Interfete grafice in Python

1. Introducere

wxPython API conţine un set de funcţii şi controale pentru construirea interfeţelor grafice în python, fiind construit peste framework-ul C++ GUI, denumit **wxWidgets.** La randul lui, **wxWidgets** este un **wrapper** peste funcţiile native ale sistemului de operare folosit. De exemplu, pe *Windows*, este un wrapper peste *Winapi*. O aplicaţie **wxWidgets** nu „desenează" controalele folosite (cum face QT, de exemplu), ci foloseşte chiar controalele native. Pe Unix/Linux, cea mai stabila versiune este un wrapper peste **GTK 2.x**.

wxWidgets a fost dezvoltat in **C++**, dar există şi diverse binding-uri pentru limbaje precum **Python** sau **Perl**. Aceste binding-uri prezinta diverse nivele de integrare cu codul **C++**, cel mai dezvoltat fiind cel de **Python** (cunoscut sub numele de **wxPython**).

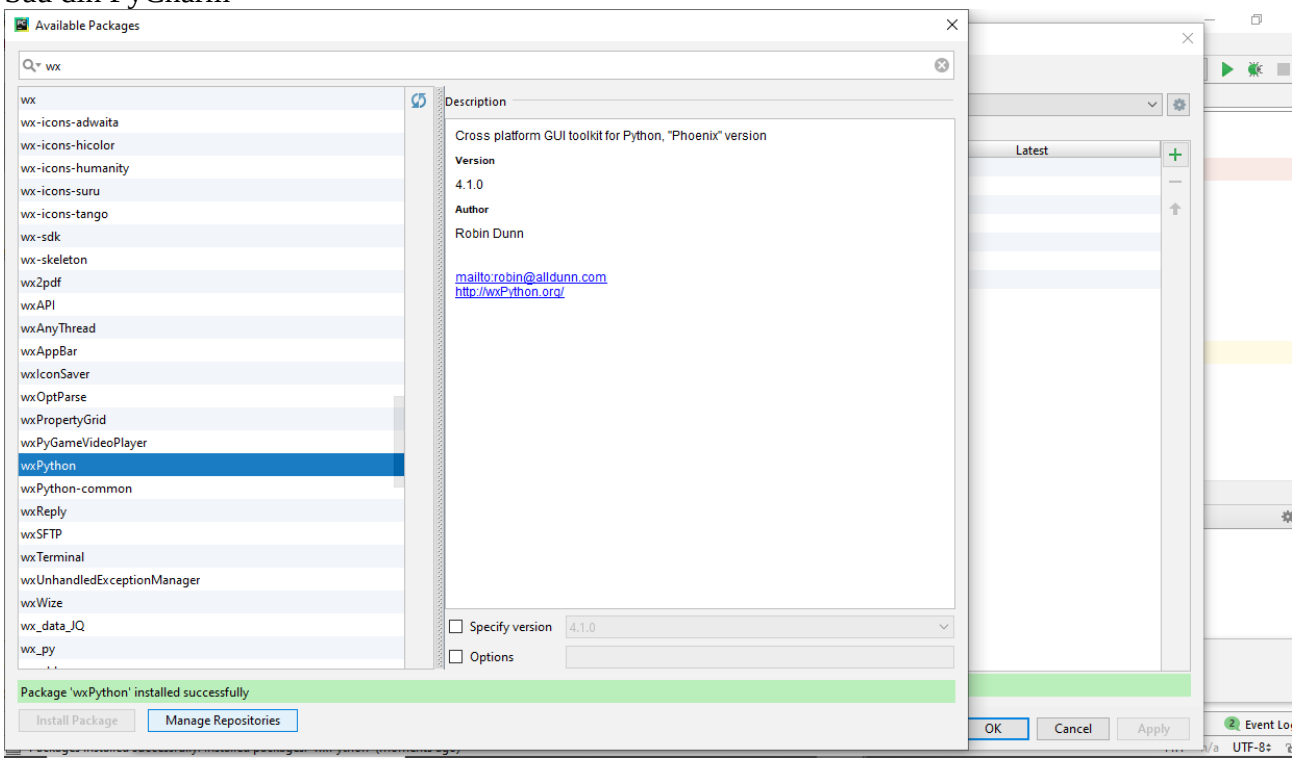
Structura framework-ului **wxWidgets**, precum şi modul in care este folosit, prezinta similaritati cu framework-ul **MFC (Microsoft Foundation Classes)**.

