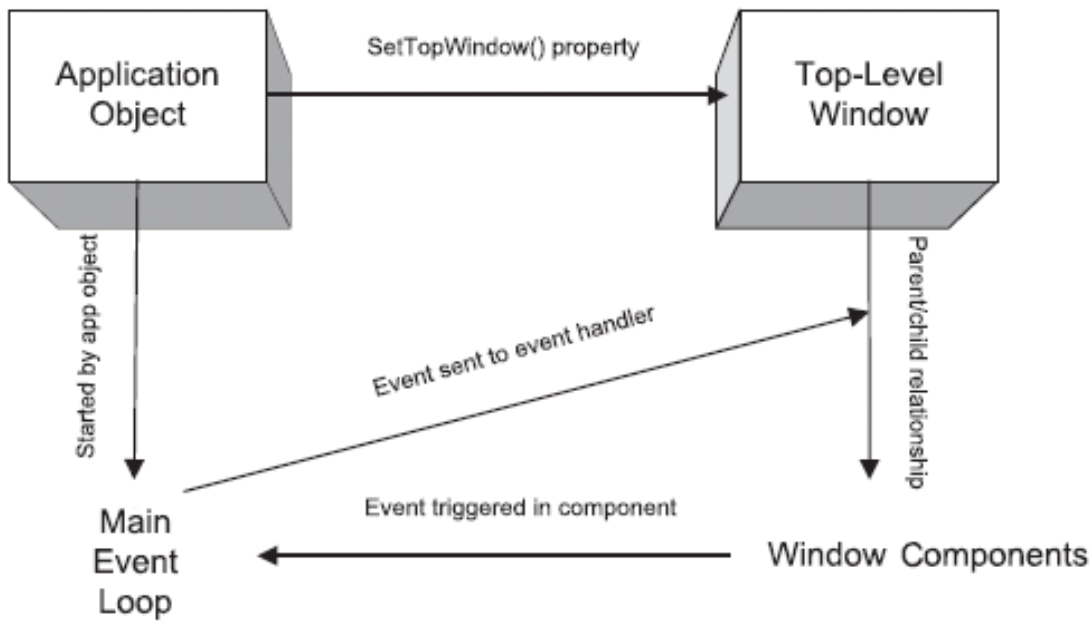
```
        self.CreateStatusBar()
        self.SetStatusText("Welcome to wxPython!")
        self.Bind(wx.EVT_MENU, self.OnAbout, id=1)
        self.Bind(wx.EVT_MENU, self.OnQuit, id=2)

    def OnQuit(self, event):
        self.Close()

    def OnAbout(self, event):
        wx.MessageBox("This is a wxPython Hello world sample",
                      "About Hello World", wx.OK | wx.ICON_INFORMATION, self)


if __name__ == '__main__':
    app = MyApp(False)
    app.MainLoop()
```

Traka statusna

Dijalog

wxPython aplikacija

- application object - objekt aplikacije iz wx.App
  - poziva glavnu petlju
  - odziv na događaje koji inače nisu napravljeni
  - sadrži glavni prozor i glavnu petlju



objekt aplikacije

1. Definira se subklasa
2. Napisati metodu OnInit() u subklasi
3. U glavnom dijelu programa napraviti instancu klase
4. Pozvati MainLoop() metodu koja prenosi kontrolu programa na wxPython

Metoda OnInit() je dio wxPythona, koristimo za sve potrebne postavke (inicijalizacije), a ne u __init__ metodi (konstruktoru). Ako koristimo konstruktor moramo pozvati konstruktor od objekta aplikacije

```
wx.App.__init__(self)
```

U OnInit() napravimo tipično barem 1 Frame objekt

wx.App subklasa

- Kada možemo izostaviti subklasu od wx.App? Obično radimo subklasu kako bi mogli definirati Frame u OnInit()
  - kada imamo samo jedan Frame, objekt aplikacije je trivijalan
  - koristimo wx.PySimpleApp klasu definiranu u wxPythonu.

```
class PySimpleApp(wx.App):

    def __init__(self, redirect=False, filename=None,
                 useBestVisual=False, clearSigInt=True):
        wx.App.__init__(self, redirect, filename, useBestVisual,
                        clearSigInt)

    def OnInit(self):
        return True
```

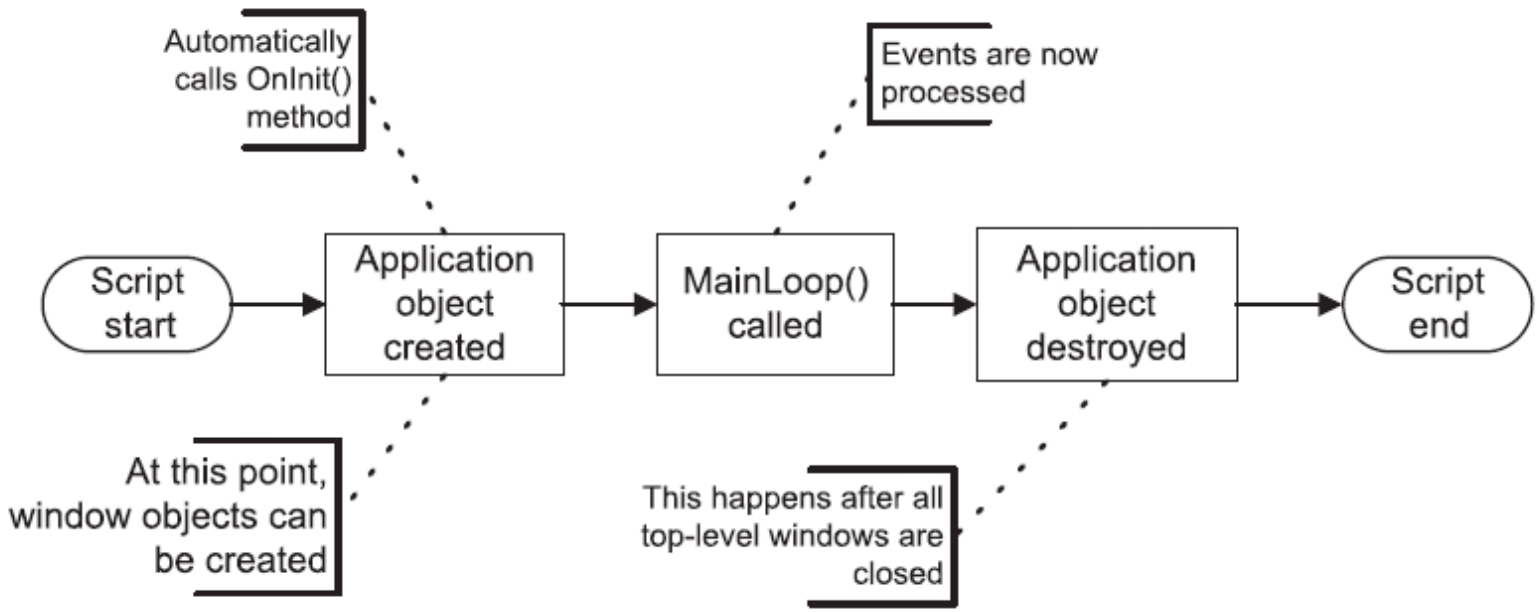
wx.PySimpleApp primjena

Klasu PySimpleApp jednostavno koristimo

```
if __name__ == '__main__':
    app = wx.PySimpleApp()
    frame = MyNewFrame(None)
    frame.Show(True)
    app.MainLoop()
```

Život objekta u wxPythonu

Zatvaranjem prozora završava MainLoop(), ne mora se podudarati s programom.



redirekcija u wxPythonu

```
#!/usr/bin/env python

import wx
import sys

class Frame(wx.Frame):

    def __init__(self, parent, id, title):
        print "Frame __init__"
        wx.Frame.__init__(self, parent, id, title)

class App(wx.App):

    def __init__(self, redirect=True, filename=None):
        print "App __init__"
        wx.App.__init__(self, redirect, filename)
```

startup.py

Koristi sys.stdout i sys.stderr, standardni izlazi za poruke i pogreške.

wxPython pod MS Windows kontrolira ove izlaze i zamjenjuje ih prozorom.

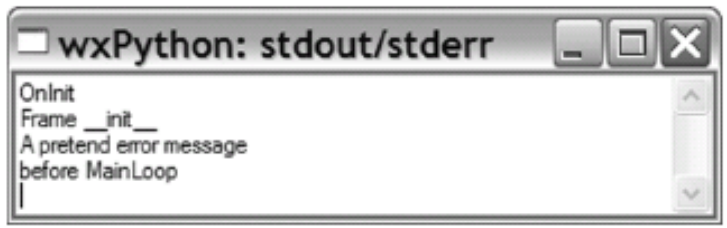
redirekcija u wxPythonu

```
    def OnInit(self):
        print "OnInit"
        self.frame = Frame(parent=None, id=-1, title='Startup')
        self.frame.Show()
        self.SetTopWindow(self.frame)
        print >> sys.stderr, "A pretend error message"
        return True

    def OnExit(self):
        print "OnExit"


if __name__ == '__main__':
    app = App(redirect=True)
    print "before MainLoop"
    app.MainLoop()
    print "after MainLoop"
```

Writing to stdout

Creating the frame

Writing to stderr



❶ Text redirection starts here

❷ The main event loop is entered here


```
if __name__ == '__main__':
    app = App(redirect=True)
    print "before MainLoop"
    app.MainLoop()
    print "after MainLoop"
```

Glavni prozor

Korisnik vidi program kao "glavni prozor" - top-level window

Glavni prozor obično dobivamo iz wx.Frame ili wx.Dialog klasa

Postoji veliki broj već definiranih dijaloga u wx.Dialog klasi

"top-level" prozor je bilo koji widget bez roditelja

Samo jedan je "glavni prozor" - pomoću metode SetTopWindow()

Default: prvi Frame u wx.App postaje "glavni" prozor

wx.Frame=prozor

- GUI korisnik vidi Frame kao prozor
- wx.Frame je roditelj svih Frame objekata u wxPythonu
- Subklasa od wx.Frame koja ima __init__ metodu mora zvati konstruktor koji ima opcije

```
wx.Frame(parent, id=-1, title="", pos=wx.DefaultPosition,
        size=wx.DefaultSize, style=wx.DEFAULT_FRAME_STYLE,
        name="frame")
```

To su parametri koje možemo poslati konstruktoru wx.Frame.__init__()

Frame parametri

Parameter	Description
parent	The parent window of the frame being created. For top-level windows, the value is `None`. If another window is used for the parent parameter then the new frame will be owned by that window and will be destroyed when the parent is. Depending on the platform, the new frame may be constrained to only appear on top of the parent window. In the case of a child MDI window, the new window is restricted and can only be moved and resized within the parent.
id	The wxPython ID number for the new window. You can pass one in explicitly, or pass `-1` which causes wxPython to automatically generate a new ID. See the section "Working with wxPython ID" for more information.
title	The window title—for most styles, it's displayed in the window title bar.
pos	A `wx.Point` object specifying where on the screen the upper left-hand corner of the new window should be. As is typical in graphics applications, the (0, 0) point is the upper left corner of the monitor. The default is (-1, -1), which causes the underlying system to decide where the window goes. See the section "Working with wx.Size and wx.Point" for more information.
size	A `wx.Size` object specifying the starting size of the window. The default is (-1, -1), which causes the underlying system to determine the starting size. See the section "Working with wx.Size and wx.Point" for more information.
style	A bitmask of constants determining the style of the window. You may use the bitwise or operator (`|`) to combine them when you want more than one to be in effect. See the section "Working with wx.Frame styles" for usage guidelines.
name	An internal name given to the frame, used on Motif to set resource values. Can also be used to find the window by name later.