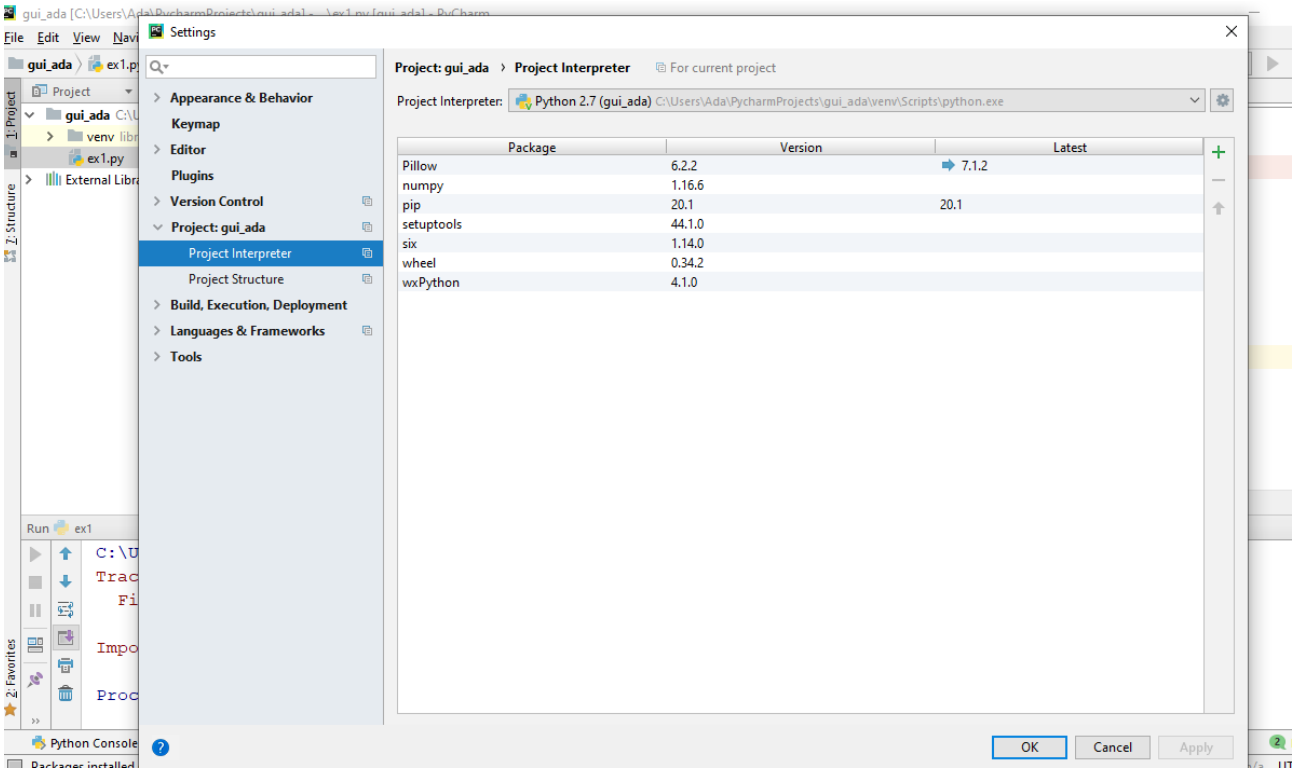
2. Crearea aplicaţiilor de tip GUI

Pentru a putea crea interfeţe grafice folosind framework-ul wxPython, trebuie importat modulul `wx`.

Inainte de acest import, se instaleaza biblioteca wxPython cu comanda: pip install -U wxPython (dintr-un terminal)

Sau din PyCharm






```
import wx
```

Instanţierea unei aplicaţii de tip GUI se realizează prin apelarea funcţiei `App()`:

```
app=wx.App()
```

Următoarea instrucţiune va instanţia un obiect de tip frame:

```
frame = wx.Frame(None, -1, "Hello World!")
```

Pentru ca acesta să apară efectiv pe ecran, trebuie apelată metoda `Show()`.

```
frame.Show()
```

Pentru ca aplicaţia să fie funcţională, trebuie introdusă funcţia `MainLoop()`. Această funcţie creează o buclă continuă care recepţionează toate evenimentele aplicaţiei. MainLoop reprezintă o parte integrantă a oricărei aplicaţii de tip GUI.

```
app.MainLoop()
```

Cea mai simplă aplicaţie de tip GUI poate fi construită astfel:

```
#ex1.py

import wx

app = wx.App()

frame = wx.Frame(None, -1, "Hello World!")
frame.Show()

app.MainLoop()
```

2.1 wx.Window

wx.Window reprezintă clasa de bază din care sunt derivate toate controalele. De exemplu, controlul **wx.Frame** moşteneşte clasa **wx.Window**, ceea ce înseamnă că metodele clasei **wx.Window** pot fi apelate de către clasa **wx.Frame**.

În exemplul următor sunt apelate o serie de metode ale clasei **wx.Window**:

```
#ex2.py

import wx

app = wx.App()

frame = wx.Frame(None, -1, '')
frame.SetToolTip(wx.ToolTip('Acesta este un frame'))
frame.SetCursor(wx.Cursor(wx.CURSOR_MAGNIFIER))
frame.SetPosition(wx.Point(0,0))
frame.SetSize(wx.Size(300,250))
frame.SetTitle('Hello World')
frame.Show()

app.MainLoop()
```

2.2 wx.Frame

wx.Frame este un container de widget-uri, având următorul constructor:

```
wx.Frame (wx.Window parent, id, string title,  wx.Point pos=wx.DefaultPosition,
wx.Size size=wx.DefaultSize, style=wx.DEFAULT_FRAME_STYLE, string name='frame')
```

Python permite utilizarea parametrilor cu valori implicite. Astfel, singurii parametri obligatorii în **wx.Frame** sunt: **parent**, **id** şi **title**.

Dacă sunt specificaţi toţi parametrii în ordinea în care apar în constructor, nu trebuie specificate şi numele acestora. De exemplu, pentru a crea un frame, **fără** să aibă **o fereastră părinte**, având indicatorul **150** şi titlul "**Hello World!**", la poziţia **(100,100)**, cu dimensiunea **(200,400)**, codul va arăta astfel:

```
frame=wx.Frame(None, 150, ' Hello World!', wx.Point(100,100), wx.Size(200,400))
```

Dacă se omit o parte dintre parametri, trebuie specificaţi toţi parametri care urmează. De exemplu, dacă se omite parametrul **pos**, atunci trebui specificat în mod explicit parametrul **size**:

```
frame = wx.Frame(None, 100, 'Title', size = wx.Size(100,100))
```

Structura standard a unui program python:

De obicei, există un singur fişier cu ajutorul căruia este lansată în execuţie aplicaţia. Programele mai complexe sunt structurate în mai multe fişiere.

Pentru a lansa în execuţie o aplicaţie, Python foloseşte o variabilă specială, denumită name. Această variabilă va avea valoarea main atunci când aplicaţia este lansată din linia de comandă sau atunci când se execută dublu click pe numele fişierului. În acel moment va fi apelată funcţia main().

2.3 wx.MenuBar

Clasa **wx.MenuBar** este folosită atunci când se doreşte crearea unei bare de meniu. Opţiunile unui meniu pot fi elemente **simple**, elemente de **selecţie (check)** sau elemente de tip **radio**.

Mai întâi, pentru a crea o bară de meniu, se instanţiază clasa `wx.MenuBar()`:

```
baraMeniu = wx.MenuBar()
```

Apoi, trebuie create meniurile:

```
meniuFisier = wx.Menu()
meniuEditare = wx.Menu()
meniuAjutor = wx.Menu()
```

Pentru a adăuga opţiuni în cadrul meniului, se foloseşte funcţia `Append()`:

```
meniuFisier.Append(11, '&Deschidere', 'Deschide document')
meniuFisier.Append(12, '&Salvare', 'Salveaza documentul')
```

Pentru a separa logic un grup de opţiuni din cadrul unui meniu, se poate folosi un separator:

```
meniuFisier.AppendSeparator()
```

Pentru a introduce icon-uri în cadrul meniului, elementele meniului trebuie create manual. Icon-urile pot fi doar imagini de tip bitmap, orice alt tip trebuie convertit la bitmap. De exemplu:

```
meniuIesire = wx.MenuItem(meniuFisier, 13, '&Iesire\tCtrl+Q', 'Parasire aplicatie')
meniuIesire.SetBitmap(wx.Image('icon_name.png', wx.BITMAP_TYPE_PNG).ConvertToBitmap())
meniuFisier.AppendItem(meniuIesire)
```

Meniurile de mai sus conţin elemente simple. Pentru a crea meniuri care să conţine elemente de tip check sau radio, se foloseşte parametrul **kind**. Valorile posibile ale parametrului **wx.ItemKind** sunt:

- `wx.ITEM_NORMAL` – element simplu
- `wx.ITEM_CHECK` – element de tip check
- `wx.ITEM_RADIO` – element de tip radio

Exemplele de mai jos descriu modul în care sunt create meniurile de tip check sau radio.

```
meniuEditare.Append(21, 'Selecteaza optiune 1', '', wx.ITEM_CHECK)
meniuEditare.Append(22, 'Selecteaza optiune 2', '', kind=wx.ITEM_CHECK)
```

O altă variantă ar fi:

```
meniuIesire = wx.MenuItem(meniuFisier, 13, '&Iesire\tCtrl+Q', 'Parasire
aplicatie', wx.ITEM_NORMAL)
```

Submeniurile sunt la rândul lor meniuri. Astfel, pentru a crea un submeniu, trebuie declarat astfel:

```
submeniu = wx.Menu()
```

Apoi trebuie adăugate elementele:

```
submeniu.Append(231, 'Element de tip radio 1', kind=wx.ITEM_RADIO)
submeniu.Append(232, 'Element de tip radio 2', kind=wx.ITEM_RADIO)
submeniu.Append(233, 'Element de tip radio 3', kind=wx.ITEM_RADIO)
```

În final, submeniul trebuie adăugat unui meniu existent:

```
meniuEditare.AppendMenu(23, 'Submeniu', submeniu)
```

Meniurile astfel create trebuie adăugate la bara de meniu:

```
baraMeniu.Append(meniuFisier, '&Fisier')
```

```
baraMeniu.Append(meniuEditare, '&Editare')
baraMeniu.Append(meniuAjutor, '&Ajutor')
```

În final, trebuie setat meniul în cadrul clasei.

```
self.SetMenuBar(baraMeniu)
```

Exemplul de mai jos creează o bară de meniu, cu diverse tipuri de opţiuni:

```
#ex3.py

import wx

class Meniu(wx.Frame):
    def __init__(self, parent, id, title):
        wx.Frame.__init__(self,   parent,   id,   title,   wx.DefaultPosition,
wx.Size(300, 150))

        baraMeniu = wx.MenuBar()

        meniuFisier = wx.Menu()
        meniuEditare = wx.Menu()
        meniuAjutor = wx.Menu()

        meniuFisier.Append(11, '&Deschidere', 'Deschide document')
        meniuFisier.Append(12, '&Salvare', 'Salveaza documentul')

        meniuFisier.AppendSeparator()

        meniuIesire = wx.MenuItem(meniuFisier, 13, '&Iesire\tCtrl+Q', 'Parasire
aplicatie')
        #quit.SetBitmap(wx.Image('stock_exit-16.png',
wx.BITMAP_TYPE_PNG).ConvertToBitmap())
        meniuFisier.Append(meniuIesire)

        meniuEditare.Append(21, 'Selecteaza optiune 1', '', wx.ITEM_CHECK)
        meniuEditare.Append(22, 'Selecteaza optiune 2', '', kind=wx.ITEM_CHECK)

        submeniu = wx.Menu()
        submeniu.Append(231, 'Element de tip radio 1', kind=wx.ITEM_RADIO)
        submeniu.Append(232, 'Element de tip radio 2', kind=wx.ITEM_RADIO)
        submeniu.Append(233, 'Element de tip radio 3', kind=wx.ITEM_RADIO)
        meniuEditare.Append(23, 'Submeniu', submeniu)

        meniuAjutor.Append(31, '&About','Informatii despre aplicatie')

        baraMeniu.Append(meniuFisier, '&Fisier')
        baraMeniu.Append(meniuEditare, '&Editare')
        baraMeniu.Append(meniuAjutor, '&Ajutor')
        self.SetMenuBar(baraMeniu)
        self.CreateStatusBar()

class Aplicatie(wx.App):
    def OnInit(self):
        frame = Meniu(None, -1, 'Bara de meniu simpla')
        frame.Show(True)
        return True

app = Aplicatie()
app.MainLoop()
```