Frame

Id prozora, cijeli broj koji mora biti jedinstven u programu

NewId() generira id.

```
id = wx.NewId()
frame = wx.Frame.__init__(None, id)
```

Ne zanima nas id

```
frame = wx.Frame.__init__(None, -1)
id = frame.GetId()
```

Klase wx.Point i wx.Size.

(0,0) je default

eksplicitno definiramo veličinu i položaj.

Dinamička promjena položaja

```
point = wx.Point(10, 12)

x = point.x
y = point.y
```

```
frame = wx.Frame(None, -1, pos=(10, 10), size=(100, 100))
```

```
frame.SetPosition((2, 3))
```

Stil Frame objekta

wx.DEFAULT_FRAME_STYLE BITMASKE

```
wx.MAXIMIZE_BOX | wx.MINIMIZE_BOX | wx.RESIZE_BORDER |
wx.SYSTEM_MENU | wx.CAPTION | wx.CLOSE_BOX
```

Default stil modificiran tako da se ne može promjeniti veličina prozora

```
wx.DEFAULT_FRAME_STYLE ^ (wx.RESIZE_BORDER | wx.MINIMIZE_BOX |
wx.MAXIMIZE_BOX)
```

Style	Description
wx.CAPTION	Adds a title bar on the frame, which displays the frame's `Title` property.
wx.CLOSE_BOX	Instructs the system to display a close box on the frame's title bar, using the system defaults for placement and style. Also enables the close item on the system menu if applicable.

stilovi

wx.DEFAULT_FRAME_STYLE	As you might expect from the name, this is the default if no style is specified. It is defined as `wx.MAXIMIZE_BOX | wx.MINIMIZE_BOX | wx.RESIZE_BORDER | wx.SYSTEM_MENU | wx.CAPTION | wx.CLOSE_BOX`.
wx.FRAME_SHAPED	Frames created with this style can use the `SetShape()` method to create a window with a non-rectangular shape.
wx.FRAME_TOOL_WINDOW	Makes the frame look like a toolbox window by giving it a smaller titlebar than normal. Under Windows a frame created with this style does not show in the taskbar listing of all open windows.
wx.MAXIMIZE_BOX	Adds a maximize box on the frame, using the system parameters for the look and placement of the box. Also enables maximize functionality in the system menu if applicable.
wx.MINIMIZE_BOX	Adds a minimize box on the frame, using the system parameters for the look and placement of the box. Also enables minimize functionality in the system menu if applicable.
wx.RESIZE_BORDER	Adds a resizable border to the frame.
wx.SIMPLE_BORDER	A plain border without decoration. May not work on all platforms.
wx.SYSTEM_MENU	Adds the system menu (with close, move, resize, etc. functionality, using system look and feel) and the close box to the window. The availability of resize and close operations within this menu depends on the styles `wx.MAXIMIZE_BOX`, `wx.MINIMIZE_BOX` and `wx.CLOSE_BOX` being chosen.

primjeri

Figure 2.4 A frame created with the default style

Figure 2.5 A frame created to be non-resizable. Notice the lack of minimize/maximize buttons.

Figure 2.6 A toolbar frame, with a smaller title bar and no system menu

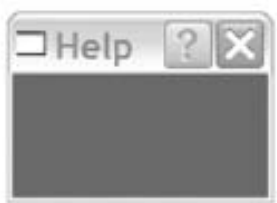
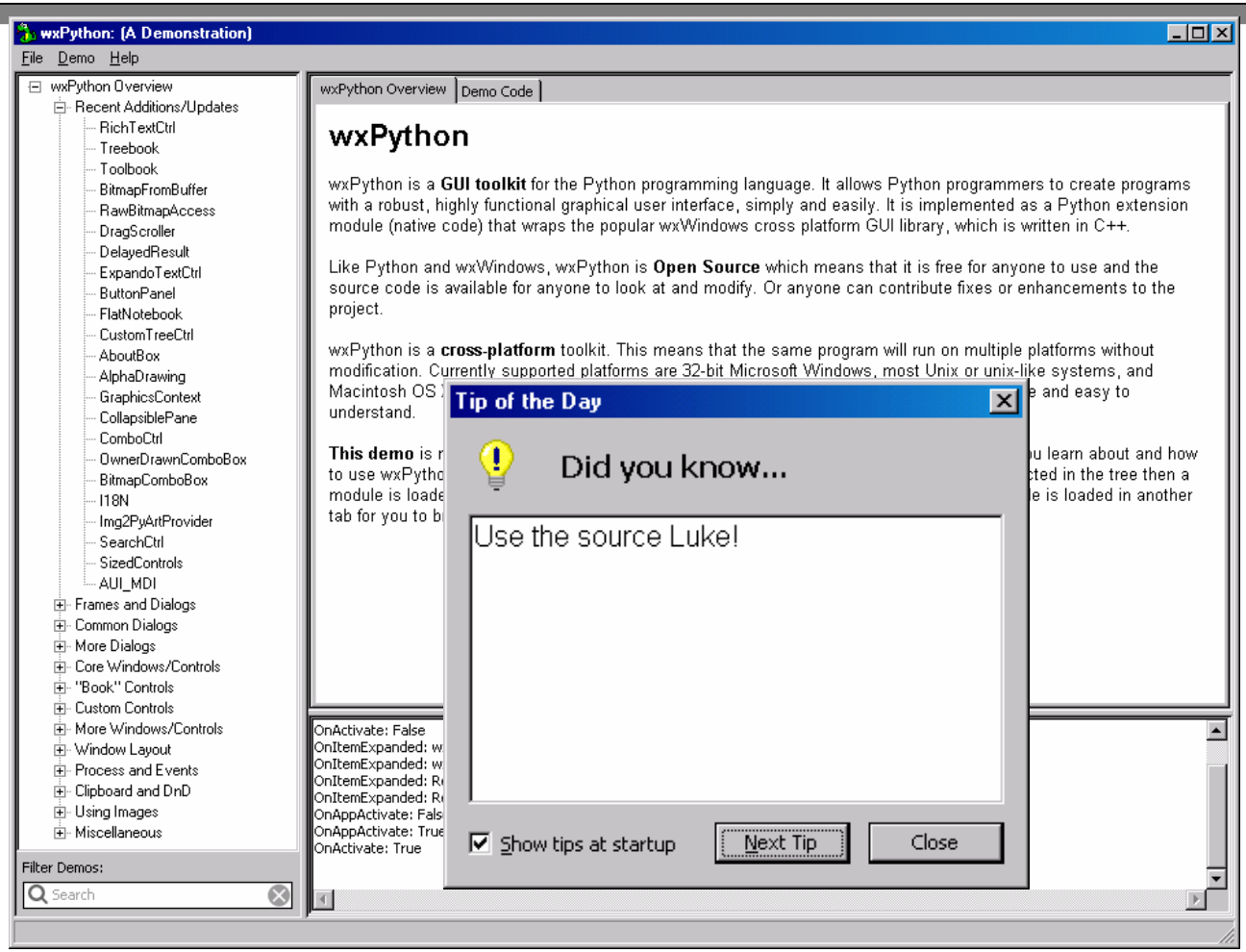


Figure 2.7 A frame with a help button

zadatak

- Prostudiraj primjere sample.py i spare.py. Napiši aplikaciju koja će imati subklasu Xapp od wx.App i subklasu Xframe od wx.Frame. U klasi Xapp inicijaliziraj Xframe. U klasi Xframe napravi statično polje "Prezime", zatim TextCtrl u kojem se upisuje prezime. Ispiši naredbom print prezime (koristi GetValue() metodu)
- Postavi ova dva polja jedno ispod drugog.
- Pokreni demo.py demo program od wxpythona
- Kako dobijemo help u python interpreteru za widget wx.TextCtrl. Da li je ova informacije korisna?

demo wxpython



demo.py

direktorij s

dokmentima od

wxpythona

Dodavanje objekata i prozora

```
#!/usr/bin/env python

import wx

class InsertFrame(wx.Frame):

    def __init__(self, parent, id):
        wx.Frame.__init__(self, parent, id, 'Frame With Button',
                size=(300, 100))
        panel = wx.Panel(self)
        button = wx.Button(panel, label="Close", pos=(125, 10),
                size=(50, 50))
        self.Bind(wx.EVT_BUTTON, self.OnCloseMe, button)
        self.Bind(wx.EVT_CLOSE, self.OnCloseWindow)

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

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

❶ Creating the panel

❷ Adding the button to the panel

❸ Binding the button click event

❹ Binding the window close event

Izbornik, alatna traka,...

```
#!/usr/bin/env python

import wx
import images

class ToolbarFrame(wx.Frame):

    def __init__(self, parent, id):
        wx.Frame.__init__(self, parent, id, 'Toolbars',
                          size=(300, 200))
        panel = wx.Panel(self)
        panel.SetBackgroundColour('White')
        statusBar = self.CreateStatusBar()
        toolbar = self.CreateToolBar()
        toolbar.AddSimpleTool(wx.NewId(), images.getNewBitmap(),
                              "New", "Long help for 'New'")
        toolbar.Realize()
        menuBar = wx.MenuBar()
```

1 Creating the status bar

2 Creating the toolbar

3 Adding a tool to the bar

Creating a menubar

4 Preparing the toolbar for display

Izbornik, alatna traka,...

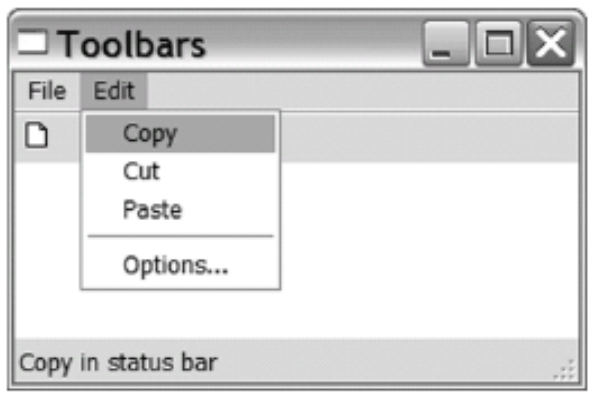
```
        menu1 = wx.Menu()
        menuBar.Append(menu1, "&File")
        menu2 = wx.Menu()
        menu2.Append(wx.NewId(), "&Copy", "Copy in status bar")
        menu2.Append(wx.NewId(), "C&ut", "")
        menu2.Append(wx.NewId(), "Paste", "")
        menu2.AppendSeparator()
        menu2.Append(wx.NewId(), "&Options...", "Display Options")
        menuBar.Append(menu2, "&Edit")
        self.SetMenuBar(menuBar)