În continuare, trebuie definite acţiunile utilizatorilor. În momentul în care utilizatorul selectează o opţiune din meniu este generat un eveniment. Pentru a realiza o anumită operaţie la selectarea unei opţiuni a meniului, trebuie implementat un manager de evenimente. Pentru aceasta se foloseşte clasa **wx.EVT_MENU**.

De exemplu, pentru a adăuga un eveniment opţiunii **Iesire** din cadrul meniului **Fisier**, trebuie furnizate 3 informaţii:

- **obiectul** de care legăm evenimentul respectiv (obiectul aplicaţiei principale – **self**);
- **id**-ul elementului din meniu (**13** – id-ul opţiunii **Iesire**);
- numele metodei care execută acţiunea (OnQuit).

```
wx.EVT_MENU(self, 13, self.OnQuit )
```

Metoda apelată la apariţia evenimentului are doi parametri: obiectul în cadrul căruia este definită metoda şi evenimentul generat.

O altă modalitate de a lega evenimentul de o acţiune, este apelarea metodei `Bind()`:

```
self.Bind(wx.EVT_MENU, self.OnQuit, id=13)
```

Exemplul de mai jos defineşte o metodă care închide frame-ul, precum şi o metodă care afişează o fereastră de tip **About**.

```
#ex4.py

import wx

class Meniu(wx.Frame):
    def __init__(self, parent, id, title):
        wx.Frame.__init__(self,   parent,   id,   title,   wx.DefaultPosition,
wx.Size(300, 150))

        baraMeniu = wx.MenuBar()

        meniuFisier = wx.Menu()
        meniuEditare = wx.Menu()
        meniuAjutor = wx.Menu()

        meniuFisier.Append(11, '&Deschidere', 'Deschide document')
        meniuFisier.Append(12, '&Salvare', 'Salveaza documentul')

        meniuFisier.AppendSeparator()

        meniuIesire = wx.MenuItem(meniuFisier, 13, '&Iesire\tCtrl+Q', 'Parasire
aplicatie')
        #quit.SetBitmap(wx.Image('stock_exit-16.png',
wx.BITMAP_TYPE_PNG).ConvertToBitmap())
        meniuFisier.AppendItem(meniuIesire)

        meniuEditare.Append(21, 'Selecteaza optiune 1', '', wx.ITEM_CHECK)
        meniuEditare.Append(22, 'Selecteaza optiune 2', '', kind=wx.ITEM_CHECK)

        submeniu = wx.Menu()
        submeniu.Append(231, 'Element de tip radio 1', kind=wx.ITEM_RADIO)
        submeniu.Append(232, 'Element de tip radio 2', kind=wx.ITEM_RADIO)
        submeniu.Append(233, 'Element de tip radio 3', kind=wx.ITEM_RADIO)
        meniuEditare.AppendMenu(23, 'Submeniu', submeniu)

        meniuAjutor.Append(31, '&About','Informatii despre aplicatie')

        baraMeniu.Append(meniuFisier, '&Fisier')
```

```
        baraMeniu.Append(meniuEditare, '&Editare')
        baraMeniu.Append(meniuAjutor, '&Ajutor')
        self.SetMenuBar(baraMeniu)
        self.CreateStatusBar()

        wx.EVT_MENU(self, 13, self.OnQuit)
        self.Bind(wx.EVT_MENU, self.OnHelp, id=31)

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

    def OnHelp(self,event):
        info = wx.AboutDialogInfo()
        info.Name = "Hello World"
        info.Version = "1.0"
        info.Copyright = "(C) 2010"
        info.Description = "Aici va aparea descrierea aplicatiei"
        info.WebSite = ("http://www.domeniu.com", "Adresa web site")
        info.Developers = [ "Popescu A.",
                            "Ionescu G.",
                            "Georgescu H." ]

        wx.AboutBox(info)

class Aplicatie(wx.App):
    def OnInit(self):
        frame = Meniu(None, -1, 'Bara de meniu simpla')
        frame.Show(True)
        return True

app = Aplicatie()
app.MainLoop()
```

3. Amplasarea controalelor în cadrul frame-urilor

Cele mai folosite metode de aranjare a controalelor în cadrul ferestrelor sunt:
a) amplasarea manuală, stabilind coordonatele la care va fi poziţionat controlul:

```
#ex5.py

import wx

class MyFrame(wx.Frame):
    def __init__(self, parent, id, title):
        wx.Frame.__init__(self,   parent,   id,   title,   wx.DefaultPosition,
wx.Size(250, 50))

        panel = wx.Panel(self, -1)
        wx.Button(panel, -1, "Buton1", (0,0))
        wx.Button(panel, -1, "Buton2", (80,0))
        wx.Button(panel, -1, "Buton3", (160,0))

class MyApp(wx.App):
    def OnInit(self):
        frame = MyFrame(None, -1, 'Panou cu layout fix')
        frame.Show(True)
        frame.Centre()
        return True

app = MyApp(0)
app.MainLoop()
```

În această situaţie, dacă fereastra este redimensionată, dimensiunea şi poziţia butoanelor nu se va schimba.

b) Cea de-a doua metodă este folosirea unor managere pentru aspect. Cele mai folosit sunt obiectele de tip Sizer:

- wx.BoxSizer
- wx.StaticBoxSizer
- wx.GridSizer
- wx.GridBagSizer

3.1 wx.BoxSizer

Utilizarea unui sizer permite ca aplicaţia precedentă să poată fi modificată astfel încât redimensionând fereastra, vor fi redimensionate automat şi butoanele:

```
#ex6.py

import wx

class MyFrame(wx.Frame):
    def __init__(self, parent, id, title):
        wx.Frame.__init__(self, parent, id, title, (-1, -1), wx.Size(250, 50))

        panel = wx.Panel(self, -1)
        box = wx.BoxSizer(wx.HORIZONTAL)
        box.Add(wx.Button(panel, -1, 'Buton1'), 1 )
        box.Add(wx.Button(panel, -1, 'Buton2'), 1 )
        box.Add(wx.Button(panel, -1, 'Buton3'), 1 )
        panel.SetSizer(box)
        self.Centre()

class MyApp(wx.App):
     def OnInit(self):
         frame = MyFrame(None, -1, 'Exemplu de obiect BoxSizer')
         frame.Show(True)
         return True

app = MyApp(0)
app.MainLoop()
```

Într-un **BoxSizer**, controalele pot fi plasate pe orizontală sau pe verticală. Sintaxa este:

```
box = wx.BoxSizer(integer orientation)
```

unde orientation poate lua valorile **wx.VERTICAL** sau **wx.HORIZONTAL**. Adăugarea controalelor într-un **wxBoxSizer** se realizează prin metoda **`Add()`**. Sintaxa acesteia este:

```
Add(wx.Window window, integer proportion=0, integer flag = 0, integer border =
0)
```

Parametrul **proportion** defineşte partajarea spaţiului disponibil pe care îl vor ocupa controalele în cadrul unui sizer, în direcţia specificată.

De exemplu, să presupunem că un sizer conţine 3 butoane cu proporţiile 0, 1, respectiv 2, aşezate într-un sizer orizontal. Astfel, butonul cu proporţia 0 nu-şi va schimba dimensiunea. Restul spaţiului din cadrul sizer-ului va fi împărţit în 3 părţi (1+2 spaţii). Butonul cu proporţia 2 va ocupa întotdeauna 2 dintre cele 3 spaţii disponibile, în timp ce butonul cu proporţia 1 va ocupa întotdeauna a treia parte.

```
#ex7.py

import wx

class MyFrame(wx.Frame):
    def __init__(self, parent, id, title):
        wx.Frame.__init__(self, parent, id, title, (-1, -1), wx.Size(250, 50))
        panel = wx.Panel(self, -1)
        box = wx.BoxSizer(wx.HORIZONTAL)
        box.Add(wx.Button(panel, -1, 'Buton1'), 0 )
        box.Add(wx.Button(panel, -1, 'Buton2'), 1 )
        box.Add(wx.Button(panel, -1, 'Buton3'), 2 )
        panel.SetSizer(box)
        self.Centre()

class MyApp(wx.App):
     def OnInit(self):
         frame = MyFrame(None, -1, 'Exemplu de utilizare a parametrului
"proportion"')
         frame.Show(True)
         return True

app = MyApp(0)
app.MainLoop()
```

Parametrul **flag** permite configurarea dimensiunilor marginilor (**padding**). Acesta poate lua următoarele valori:

- wx.LEFT
- wx.RIGHT
- wx.BOTTOM
- wx.TOP
- wx.ALL

Aceste valori pot fi combinate cu ajutorul operatorului |. De exemplu: **`wx.LEFT | wx.TOP`**.

Folosirea flag-ului **wx.EXPAND** permite ca un control să ocupe tot spaţiu disponibil în direcţia perpendiculară pe direcţia de orientare a sizer-ului.

De asemenea poate fi stabilită alinierea controalelor faţă de marginile ferestrei, folosind următoarele flag-uri:

- wx.ALIGN_LEFT
- wx.ALIGN_RIGHT
- wx.ALIGN_TOP
- wx.ALIGN_BOTTOM
- wx.ALIGN_CENTER_VERTICAL
- wx.ALIGN_CENTER_HORIZONTAL
- wx.ALIGN_CENTER

Exemplu: aplicarea flag-urilor prezentate mai sus

```
#ex8.py

import wx

class MyFrame(wx.Frame):
    def __init__(self, parent, id, title):
        wx.Frame.__init__(self, parent, id, title, (-1, -1), wx.Size(450, 300))

        panel = wx.Panel(self, -1)
        box = wx.BoxSizer(wx.HORIZONTAL)
        box.Add(wx.Button(panel, -1, 'Buton1'), 1, wx.ALL, 5)
        box.Add(wx.Button(panel, -1, 'Buton2'), 0, wx.EXPAND)
```

```
        box.Add(wx.Button(panel, -1, 'Buton3'), 0, wx.ALIGN_CENTER)
        box.Add(wx.Button(panel, -1, 'Buton4'), 0, wx.TOP | wx.RIGHT | wx.LEFT,
50)
        panel.SetSizer(box)


class MyApp(wx.App):
    def OnInit(self):
        frame = MyFrame(None, -1, 'Utilizarea flag-urilor pentru margini si pentru
aliniere')
        frame.Show(True)
        return True

app = MyApp(0)
app.MainLoop()
```