if __name__ == '__main__':
    app = wx.PySimpleApp()
    frame = ToolbarFrame(parent=None, id=-1)
    frame.Show()
    app.MainLoop()
```

5 Creating two individual menus

6 Creating individual menu items

Attaching the menu to the menubar

Attaching the menubar to the frame



Dijalog

YES/NO dijalog

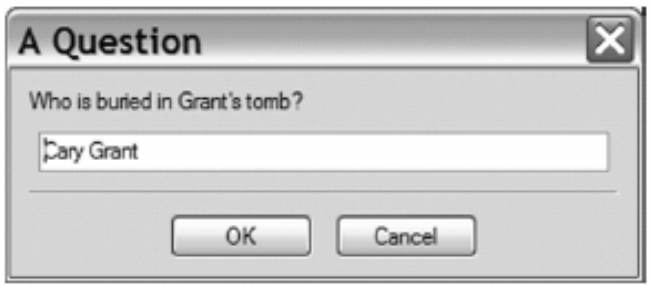
```
dlg = wx.MessageDialog(None, 'Is this the coolest thing ever!',
                       'MessageDialog', wx.YES_NO | wx.ICON_QUESTION)
result = dlg.ShowModal()
dlg.Destroy()
```

konstrukcija dijaloga

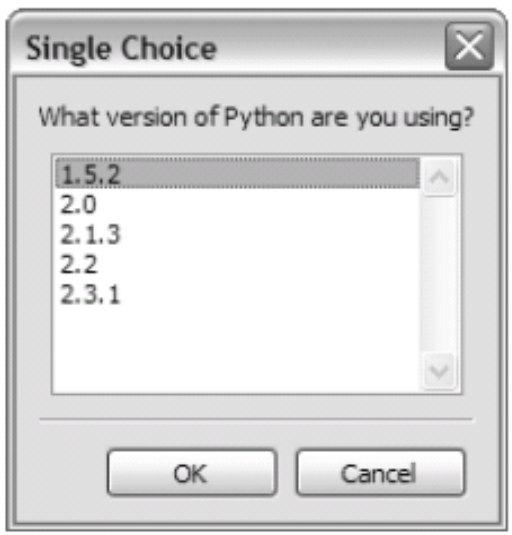
```
wx.MessageDialog(parent, message,
        caption="Message box",
        style=wx.OK | wx.CANCEL,
        pos=wx.DefaultPosition)
```

Entry i lista

```
dlg = wx.TextEntryDialog(None, "Who is buried in Grant's tomb?",
        'A Question', 'Cary Grant')
if dlg.ShowModal() == wx.ID_OK:
    response = dlg.GetValue()
```




```
dlg = wx.SingleChoiceDialog(None,
        'What version of Python are you using?',
        'Single Choice',
        ['1.5.2', '2.0', '2.1.3', '2.2', '2.3.1'],
if dlg.ShowModal() == wx.ID_OK:
    response = dlg.GetStringSelection()
```



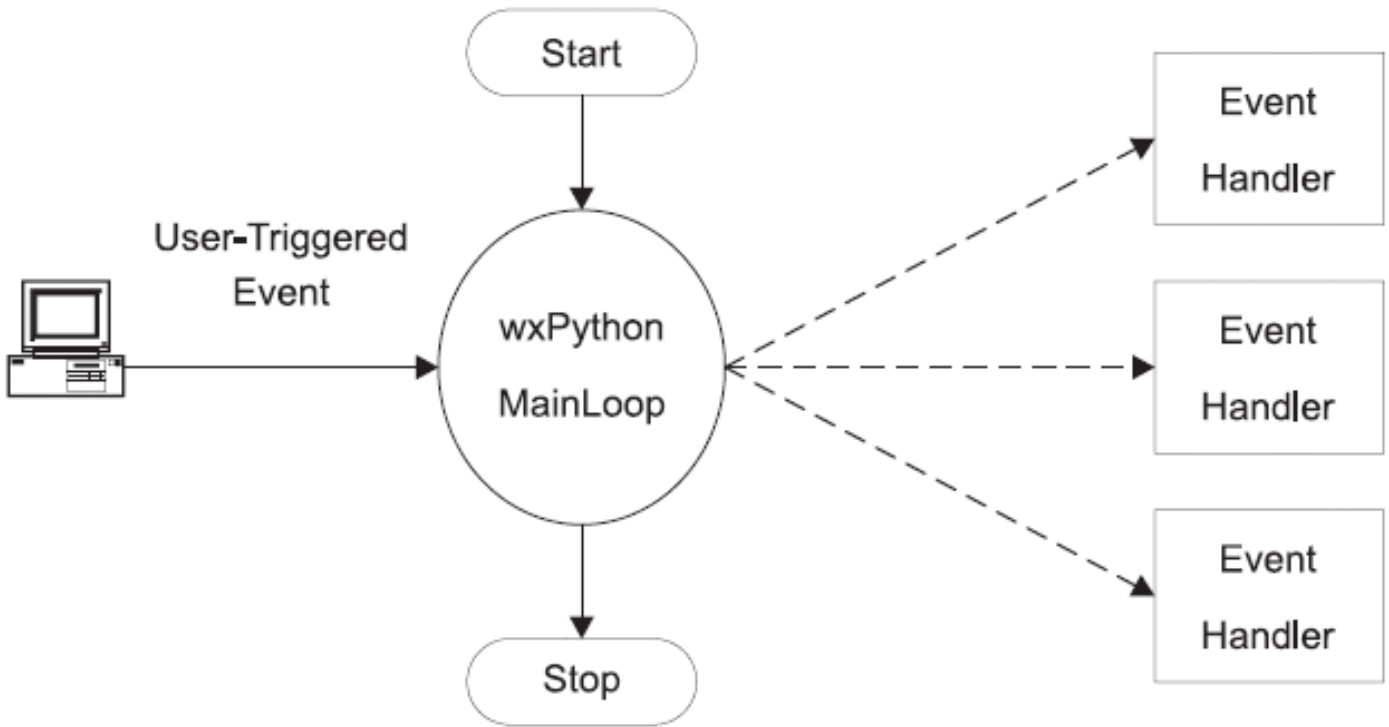
DOGAĐAJI

Term	Definition
event	Something that happens during your application that requires a response.
event object	The concrete representation of an event in wxPython including data attributes that encapsulate the specifics of the event. Events are represented as instances of the `wx.Event` class and its subclasses, such as `wx.CommandEvent` and `wx.MouseEvent`.
event type	An integer ID that wxPython adds to every event object. The event type gives further information about the nature of the event. For example, the event type of a `wx.MouseEvent` indicates whether the event is a mouse click or a mouse move.
event source	Any wxPython object that creates events. Examples are buttons, menu items, list boxes, or any other widget.
event-driven	A program structure where the bulk of time is spent waiting for, or responding to, events.

DOGAĐAJI

Term	Definition
event queue	A continuously maintained list of events that have already occurred, but have not yet been processed.
event handler	A written function or method that is called in response to an event. Also called a *handler function* or *handler method*.
event binder	A wxPython object that encapsulates the relationship between a specific widget, a specific event type, and an event handler. In order to be invoked, all event handlers must be registered with an event binder.
`wx.EvtHandler`	A wxPython class that allows its instances to create a binding between an event binder of a specific type, an event source, and an event handler. Note that the class `wx.EvtHandler` is not the same thing as an event handler function or method defined previously.

DOGAĐAJI


Start
User-Triggered
Event
wxPython
MainLoop
Stop
Event
Handler
Event
Handler
Event
Handler

subklasa događaja

Event	Description
wx.CloseEvent	Triggered when a frame closes. The event type distinguishes between a normal frame closing and a system shutdown event.
wx.KeyEvent	A key press event. The event types distinguish between key down, key up, and complete key press.
wx.MouseEvent	A mouse event. The event types distinguish between a mouse move and a mouse click. There are separate event types depending on which button is clicked and whether it's a single or double click.
wx.PaintEvent	Triggered when a window's contents need to be redrawn.
wx.SizeEvent	This event is triggered when a window is resized, and typically results in a change to the window layout.
wx.TimerEvent	Can be created by the `wx.Timer` class, which allows periodic events.

miš i događaji

wx.MouseEvent sadrži događaje
ukupno 14 događaja

```
wx.EVT_LEFT_DOWN
wx.EVT_LEFT_UP
wx.EVT_LEFT_DCLICK
wx.EVT_MIDDLE_DOWN
wx.EVT_MIDDLE_UP
wx.EVT_MIDDLE_DCLICK
wx.EVT_RIGHT_DOWN
wx.EVT_RIGHT_UP
wx.EVT_RIGHT_DCLICK
```

sve događaje miša možemo vezati na
jedan događaj wx.EVT_MOUSE_EVENTS

wx.EVT_MOTION - položaj kursora u widgetu
wx.ENTER_WINDOW - kursor ulazi u prozor
wx.LEAVE_WINDOW - kursor napušta prozor

wx.CommandEvent - sadrži 28 događaja, većina ih je vezana za specifične

widgete, npr gumb preko wx.EVT_BUTTON ili izbornik wx.EVT_MENU

bind

```
self.Bind(wx.EVT_BUTTON, self.OnClick, button)
```

funkcija Bind povezuje događaj (wx.EVT_BUTTON) u objektu (button) s metodom (OnClick()). Sintaksa metode Bind

```
Bind(event, handler, source=None, id=wx.ID_ANY, id2=wx.ID_ANY)
```

Dodatna pomoć

- Interaktivno u interpreteru: npr. help("wx.MouseEvent")
  - izlista mnogo informacija uglavnom za C++ sintaksu
- wxwidgets reference - npr. preko search funkcije u helpu tražimo wxMouseEvent
- demo.py - primjeri za određene funkcije i događaje

metode za događaje

Table 3.3 Commonly used methods of `wx.EvtHandler`

Method	Description
AddPendingEvent(event)	Places the event argument into the event processing system. Similar to `ProcessEvent()`, but it does not actually trigger immediate processing of the event. Instead, the event is added to the event queue. Useful for event-based communication between threads.
Bind(event, handler, source=None, id=wx.ID_ANY, id2=wx.ID_ANY)	See full description in section 3.3.1.
GetEvtHandlerEnabled() SetEvtHandlerEnabled(boolean)	The property is `True` if the handler is currently processing events, `False` if otherwise.
ProcessEvent(event)	Puts the event object into the event processing system for immediate handling.

izbornik

menu_event.py

```
#!/usr/bin/env python

import wx

class MenuEventFrame(wx.Frame):

    def __init__(self, parent, id):
        wx.Frame.__init__(self, parent, id, 'Menus',
             size=(300, 200))
        menuBar = wx.MenuBar()
        menu1 = wx.Menu()
        menuItem = menu1.Append(-1, "&Exit...")
        menuBar.Append(menu1, "&File")
        self.SetMenuBar(menuBar)
        self.Bind(wx.EVT_MENU, self.OnCloseMe, menuItem)

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

if __name__ == '__main__':
    app = wx.PySimpleApp()
    frame = MenuEventFrame(parent=None, id=-1)
    frame.Show()
    app.MainLoop()
```

miš

mouse_event.py