Crearea unei interfeţe mai complexe necesită ca obiectele de tip **BoxSizer** să poată fi combinate, incluzând un **BoxSizer** în alt **BoxSizer**.

```
#ex9.py

import wx

class MyForm(wx.Frame):

    def __init__(self):
        wx.Frame.__init__(self, None, wx.ID_ANY, title='Controale imbricate')

        self.panel = wx.Panel(self, wx.ID_ANY)

        lblTitlu = wx.StaticText(self.panel, wx.ID_ANY, 'Formular inregistrare')

        lblNume = wx.StaticText(self.panel, wx.ID_ANY, 'Nume:')
        txtNume = wx.TextCtrl(self.panel, wx.ID_ANY, '')

        lblPrenume = wx.StaticText(self.panel, wx.ID_ANY, 'Prenume:')
        txtPrenume = wx.TextCtrl(self.panel, wx.ID_ANY, '')

        lblTelefon = wx.StaticText(self.panel, wx.ID_ANY, 'Telefon:')
        txtTelefon = wx.TextCtrl(self.panel, wx.ID_ANY, '')

        lblEmail = wx.StaticText(self.panel, wx.ID_ANY, 'E-mail:')
        txtEmail = wx.TextCtrl(self.panel, wx.ID_ANY, '')

        btnOK = wx.Button(self.panel, wx.ID_ANY, 'OK')
        btnCancel = wx.Button(self.panel, wx.ID_ANY, 'Cancel')

        self.Bind(wx.EVT_BUTTON, self.onOK, btnOK)
        self.Bind(wx.EVT_BUTTON, self.onCancel, btnCancel)

        sizerPrincipal   = wx.BoxSizer(wx.VERTICAL)
        sizerTitlu   = wx.BoxSizer(wx.HORIZONTAL)
        sizerNume    = wx.BoxSizer(wx.HORIZONTAL)
        sizerPrenume = wx.BoxSizer(wx.HORIZONTAL)
        sizerTelefon = wx.BoxSizer(wx.HORIZONTAL)
        sizerEmail   = wx.BoxSizer(wx.HORIZONTAL)
        btnSizer     = wx.BoxSizer(wx.HORIZONTAL)

        sizerTitlu.Add(lblTitlu, 0, wx.ALL, 5)

        sizerNume.Add(lblNume, 0, wx.ALL, 5)
        sizerNume.Add(txtNume, 1, wx.ALL|wx.EXPAND, 5)
```

```
        sizerPrenume.Add(lblPrenume, 0, wx.ALL, 5)
        sizerPrenume.Add(txtPrenume, 1, wx.ALL|wx.EXPAND, 5)

        sizerTelefon.Add(lblTelefon, 0, wx.ALL, 5)
        sizerTelefon.Add(txtTelefon, 1, wx.ALL|wx.EXPAND, 5)

        sizerEmail.Add(lblEmail, 0, wx.ALL, 5)
        sizerEmail.Add(txtEmail, 1, wx.ALL|wx.EXPAND, 5)

        btnSizer.Add(btnOK, 0, wx.ALL, 5)
        btnSizer.Add(btnCancel, 0, wx.ALL, 5)

        sizerPrincipal.Add(sizerTitlu, 0, wx.CENTER)
        sizerPrincipal.Add(wx.StaticLine(self.panel), 0, wx.ALL|wx.EXPAND, 5)
        sizerPrincipal.Add(sizerNume, 0, wx.ALL|wx.EXPAND, 5)
        sizerPrincipal.Add(sizerPrenume, 0, wx.ALL|wx.EXPAND, 5)
        sizerPrincipal.Add(sizerTelefon, 0, wx.ALL|wx.EXPAND, 5)
        sizerPrincipal.Add(sizerEmail, 0, wx.ALL|wx.EXPAND, 5)
        sizerPrincipal.Add(wx.StaticLine(self.panel), 0, wx.ALL|wx.EXPAND, 5)
        sizerPrincipal.Add(btnSizer, 0, wx.ALL|wx.CENTER, 5)

        self.panel.SetSizer(sizerPrincipal)
        sizerPrincipal.Fit(self)


    def onOK(self, event):
        print 'Ati apasat butonul OK!'

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


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

4. wxDialogs

În **wxPython** se pot folosi ferestre de dialog predefinite sau programatorii îşi pot crea propriile ferestre de dialog.

Există două tipuri de ferestre de dialog: **modale** şi **nemodale**. Cele modale nu permit ca utilizatorul să mai interacţioneze cu restul aplicaţiei până când acestea sunt distruse. Sunt create cu ajutorul funcţiei `ShowModal()`.

Ferestrele de dialog nemodale sunt apelate cu metoda `Show()`.

În ambele cazuri, metoda `Destroy()` este obligatorie. Ea şterge obiectul din memorie, iar dacă nu este distrus, scriptul nu va funcţiona corect.

Pentru a crea ferestre de dialog există două metode:

```
CreateTextSizer(self, string message)
CreateButtonSizer(self, long flags)
```

Metoda `CreateTextSizer()` creează un sizer ce afişează un text şi care poate conţine butoane paticularizate.