```
import wx

class MouseEventFrame(wx.Frame):

    def __init__(self, parent, id):
        wx.Frame.__init__(self, parent, id, 'Frame With Button',
                size=(300, 100))
        self.panel = wx.Panel(self)
        self.button = wx.Button(self.panel,
            label="Not Over", pos=(100, 15))
        self.Bind(wx.EVT_BUTTON, self.OnButtonClick,
            self.button)
        self.button.Bind(wx.EVT_ENTER_WINDOW,
            self.OnEnterWindow)
        self.button.Bind(wx.EVT_LEAVE_WINDOW,
            self.OnLeaveWindow)

    def OnButtonClick(self, event):
        self.panel.SetBackgroundColour('Green')
        self.panel.Refresh()
```

1 Binding the button event

2 Binding the mouse enter event

3 Binding the mouse leave event

nastavak

```
def OnEnterWindow(self, event):
    self.button.SetLabel("Over Me!")
    event.Skip()

def OnLeaveWindow(self, event):
    self.button.SetLabel("Not Over")
    event.Skip()


 if __name__ == '__main__':
     app = wx.PySimpleApp()
     frame = MouseEventFrame(parent=None, id=-1)
     frame.Show()
     app.MainLoop()
```

tekst

static_text.py

```
class StaticTextFrame(wx.Frame):
    def __init__(self):
        wx.Frame.__init__(self, None, -1, 'Static Text Example',
                size=(400, 300))
        panel = wx.Panel(self, -1)
        wx.StaticText(panel, -1, "This is an example of static text",
                (100, 10))
        rev = wx.StaticText(panel, -1,
                "Static Text With Reversed Colors",
                (100, 30))
        rev.SetForegroundColour('white')
        rev.SetBackgroundColour('black')
        center = wx.StaticText(panel, -1,
                "align center", (100, 50),
                (160, -1), wx.ALIGN_CENTER)
        center.SetForegroundColour('white')
        center.SetBackgroundColour('black')
        right = wx.StaticText(panel, -1,
                "align right", (100, 70),
                (160, -1), wx.ALIGN_RIGHT)
        right.SetForegroundColour('white')
```

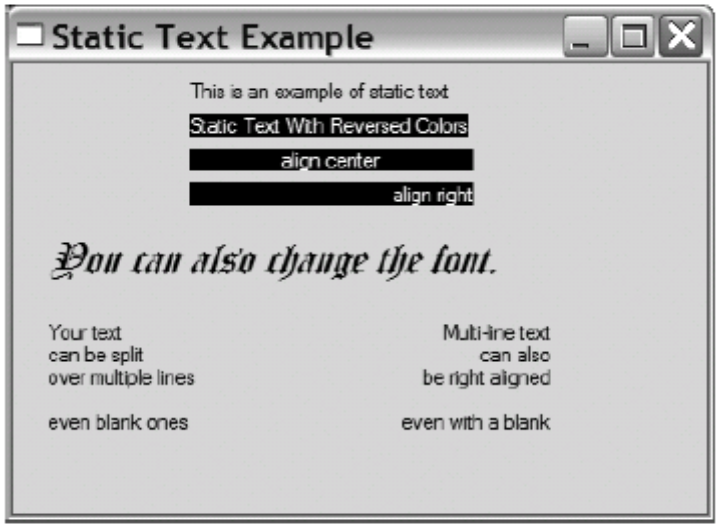
Viewing basic static text

Designating reversed colors

Designating center aligned

Designating right aligned

tekst




```
right.SetBackgroundColour('black')
str = "You can also change the font."
text = wx.StaticText(panel, -1, str, (20, 100))
font = wx.Font(18, wx.DECORATIVE,
       wx.ITALIC, wx.NORMAL)
text.SetFont(font)
wx.StaticText(panel, -1,
        "Your text\ncan be split\n"
        "over multiple lines\n\neven blank ones", (20,150))
wx.StaticText(panel, -1,
        "Multi-line text\ncan also\n"
        "be right aligned\n\neven with a blank", (220,150),
        style=wx.ALIGN_RIGHT)
```

Displaying multi-lines

Displaying aligned multi-lines

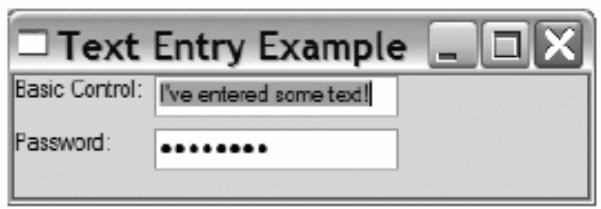
wx.StaticText

Parameter	Purpose
parent	The containing widget
id	The wxPython identifier. To automatically create a unique identifier, use -1
label	Contains the text that you want to display in the static control.
pos	The position of the widget as a wx.Point object or a Python tuple
size	The size of the widget as a wx.Size object or a Python tuple
style	The style flag
name	Name used for finding the object

wx.TextCtrl

```
class TextFrame(wx.Frame):

    def __init__(self):
        wx.Frame.__init__(self, None, -1, 'Text Entry Example',
                size=(300, 100))
        panel = wx.Panel(self, -1)
        basicLabel = wx.StaticText(panel, -1, "Basic Control:")
        basicText = wx.TextCtrl(panel, -1, "I've entered some text!",
                size=(175, -1))
        basicText.SetInsertionPoint(0)

        pwdLabel = wx.StaticText(panel, -1, "Password:")
        pwdText = wx.TextCtrl(panel, -1, "password", size=(175, -1),
                style=wx.TE_PASSWORD)
        sizer = wx.FlexGridSizer(cols=2, hgap=6, vgap=6)
        sizer.AddMany([basicLabel, basicText, pwdLabel, pwdText])
        panel.SetSizer(sizer)
```

text_ctrl.py



stil

wx.TE_CENTER	The text is centered within the control.
wx.TE_LEFT	The text is left justified within the control. This is the default behavior.
wx.TE_NOHIDESEL	The name of this option parses to “no hide sel,” in case you were having trouble decoding it. It’s a Windows option to override a default behavior of the Windows text widget, namely that it doesn’t highlight the selected text unless the widget has focus. With this option selected, the widget will always highlight the text. Has no effect on other systems.
wx.TE_PASSWORD	The text entered will not be displayed, but instead masked by asterisks.
wx.TE_PROCESS_ENTER	If this bit is specified, a text enter event is triggered when the user presses the enter key within the control. Otherwise, the keypress is managed internally by either the text control or the dialog.
wx.TE_PROCESS_TAB	If this bit is specified, a normal character event will be created for a tab key pressed (generally meaning a tab will be inserted into the text). If not specified, then the tab will be managed by the dialog, usually for keyboard navigation between controls.
wx.TE_READONLY	The text control is read-only, and cannot be modified by user input.
wx.TE_RIGHT	The text is right-justified within the control.

metode

AppendText(text)	Appends the text argument to the end of the text in the control. The insertion point also moves to the end of the control.
Clear()	Resets the text value of the control to "". Also generates a text updated event.
EmulateKeyPress(event)	Given a keypress event, inserts into the control the character associated with the event, just as if the actual keypress had occured.
GetInsertionPoint() SetInsertionPoint(pos) SetInsertionPointEnd()	The position is the integer index of the current insertion point, or to put it another way, the index where the next inserted character would be placed. The beginning of the control is `0`.
GetRange(from, to)	Returns the string between the given integer positions of the control.
GetSelection() GetStringSelection() SetSelection(from, to)	`GetSelection()` returns a tuple (start, end) with the indexes of the currently selected text. `GetStringSelection()` returns the string contents of that range. The setter takes the integer endpoints of the range.
GetValue() SetValue(value)	`SetValue()` changes the entire value of the control. The getter returns the entire string.
Remove(from, to)	Removes the given range from the text.
Replace(from, to, value)	Replaces the given range with new value. This can change the length of the text.
WriteText(text)	Similar to `AppendText()` except that the new text is placed at the current insertion point.

tekst-više linija, stil

```
class TextFrame(wx.Frame):

    def __init__(self):
        wx.Frame.__init__(self, None, -1, 'Text Entry Example',
                          size=(300, 250))
        panel = wx.Panel(self, -1)
        multiLabel = wx.StaticText(panel, -1, "Multi-line")
        multiText = wx.TextCtrl(panel, -1,
                "Here is a looooooooooooooong line "
                "of text set in the control.\n\n"
                "See that it wrapped, and that "
                "this line is after a blank",
                size=(200, 100), style=wx.TE_MULTILINE)
        multiText.SetInsertionPoint(0)

        richLabel = wx.StaticText(panel, -1, "Rich Text")
        richText = wx.TextCtrl(panel, -1,
                "If supported by the native control, "
                "this is reversed, and this is a different font.",
                size=(200, 100),
                style=wx.TE_MULTILINE|wx.TE_RICH2)
        richText.SetInsertionPoint(0)
        richText.SetStyle(44, 52, wx.TextAttr("white", "black"))
        points = richText.GetFont().GetPointSize()
        f = wx.Font(points + 3, wx.ROMAN,
            wx.ITALIC, wx.BOLD, True)
        richText.SetStyle(68, 82, wx.TextAttr("blue",
            wx.NullColour, f))
        sizer = wx.FlexGridSizer(cols=2, hgap=6, vgap=6)
```

Creating a text control

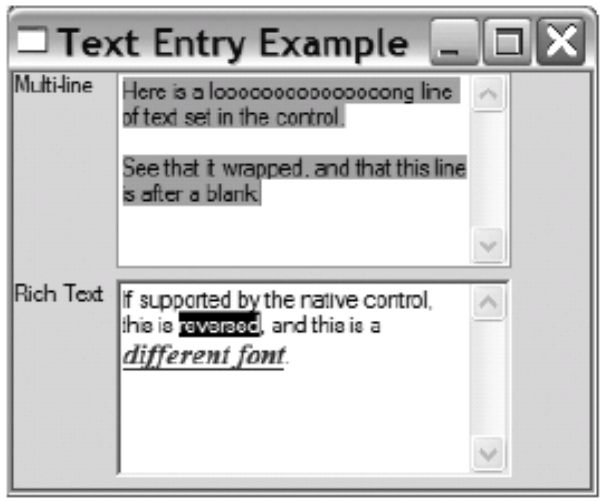
Setting the cursor point

Creating a rich text control

Setting text styles

Creating a font

Setting a style in the new font



text_ctrl_multiple.py

tekst-više linija, stil

```
        sizer.AddMany([multiLabel, multiText, richLabel, richText])
        panel.SetSizer(sizer)

if __name__ == '__main__':
    app = wx.PySimpleApp()
    frame = TextFrame()
    frame.Show()
    app.MainLoop()
```

Metode

GetNumberOfLines()	Returns the number of lines in the control. For a single-line control, returns `1`.
IsMultiLine() IsSingleLine()	Boolean methods for determining state of the control
PositionToXY(pos)	Given an integer position within the text, returns a tuple with the (col, row) index of the position. The column and row indexes both start at 0.
SetStyle(start, end, style)	Immediately changes the style for the given range of text.
GetLineLength(lineNo)	Returns the integer length of the given line.
GetLineText(lineNo)	Returns the text of the given line

font

```
wx.Font(pointSize, family, style, weight, underline=False,
    faceName="", encoding=wx.FONTENCODING_DEFAULT)
```

Font	Description
wx.DECORATIVE	A formal, old-English style font
wx.DEFAULT	The system default font
wx.MODERN	A monospace (fixed-pitch) font
wx.ROMAN	A serif font, generally something like Times New Roman
wx.SCRIPT	*A handwriting or cursive font*
wx.SWISS	A sans-serif font, generally something like Helvetica or Arial

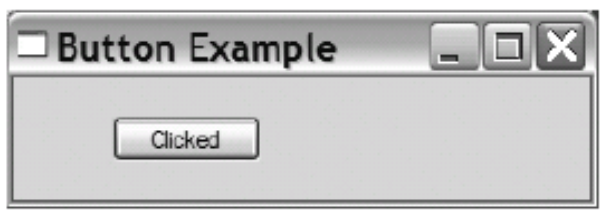
```
e = wx.FontEnumerator()
e.EnumerateFacenames()
fontList = e.GetFacenames()
```

svi fontovi u listi

gumbi

```
import wx

class ButtonFrame(wx.Frame):
    def __init__(self):
        wx.Frame.__init__(self, None, -1, 'Button Example',
                          size=(300, 100))
        panel = wx.Panel(self, -1)
        self.button = wx.Button(panel, -1, "Hello", pos=(50, 20))
        self.Bind(wx.EVT_BUTTON, self.OnClick, self.button)
        self.button.SetDefault()

    def OnClick(self, event):
        self.button.SetLabel("Clicked")

if __name__ == '__main__':
    app = wx.PySimpleApp()
    frame = ButtonFrame()
    frame.Show()
    app.MainLoop()
```

Button.py



gumb sa slikom

```
import wx

class BitmapButtonFrame(wx.Frame):
    def __init__(self):
        wx.Frame.__init__(self, None, -1, 'Bitmap Button Example',
                size=(200, 150))
        panel = wx.Panel(self, -1)
        bmp = wx.Image("bitmap.bmp", wx.BITMAP_TYPE_BMP).ConvertToBitmap()
        self.button = wx.BitmapButton(panel, -1, bmp, pos=(10, 20))
        self.Bind(wx.EVT_BUTTON, self.OnClick, self.button)
        self.button.SetDefault()
        self.button2 = wx.BitmapButton(panel, -1, bmp, pos=(100, 20),
            style=0)
        self.Bind(ex.EVT_BUTTON, self.OnClick, self.button2)

    def OnClick(self, event):
        self.Destroy()

if __name__ == '__main__':
    app = wx.PySimpleApp()
    frame = BitmapButtonFrame()
    frame.Show()
    app.MainLoop()
```

bitmap_button.py

check box

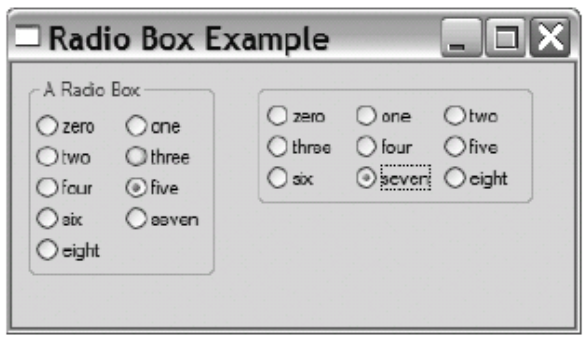
```
import wx

class CheckBoxFrame(wx.Frame):
    def __init__(self):
        wx.Frame.__init__(self, None, -1, 'Checkbox Example',
                size=(150, 200))
        panel = wx.Panel(self, -1)

        wx.CheckBox(panel, -1, "Alpha", (35, 40), (150, 20))
        wx.CheckBox(panel, -1, "Beta", (35, 60), (150, 20))
        wx.CheckBox(panel, -1, "Gamma", (35, 80), (150, 20))

if __name__ == '__main__':
    app = wx.PySimpleApp()
    CheckBoxFrame().Show()
    app.MainLoop()
```

Checkbox.py

radio box

```
class RadioBoxFrame(wx.Frame):
    def __init__(self):
        wx.Frame.__init__(self, None, -1, 'Radio Box Example',
                size=(350, 200))
        panel = wx.Panel(self, -1)
        sampleList = ['zero', 'one', 'two', 'three', 'four', 'five',
                      'six', 'seven', 'eight']
        wx.RadioBox(panel, -1, "A Radio Box", (10, 10), wx.DefaultSize,
                    sampleList, 2, wx.RA_SPECIFY_COLS)

        wx.RadioBox(panel, -1, "", (150, 10), wx.DefaultSize,
                    sampleList, 3, wx.RA_SPECIFY_COLS)

if __name__ == '__main__':
    app = wx.PySimpleApp()
    RadioBoxFrame().Show()
    app.MainLoop()
```

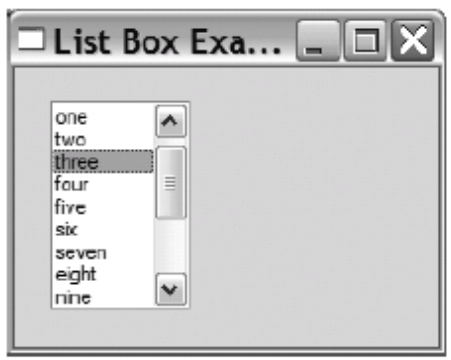
radio_box.py



list box

```
class ListBoxFrame(wx.Frame):
    def __init__(self):
        wx.Frame.__init__(self, None, -1, 'List Box Example',
                size=(250, 200))
        panel = wx.Panel(self, -1)

        sampleList = ['zero', 'one', 'two', 'three', 'four', 'five',
                      'six', 'seven', 'eight', 'nine', 'ten', 'eleven',
                      'twelve', 'thirteen', 'fourteen']

        listBox = wx.ListBox(panel, -1, (20, 20), (80, 120), sampleList,
                wx.LB_SINGLE)
        listBox.SetSelection(3)
```

List_box.py



stil

stil za list box

Style	Description
wx.LB_EXTENDED	The user can select a range of multiple items by using a mouse shift-click, or the keyboard equivalent.
wx.LB_MULTIPLE	The user can have more than one item selected at a time. Essentially, in this case, the list box acts like a group of checkboxes.
wx.LB_SINGLE	The user can have only one item selected at a time. Essentially, in this case, the list box acts like a group of radio buttons.

stil za scroll bar

Style	Description
wx.LB_ALWAYS_SB	The list box will always display a vertical scroll bar, whether or not it is needed.
wx.LB_HSCROLL	If the native widget supports it, the list box will create a horizontal scrollbar if items are too wide to fit.
wx.LB_NEEDED_SB	The list box will only display a vertical scroll bar if needed. This is the default.

metode

Method	Description
Append(item)	Appends the string item to the end of the list.
Clear()	Empties the list box.
Delete(n)	Removes the item at index `n` from the list.
Deselect(n)	In a multiple select list box, causes the item at position `n` to be deselected. No effect in other styles.
FindString(string)	Returns the integer position of the given string, or `-1` if not found.
GetCount()	Returns the number of strings in the list.
GetSelection() SetSelection(n, select) GetStringSelection() SetStringSelection(string, select) GetSelections()	Get selection returns the integer index currently selected (single list only). For a multiple list, use `GetSelections()`, which returns a tuple of integer positions. For a single list, `GetStringSelection()` returns the string at the selected index. The set methods set the given position or string to the state specified by the Boolean argument. Changing the selection in this way does not trigger the `EVT_LISTBOX` event.

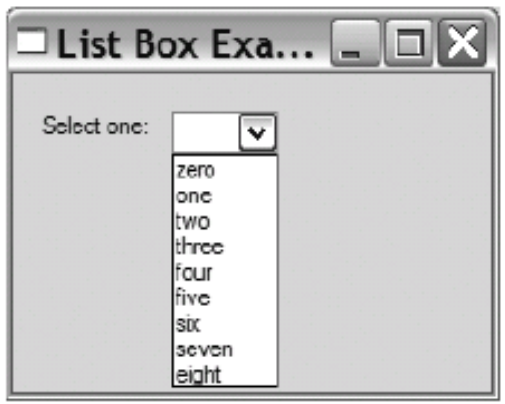
metode

Method	Description
GetString(n) SetString(n, string)	Gets or sets the string at position `n`.
InsertItems(items, pos)	Inserts the list of strings in the `items` argument into the list box before the position in the `pos` argument. A `pos` of `0` puts the items at the beginning of the list.
Selected(n)	Returns a Boolean corresponding to the selected state of the item at index `n`.
Set(choices)	Resets the list box to the list given in `choices`—that is, the current elements are removed from the list and replaced by the new list.

list box i selekcija

```
import wx

class ChoiceFrame(wx.Frame):
    def __init__(self):
        wx.Frame.__init__(self, None, -1, 'Choice Example',
                size=(250, 200))
        panel = wx.Panel(self, -1)
        sampleList = ['zero', 'one', 'two', 'three', 'four', 'five',
                      'six', 'seven', 'eight']
        wx.StaticText(panel, -1, "Select one:", (15, 20))
        wx.Choice(panel, -1, (85, 18), choices=sampleList)
```

Choice.py

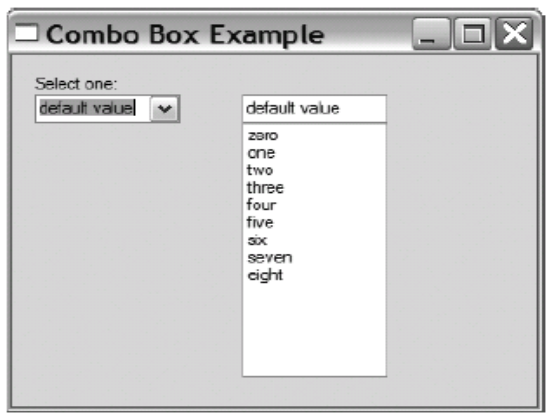


combo box

```
class ComboBoxFrame(wx.Frame):
    def __init__(self):
        wx.Frame.__init__(self, None, -1, 'Combo Box Example',
                size=(350, 300))
        panel = wx.Panel(self, -1)
        sampleList = ['zero', 'one', 'two', 'three', 'four', 'five',
                      'six', 'seven', 'eight']
        wx.StaticText(panel, -1, "Select one:", (15, 15))
        wx.ComboBox(panel, -1, "default value", (15, 30), wx.DefaultSize,
                    sampleList, wx.CB_DROPDOWN)
        wx.ComboBox(panel, -1, "default value", (150, 30), wx.DefaultSize,
                    sampleList, wx.CB_SIMPLE)
```

combo_box.py

povezuje tekst (entry) i listu



tel. imenik

phbk1.py

Phones.py

```
class App(wx.App):
    def OnInit(self):
        self.frame = Frame(parent=None, title='Phonebook',id=-1)
        self.frame.Show()
        self.SetTopWindow(self.frame)
        return True
if __name__ == '__main__':
    app = App()
    app.MainLoop()
```

```
class Frame(wx.Frame):
    #pass
    def __init__(self, parent,id,title):
        wx.Frame.__init__(self, parent,id,title,size=(350,200))
        self.panel = wx.Panel(self, -1)
        panel=self.panel
        panel.SetBackgroundColour("White")
```

imenik