```
#ex10.py
```

```
import wx

class MyDialog(wx.Dialog):
    def __init__(self, parent, id, title):
        wx.Dialog.__init__(self, parent, id, title, size=(300,300))

        sizer =  self.CreateTextSizer('Acest TextSizer contine 5 butoane')
        sizer.Add(wx.Button(self, -1, 'Buton1'), 0, wx.ALL, 5)
        sizer.Add(wx.Button(self, -1, 'Buton2'), 0, wx.ALL, 5)
        sizer.Add(wx.Button(self, -1, 'Buton3'), 0, wx.ALL, 5)
        sizer.Add(wx.Button(self, -1, 'Buton4'), 0, wx.ALL|wx.ALIGN_CENTER, 5)
        sizer.Add(wx.Button(self, -1, 'Buton5'), 0, wx.ALL|wx.EXPAND, 5)
        self.SetSizer(sizer)

class MyFrame(wx.Frame):
    def __init__(self, parent, id, title):
        wx.Frame.__init__(self, parent, id, title, size=(400,400))

        panel = wx.Panel(self, -1)
        wx.Button(panel, 1, 'Deschide fereastra de dialog', (150,150))
        self.Bind (wx.EVT_BUTTON, self.OnTextSizerDialog, id=1)

    def OnTextSizerDialog(self, event):
        d = MyDialog(self, -1, 'Butoane')
        d.ShowModal()
        d.Destroy()

class MyApp(wx.App):
    def OnInit(self):
        frame = MyFrame(None, -1, 'TextSizer')
        frame.Show(True)
        frame.Centre()
        return True

app = MyApp(0)
app.MainLoop()
```

Metoda **`CreateButtonSizer()`** creează un dialog ce conţine butoane predefinite. Variantele sunt:

- wx.OK
- wx.CANCEL
- wx.YES
- wx.NO
- wx.HELP
- wx.NO_DEFAULT

```
#ex11.py

import wx

class MyDialog(wx.Dialog):
    def __init__(self, parent, id, title):
        wx.Dialog.__init__(self, parent, id, title)

        vbox = wx.BoxSizer(wx.VERTICAL)
        stline = wx.StaticText(self, 11, 'Aceasta fereastra de dialog contine
butoane predefinite')
        vbox.Add(stline, 1, wx.ALIGN_CENTER|wx.TOP, 45)
        sizer =  self.CreateButtonSizer(wx.NO|wx.YES|wx.HELP)
        vbox.Add(sizer, 0, wx.ALIGN_CENTER)
        self.SetSizer(vbox)
        self.Bind(wx.EVT_BUTTON, self.OnYes, id=wx.ID_YES)
```

```
    def OnYes(self, event):
        self.Close()

class MyFrame(wx.Frame):
    def __init__(self, parent, id, title):
        wx.Frame.__init__(self, parent, id, title)
        panel = wx.Panel(self, -1)
        wx.Button(panel, 1, 'Afiseaza fereastra de dialog', (50,50))
        self.Bind(wx.EVT_BUTTON, self.OnButtonSizer, id=1)

    def OnButtonSizer(self, event):
        d = MyDialog(self, -1, '')
        val = d.ShowModal()
        print d
        d.Destroy()

class MyApp(wx.App):
    def OnInit(self):
        frame = MyFrame(None, -1, 'ButtonTextSizer')
        frame.Show(True)
        frame.Centre()
        return True

app = MyApp(0)
app.MainLoop()
```

Ordinea flag-urilor nu este importantă, butoanele vor fi create în mod standard.

5. Ferestrele de dialog predefinite

wxPython oferă o serie de ferestre de dialog standard:

- wx.MessageDialog
- wx.ColourDialog
- wx.FileDialog
- wx.PageSetupDialog
- wx.FontDialog
- wx.DirDialog
- wx.SingleChoiceDialog
- wx.TextEntryDialog

```
#ex12.py

import wx
import os, sys

class MyFrame(wx.Frame):
    def __init__(self, parent, id, title):
      wx.Frame.__init__(self, parent, id, title)

      self.CreateStatusBar()
      menuBar = wx.MenuBar()
      menu = wx.Menu()
      menu.Append(99,  "&Message Dialog", "Shows a Message Dialog")
      menu.Append(100, "&Color Dialog", "Shows a Color Dialog")
      menu.Append(101, "&File Dialog", "Shows a File Dialog")
      menu.Append(102, "&Page Setup Dialog", "Shows a Page Setup Dialog")
      menu.Append(103, "&Font Dialog", "Shows a Font Dialog")
      menu.Append(104, "&Directory Dialog", "Shows a Directory Dialog")
      menu.Append(105, "&SingleChoice Dialog", "Shows a SingleChoice Dialog")
      menu.Append(106, "&TextEntry Dialog", "Shows a TextEntry Dialog")
```

```
      menuBar.Append(menu, "&Dialogs")
      self.SetMenuBar(menuBar)

      self.Bind(wx.EVT_MENU, self.message, id=99)
      self.Bind(wx.EVT_MENU, self.choosecolor, id=100)
      self.Bind(wx.EVT_MENU, self.openfile, id=101)
      self.Bind(wx.EVT_MENU, self.pagesetup, id=102)
      self.Bind(wx.EVT_MENU, self.choosefont, id=103)
      self.Bind(wx.EVT_MENU, self.opendir, id=104)
      self.Bind(wx.EVT_MENU, self.singlechoice, id=105)
      self.Bind(wx.EVT_MENU, self.textentry, id=106)

    def message(self, event):
        dlg = wx.MessageDialog(self, 'To save one life is as if you have saved
the world.', 'Talmud', wx.OK|wx.ICON_INFORMATION)
        dlg.ShowModal()
        dlg.Destroy()