```
l1 = wx.StaticText(self.panel, -1, "Name",pos=(10,4))

self.vname = wx.TextCtrl(panel, -1, "", size=(125, -1),pos=(80,1))

.....

self.vphone = wx.TextCtrl(panel, -1, "", size=(125, -1),pos=(80,21))

b1 = wx.Button(panel,-1," Add  ",pos=(10,44))

self.Bind(wx.EVT_BUTTON, self.addEntry, b1)

.....

b4 = wx.Button(panel,-1," Load ",pos=(220,44))

.....

self.Bind(wx.EVT_BUTTON, self.loadEntry, b4)

self.select=wx.ListBox(panel,-1,pos=(10,74),style=wx.LB_SINGLE)
```

```
print "done inserting"

self.setSelect()
```

imenik metode

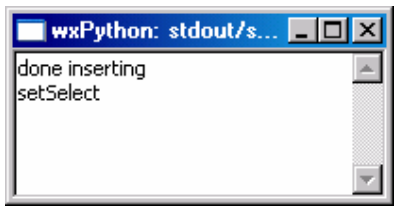
```
def OnCloseMe(self, event):
        self.Close(True)
def whichSelected (self) :
        print "At %s of %d" % (self.select.GetSelection(), len(phonelist))
        return int(self.select.GetSelection())
```

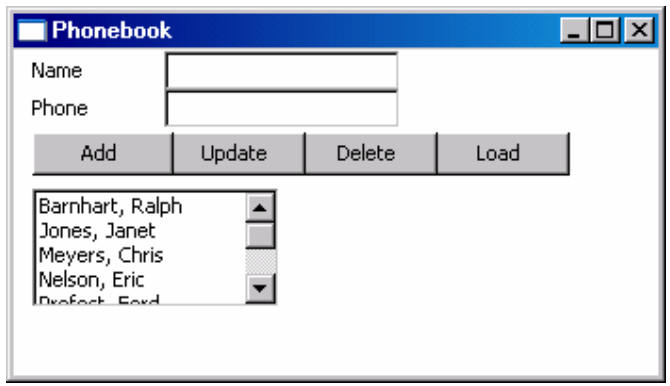
```
    def addEntry (self,event) :
        phonelist.append ([self.vname.GetValue(), \
        self.vphone.GetValue()])
        self.setSelect ()
```

```
def loadEntry  (self, event):
name, phone = phonelist[self.whichSelected()]
        print name, phone
        self.vname.SetValue(name)
        self.vphone.SetValue(phone)
```

imenik

```
def setSelect (self) :
    print "setSelect"
    #pass
    phonelist.sort()
    self.select.Clear()
    for name,phone in phonelist :
        self.select.Append (name)
```





Zadatak

Unaprijedi GUI za imenik. Uljepšaj izgled pomoću pos i size argumenata za widgete
Dodaj događaj za list box, selektirani element se odmah prikazuje, ukloni suvišan gumb Load .

širina i visina widgeta, sizer

tipovi Tkinter- pack, pakiranje widgeta

Sizer Type	Description
Grid	A very basic grid layout. Best used when the widgets you are placing are all exactly the same size and neatly fall into a regular grid.
Flex grid	A slight change from the grid sizer, allowing better results when the widgets are different sizes.
Grid bag	The most flexible member of the grid sizer family, allowing for more arbitrary placement of widgets in the grid. Useful for layouts where the display can be thought of as an irregular grid, with perhaps some items that take up more than one grid square.
Box	Either a horizontal or vertical box with widgets laid out in a line. Very flexible in controlling widget behavior when resized. Generally used in a nested fashion. Useful for nearly any kind of layout, although figuring out exactly how to nest the boxes can be tricky.
Static box	A standard box sizer with a line and a title around it.

3 Koraka

1. Dodamo sizer u kontejner (klasa roditelja, npr. Panel ili Frame) odnosno widget. Pomoću sizer-a kontroliramo "potomke" roditelja, npr. gumbe u Panelu. Kontrolu postižemo metodom SetSizer(sizer). sizer je u wx.Window klasi, tj. svaki widget može imati sizer premda je koristan samo za klase koji su kontejneri.
2. Dodaj widgete sizer-u pomoću metode Add().
3. Kada je veličina prozora ovisna o widgetima, koristi metodu Fit(). Nema smisla kada je veličina prozora zadana, u tom slučaju koristi FitInside() koji ne mijenja veličinu "prozora" roditelja. Ovaj korak nije obavezan, fitanje je u nekim slučajevima nepotrebno.

grid

```
import wx
from blockwindow import BlockWindow

labels = "one two three four five six seven eight nine".split()

class GridSizerFrame(wx.Frame):
    def __init__(self):
        wx.Frame.__init__(self, None, -1, "Basic Grid Sizer")
        sizer = wx.GridSizer(rows=3, cols=3, hgap=5, vgap=5)
        for label in labels:
            bw = BlockWindow(self, label=label)
            sizer.Add(bw, 0, 0)
        self.SetSizer(sizer)
        self.Fit()

app = wx.PySimpleApp()
GridSizerFrame().Show()
app.MainLoop()
```

Create the grid sizer

Add widget to sizer

Associate sizer with frame

dodavanje objekata

3 metode

```
Add(window, proportion=0, flag=0, border=0, userData=None)
Add(sizer, proportion=0, flag=0, border=0, userData=None)
Add(size, proportion=0, flag=0, border=0, userData=None)
```

```
Insert(index, window, proportion=0, flag=0, border=0, userData=None)
Insert(index, sizer, proportion=0, flag=0, border=0, userData=None)
Insert(index, size, proportion=0, flag=0, border=0, userData=None)
```

```
Prepend(window, proportion=0, flag=0, border=0, userData=None)
Prepend(sizer, proportion=0, flag=0, border=0, userData=None)
Prepend(size, proportion=0, flag=0, border=0, userData=None)
```

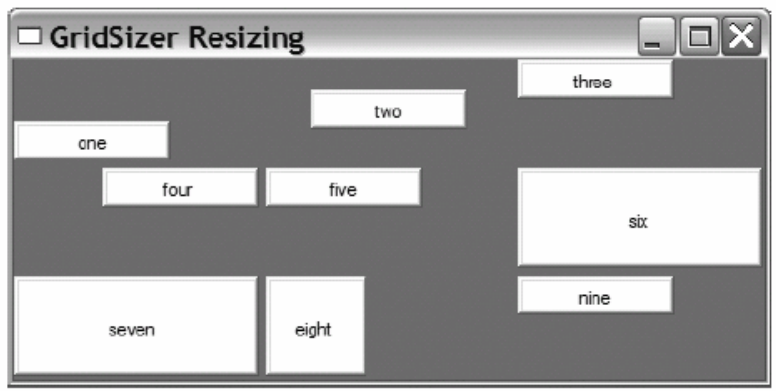
uklanjanje objekata

```
Detach(window)
Detach(sizer)
Detach(index)
```

veličina prozora

```
import wx
from blockwindow import BlockWindow

labels = "one two three four five six seven eight nine".split()
flags = {"one": wx.ALIGN_BOTTOM, "two": wx.ALIGN_CENTER,
         "four": wx.ALIGN_RIGHT, "six": wx.EXPAND, "seven": wx.EXPAND,
         "eight": wx.SHAPED}

class TestFrame(wx.Frame):
    def __init__(self):
        wx.Frame.__init__(self, None, -1, "GridSizer Resizing")
        sizer = wx.GridSizer(rows=3, cols=3, hgap=5, vgap=5)
        for label in labels:
            bw = BlockWindow(self, label=label)
            flag = flags.get(label, 0)
            sizer.Add(bw, 0, flag)
        self.SetSizer(sizer)
        self.Fit()

app = wx.PySimpleApp()
TestFrame().Show()
app.MainLoop()
```

Alignment flags



metode

Function	Description
Add(window, proportion=0, flag=0, border=0, userData=None) Add(sizer, proportion=0, flag=0, border=0, userData=None) Add(size, proportion=0, flag=0, border=0, userData=None)	Adds an item to the sizer. The first version adds a wxWindow, the second a nested sizer. The third version adds empty space which is used as a separator and is subject to the same rules for positioning as a window would be. The `proportion` argument manages the size amount that the window changes relative to other windows—it's only meaningful for a `wx.BoxSizer`. The `flag` argument is a bitmap with many different flags for alignment, border position, and growth. A full list is in chapter 11. The `border` argument is the amount of space in pixels to place around the window or sizer. `userData` allows you to associate data with the object, for example in a subclass that might need more information for sizing.
Fit(window) FitInside(window)	Causes the `window` argument to resize to the sizer's minimum size. The argument is usually the window using the sizer. The `FitInside()` method is similar, but instead of changing the screen display of the window, only changes its internal representation. This is used for a window inside a scroll panel to trigger scroll bar display.

metode

GetSize()	Returns the size of the sizer as a `wx.Size` object.
GetPosition()	Returns the position of the sizer as a `wx.Point` object.
GetMinSize()	Returns the minimum size needed to fully lay out the sizer as a `wx.Size` object.
Layout()	Programatically forces the sizer to recalculate the size and position of its children. Call after dynamically adding or removing a child.
Prepend(...)	Identical to `Add()` (all three versions, but the new object is placed at the beginning of the sizer list for layout purposes).
Remove(window) Remove(sizer) Remove(nth)	Removes an object from the sizer. Depending on the version, either a specific object or the nth in the sizer list is removed. If this is done after startup, call `Layout()` after.
SetDimension(x, y, width, height)	Programatically forces the sizer to take the given size, and causes all children to reposition themselves

poravnavanje

Flag	Description
wx.ALIGN_BOTTOM	Aligns the widget to the bottom of its allotted space.
wx.ALIGN_CENTER	Places the widget so that the center of the widget is in the center of its allotted space.
wx.ALIGN_CENTER_HORIZONTAL	Places the widget so that it is centered horizontally in its allotted space.
wx.ALIGN_CENTER_VERTICAL	Places the widget so that it is centered vertically in its allotted space.
wx.ALIGN_LEFT	Aligns the widget so that it is against the left edge of its allotted space. This is the default behavior.
wx.ALIGN_TOP	Aligns the widget so that it is against the top edge of its allotted space. This is the default behavior.
wx.EXPAND	Changes the size of the widget to fill its allotted space any time the size of the parent window changes.

minimalna veličina

```
import wx
from blockwindow import BlockWindow

labels = "one two three four five six seven eight nine".split()

class TestFrame(wx.Frame):
    def __init__(self):
        wx.Frame.__init__(self, None, -1, "GridSizer Test")
        sizer = wx.GridSizer(rows=3, cols=3, hgap=5, vgap=5)
        for label in labels:
            bw = BlockWindow(self, label=label)
            sizer.Add(bw, 0, 0)
        center = self.FindWindowByName("five")
        center.SetMinSize((150,50))
        self.SetSizer(sizer)
        self.Fit()
app = wx.PySimpleApp()
TestFrame().Show()
app.MainLoop()
```

minimalna veličina pojedinih widgeta

```
SetItemMinSize(window, size)
SetItemMinSize(sizer, size)
SetItemMinSize(index, size)
```

box sizer

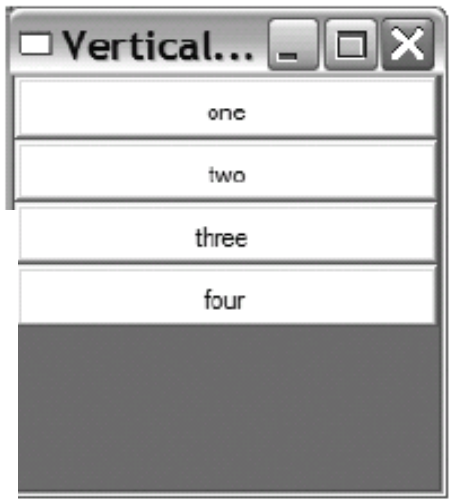


Figure 11.13
A vertical sizer with one stretch element

Figure 11.14
A vertical sizer with two stretch elements

box sizer

```
import wx
from blockwindow import BlockWindow

labels = "one two three four".split()

class TestFrame(wx.Frame):
    title = "none"
    def __init__(self):
        wx.Frame.__init__(self, None, -1, self.title)
        sizer = self.CreateSizerAndWindows()
        self.SetSizer(sizer)
        self.Fit()

class VBoxSizerFrame(TestFrame):
    title = "Vertical BoxSizer"

    def CreateSizerAndWindows(self):
        sizer = wx.BoxSizer(wx.VERTICAL)
        for label in labels:
            bw = BlockWindow(self, label=label, size=(200,30))
            sizer.Add(bw, flag=wx.EXPAND)
        return sizer
```

The vertical sizer

box sizer

```
class HBoxSizerFrame(TestFrame):
    title = "Horizontal BoxSizer"

    def CreateSizerAndWindows(self):
        sizer = wx.BoxSizer(wx.HORIZONTAL)
        for label in labels:
            bw = BlockWindow(self, label=label, size=(75,30))
            sizer.Add(bw, flag=wx.EXPAND)
        return sizer

class VBoxSizerStretchableFrame(TestFrame):
    title = "Stretchable BoxSizer"

    def CreateSizerAndWindows(self):
        sizer = wx.BoxSizer(wx.VERTICAL)
        for label in labels:
            bw = BlockWindow(self, label=label, size=(200,30))
            sizer.Add(bw, flag=wx.EXPAND)

        # Add an item that takes all the free space
        bw = BlockWindow(self, label="gets all free space", size=(200,30))
        sizer.Add(bw, 1, flag=wx.EXPAND)
        return sizer
```

The horizontal sizer

Horizontal with free space

box sizer

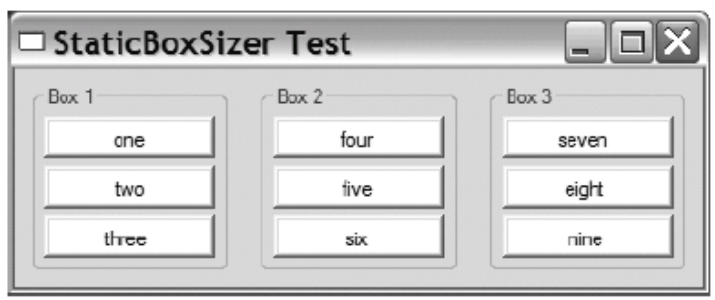
```
class VBoxSizerMultiProportionalFrame(TestFrame):    <-- Proportional sizing
    title = "Proportional BoxSizer"

    def CreateSizerAndWindows(self):
        sizer = wx.BoxSizer(wx.VERTICAL)
        for label in labels:
            bw = BlockWindow(self, label=label, size=(200,30))
            sizer.Add(bw, flag=wx.EXPAND)

        # Add an item that takes one share of the free space
        bw = BlockWindow(self,
                label="gets 1/3 of the free space",
                size=(200,30))
        sizer.Add(bw, 1, flag=wx.EXPAND)

        # Add an item that takes 2 shares of the free space
        bw = BlockWindow(self,
                label="gets 2/3 of the free space",
                size=(200,30))
        sizer.Add(bw, 2, flag=wx.EXPAND)
        return sizer
```

static box sizer

```
import wx
from blockwindow import BlockWindow

labels = "one two three four five six seven eight nine".split()

class TestFrame(wx.Frame):
    def __init__(self):
        wx.Frame.__init__(self, None, -1, "StaticBoxSizer Test")
        self.panel = wx.Panel(self)

        box1 = self.MakeStaticBoxSizer("Box 1", labels[0:3])
        box2 = self.MakeStaticBoxSizer("Box 2", labels[3:6])
        box3 = self.MakeStaticBoxSizer("Box 3", labels[6:9])

        sizer = wx.BoxSizer(wx.HORIZONTAL)
        sizer.Add(box1, 0, wx.ALL, 10)
        sizer.Add(box2, 0, wx.ALL, 10)
        sizer.Add(box3, 0, wx.ALL, 10)

        self.panel.SetSizer(sizer)
        sizer.Fit(self)

    def MakeStaticBoxSizer(self, boxlabel, itemlabels):
        box = wx.StaticBox(self.panel, -1, boxlabel)
        sizer = wx.StaticBoxSizer(box, wx.VERTICAL)
```

Make static boxes

Use sizer to manage others



Make static box

static box sizer

```
        for label in itemlabels:
            bw = BlockWindow(self.panel, label=label)
            sizer.Add(bw, 0, wx.ALL, 2)

        return sizer

app = wx.PySimpleApp()
TestFrame().Show()
app.MainLoop()
```

Add items to box

```
wx.StaticBox(parent, id, label, pos=wx.DefaultPosition,
        size=wx.DefaultSize, style=0, name="staticBox")
```

```
box = wx.StaticBox(self.panel, -1, boxlabel)
```

Aplikacija


Real World Test
Account Information
Name:
Address:
City, State, Zip:
Phone:
Email:
Save
Cancel

Aplikacija

```
import wx

class TestFrame(wx.Frame):
    def __init__(self):
        wx.Frame.__init__(self, None, -1, "Real World Test")
        panel = wx.Panel(self)

        # First create the controls
        topLbl = wx.StaticText(panel, -1, "Account Information")
        topLbl.SetFont(wx.Font(18, wx.SWISS, wx.NORMAL, wx.BOLD))

        nameLbl = wx.StaticText(panel, -1, "Name:")
        name = wx.TextCtrl(panel, -1, "");

        addrLbl = wx.StaticText(panel, -1, "Address:")
        addr1 = wx.TextCtrl(panel, -1, "");
        addr2 = wx.TextCtrl(panel, -1, "");

        cstLbl = wx.StaticText(panel, -1, "City, State, Zip:")
        city  = wx.TextCtrl(panel, -1, "", size=(150,-1));
        state = wx.TextCtrl(panel, -1, "", size=(50,-1));
        zip   = wx.TextCtrl(panel, -1, "", size=(70,-1));
```

Creating widgets ❶

Aplikacija

```
phoneLbl = wx.StaticText(panel, -1, "Phone:")
phone = wx.TextCtrl(panel, -1, "");

emailLbl = wx.StaticText(panel, -1, "Email:")
email = wx.TextCtrl(panel, -1, "");

saveBtn = wx.Button(panel, -1, "Save")
cancelBtn = wx.Button(panel, -1, "Cancel")
# mainSizer is the top-level one that manages everything
mainSizer = wx.BoxSizer(wx.VERTICAL)
mainSizer.Add(topLbl, 0, wx.ALL, 5)
mainSizer.Add(wx.StaticLine(panel), 0,
        wx.EXPAND|wx.TOP|wx.BOTTOM, 5)

# addrSizer is a grid that holds all of the address info
addrSizer = wx.FlexGridSizer(cols=2, hgap=5, vgap=5)
addrSizer.AddGrowableCol(1)
addrSizer.Add(nameLbl, 0,
        wx.ALIGN_RIGHT|wx.ALIGN_CENTER_VERTICAL)
addrSizer.Add(name, 0, wx.EXPAND)
addrSizer.Add(addrLbl, 0,
        wx.ALIGN_RIGHT|wx.ALIGN_CENTER_VERTICAL)
addrSizer.Add(addr1, 0, wx.EXPAND)
```

❷ **Vertical sizer**

❸ **Columns for address**

Aplikacija

```
addrSizer.Add((10,10)) # some empty space
addrSizer.Add(addr2, 0, wx.EXPAND)

addrSizer.Add(cstLbl, 0,
        wx.ALIGN_RIGHT|wx.ALIGN_CENTER_VERTICAL)

# the city, state, zip fields are in a sub-sizer
cstSizer = wx.BoxSizer(wx.HORIZONTAL)
cstSizer.Add(city, 1)
cstSizer.Add(state, 0, wx.LEFT|wx.RIGHT, 5)
cstSizer.Add(zip)
addrSizer.Add(cstSizer, 0, wx.EXPAND)
 addrSizer.Add(phoneLbl, 0,
         wx.ALIGN_RIGHT|wx.ALIGN_CENTER_VERTICAL)
 addrSizer.Add(phone, 0, wx.EXPAND)
 addrSizer.Add(emailLbl, 0,
         wx.ALIGN_RIGHT|wx.ALIGN_CENTER_VERTICAL)
 addrSizer.Add(email, 0, wx.EXPAND)
  # now add the addrSizer to the mainSizer
  mainSizer.Add(addrSizer, 0, wx.EXPAND|wx.ALL, 10)
```

4 Row with empty space

5 Nested horizontal

6 Phone and email

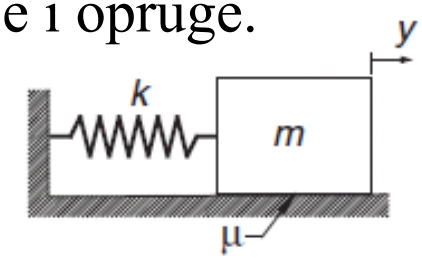
Aplikacija

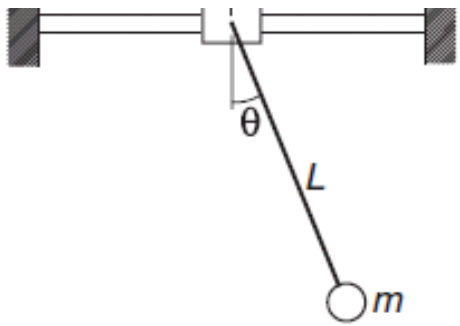
```
        btnSizer = wx.BoxSizer(wx.HORIZONTAL)
        btnSizer.Add((20,20), 1)
        btnSizer.Add(saveBtn)
        btnSizer.Add((20,20), 1)
        btnSizer.Add(cancelBtn)
        btnSizer.Add((20,20), 1)

        mainSizer.Add(btnSizer, 0, wx.EXPAND|wx.BOTTOM, 10)

        panel.SetSizer(mainSizer)

        # Fit the frame to the needs of the sizer. The frame will
        # automatically resize the panel as needed. Also prevent the
        # frame from getting smaller than this size.
        mainSizer.Fit(self)
        mainSizer.SetSizeHints(self)

app = wx.PySimpleApp()
TestFrame().Show()
app.MainLoop()
```

8 Button row

projekt nr 2

Diferencijalne jednadžba HO istog je oblika za matematičko njihalo i sustav mase i opruge.





$$\frac{\mathrm{d}^2}{\mathrm{d}t^2}y(t) + \frac{\mu\left(\frac{\mathrm{d}}{\mathrm{d}t}y(t)\right)}{m} + \frac{k\,y(t)}{m} = 0$$

Oblik koji odgovara simulaciji

$$\frac{\mathrm{d}^2}{\mathrm{d}t^2}y(t) = -A\left(\frac{\mathrm{d}}{\mathrm{d}t}y(t)\right) - B\,y(t)$$

Djelovanje vanjske sile

$$\frac{\mathrm{d}^2}{\mathrm{d}t^2}y(t) = -A\left(\frac{\mathrm{d}}{\mathrm{d}t}y(t)\right) - B\,y(t) + f(x)$$

RK 4 algoritam

```
 X,Y = integrate(F,x,y,xStop,h).
4th-order Runge-Kutta method for solving the
initial value problem {y}' = {F(x,{y})}, where
{y} = {y[0],y[1],...y[n-1]}.
x,y   = initial conditions.
xStop = terminal value of x.
h     = increment of x used in integration.
F     = user-supplied function that returns the
        array F(x,y) = {y'[0],y'[1],...,y'[n-1]}.
```

RK4

$$K_0 = hF(x, y)$$

$$K_1 = hF\left(x + \frac{h}{2}, y + \frac{K_0}{2}\right)$$

$$K_2 = hF\left(x + \frac{h}{2}, y + \frac{K_1}{2}\right)$$

$$K_3 = hF(x + h, y + K_2)$$

$$y(x + h) = y(x) + \frac{1}{6}(K_0 + 2K_1 + 2K_2 + K_3)$$

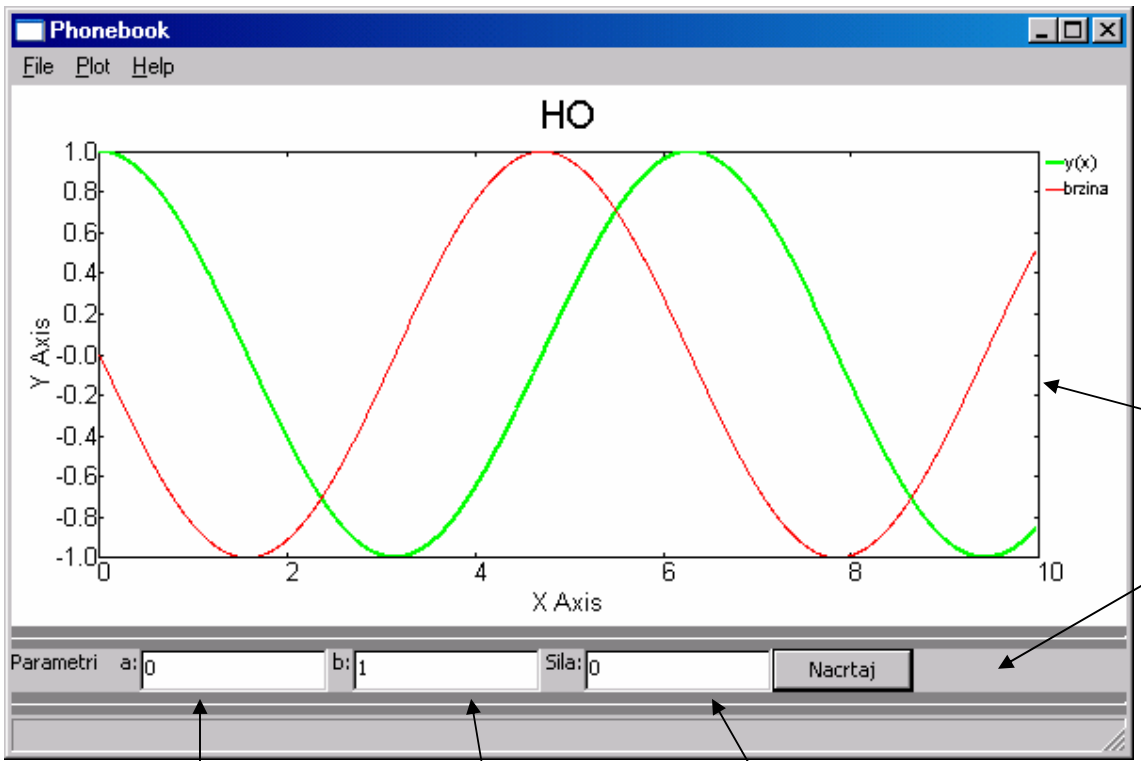
$$y' = F(x, y) = \begin{bmatrix} y_0' \\ y_1' \end{bmatrix}$$

```
def F1(x,y):
    F1 = zeros((2))
    F1[0] = y[1]
    F1[1] = -A*y[1] -B* y[0]
    return F1
```

$$\frac{d}{dx} y(x) = v(x)$$

$$\frac{d}{dx} v(x) = -A\, v(x) - B\, y(x)$$

aplikacija



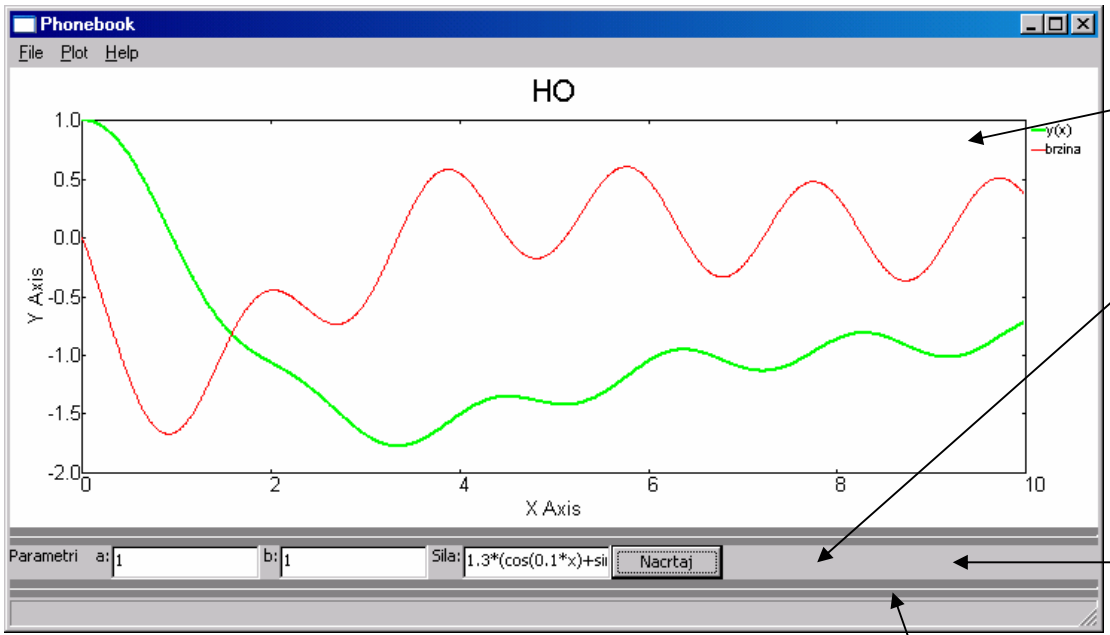
wx.lib.plot

wx/lib/plot.py file

2 panela - graf i ulazni parametri

gušenje frekvencija vanjska sila

aplikacija



self.client - graf panel

self.panel - parametri

box=wx.BoxSizer(wx.HORIZONTAL)

box.Add(lb1,0,wx.EXPAND)

.....

sizer=wx.BoxSizer(wx.VERTICAL)

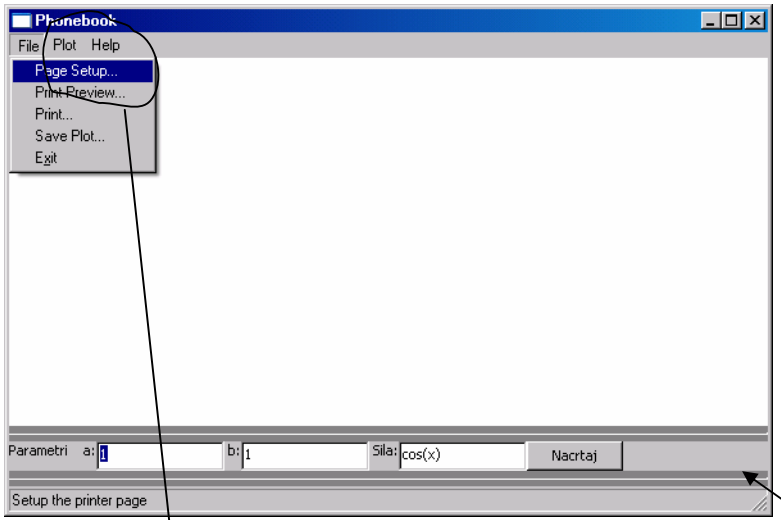
sizer.Add(self.client,1,wx.EXPAND)

horizontalna crta

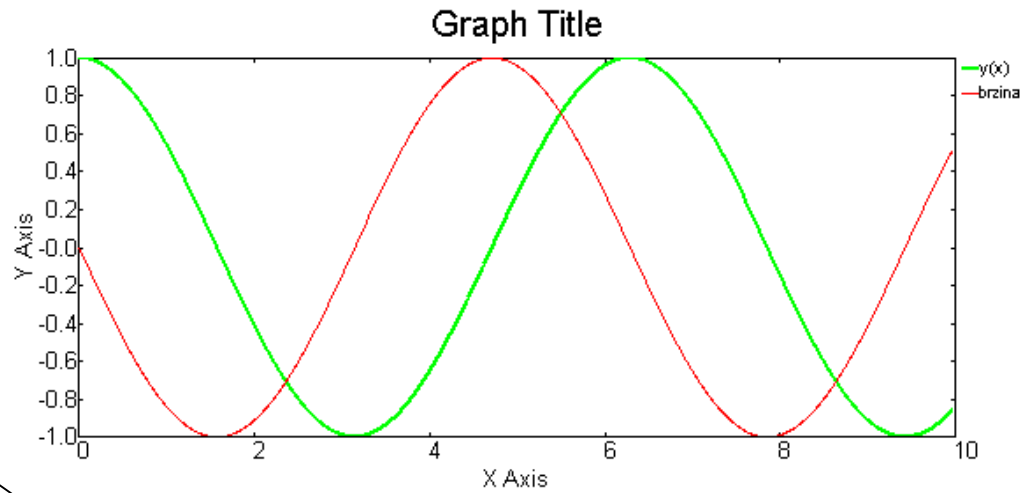
sizer.Add(wx.StaticLine(self),0,wx.EXPAND|wx.BOTTOM|wx.TOP,5)

sizer.Add(self.panel,0,wx.EXPAND)

wx.lib.plot



slika png format



Nije dovršeno, ostatak iz primjera.

Da li treba izbornik?

Ne mjenja veličinu HORIZONTAL BOX

Nedostaje: izbor koraka integracije, vrijeme simulacije i početne vrijednosti.