    def choosecolor(self, event):
        dlg = wx.ColourDialog(self)
        dlg.GetColourData().SetChooseFull(True)
        if dlg.ShowModal() == wx.ID_OK:
            data = dlg.GetColourData()
            self.SetStatusText('You          selected:          %s\n'          %
str(data.GetColour().Get()))
        dlg.Destroy()

    def openfile(self, event):
       dlg = wx.FileDialog(self, "Choose a file", os.getcwd(), "", "*.*", wx.OPEN)
       if dlg.ShowModal() == wx.ID_OK:
                path = dlg.GetPath()
                mypath = os.path.basename(path)
                self.SetStatusText("You selected: %s" % mypath)
       dlg.Destroy()

    def pagesetup(self, event):
        dlg = wx.PageSetupDialog(self)
        if dlg.ShowModal() == wx.ID_OK:
            data = dlg.GetPageSetupData()
            tl = data.GetMarginTopLeft()
            br = data.GetMarginBottomRight()
            self.SetStatusText('Margins are: %s %s' % (str(tl), str(br)))
        dlg.Destroy()

    def choosefont(self, event):
        default_font  =  wx.Font(10,  wx.SWISS  ,  wx.NORMAL,  wx.NORMAL,  False,
"Verdana")
        data = wx.FontData()
        if sys.platform == 'win32':
            data.EnableEffects(True)
        data.SetAllowSymbols(False)
        data.SetInitialFont(default_font)
        data.SetRange(10, 30)
        dlg = wx.FontDialog(self, data)
        if dlg.ShowModal() == wx.ID_OK:
            data = dlg.GetFontData()
            font = data.GetChosenFont()
            color = data.GetColour()
            text  =  'Face:  %s,  Size:  %d,  Color:  %s'  %  (font.GetFaceName(),
font.GetPointSize(),  color.Get())
            self.SetStatusText(text)
        dlg.Destroy()
```

```
    def opendir(self, event):
        dlg = wx.DirDialog(self, "Choose a directory:", style=wx.DD_DEFAULT_STYLE
| wx.DD_NEW_DIR_BUTTON)
        if dlg.ShowModal() == wx.ID_OK:
            self.SetStatusText('You selected: %s\n' % dlg.GetPath())
        dlg.Destroy()

    def singlechoice(self, event):
        sins = ['Greed', 'Lust', 'Gluttony', 'Pride', 'Sloth', 'Envy', 'Wrath']
        dlg = wx.SingleChoiceDialog(self, 'Seven deadly sins', 'Which one?', sins,
wx.CHOICEDLG_STYLE)
        if dlg.ShowModal() == wx.ID_OK:
            self.SetStatusText('You chose: %s\n' % dlg.GetStringSelection())
        dlg.Destroy()

    def textentry(self, event):
        dlg = wx.TextEntryDialog(self, 'Enter some text','Text Entry')
        dlg.SetValue("Default")
        if dlg.ShowModal() == wx.ID_OK:
            self.SetStatusText('You entered: %s\n' % dlg.GetValue())
        dlg.Destroy()

class MyApp(wx.App):
    def OnInit(self):
        myframe = MyFrame(None, -1, "Ferestre de dialog standard")
        myframe.CenterOnScreen()
        myframe.Show(True)
        return True

app = MyApp(0)
app.MainLoop()
```

6. Controalele de bază în Python

6.1 wx.Button

wx.Button este un control simplu care conţine un text şi este folosit pentru a genera anumite acţiuni. Stilurile controlului **wx.Button** sunt:

- wx.BU_LEFT - text aliniat la stânga
- wx.BU_TOP - text aliniat în partea de sus
- wx.BU_RIGHT - text aliniat la dreapta
- wx.BU_BOTTOM - text aliniat în partea de jos
- wx.BU_EXACTFIT - butonul este redimensionat în funcţie de text
- wx.NO_BORDER - fără borduri

```
#ex13.py

import wx
import random

APP_SIZE_X = 300
APP_SIZE_Y = 200

class MyButtons(wx.Dialog):
    def __init__(self, parent, id, title):
        wx.Dialog.__init__(self,    parent,    id,    title,    size=(APP_SIZE_X,
APP_SIZE_Y))

        wx.Button(self, 1, 'Close', (50, 130), style=wx.BU_EXACTFIT)
        wx.Button(self, 2, 'Random Move', (150, 130), (110, -1))

        self.Bind(wx.EVT_BUTTON, self.OnClose, id=1)
        self.Bind(wx.EVT_BUTTON, self.OnRandomMove, id=2)
```

```
        self.Centre()
        self.ShowModal()
        self.Destroy()

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

    def OnRandomMove(self, event):
        screensize = wx.GetDisplaySize()
        randx = random.randrange(0, screensize.x - APP_SIZE_X)
        randy = random.randrange(0, screensize.y - APP_SIZE_Y)
        self.Move((randx, randy))

app = wx.App(0)
MyButtons(None, -1, Butoane')
app.MainLoop()
```

6.2 wx.ToggleButton

Butoanele de tip **ToggleButton** sunt butoane care au două stări: **activat** sau **neactivat**. Un exemplu este prezentat în scriptul de mai jos:

```
#ex14.py

import wx

class ToggleButtons(wx.Dialog):
    def __init__(self, parent, id, title):
        wx.Dialog.__init__(self, parent, id, title, size=(300, 200))

        self.culoare = wx.Colour(0, 0, 0)

        wx.ToggleButton(self, 1, 'Rosu', (20, 25))
        wx.ToggleButton(self, 2, 'Verde', (20, 60))
        wx.ToggleButton(self, 3, 'Albastru', (20, 100))

        self.panel      =   wx.Panel(self,   -1,   (150,   20),   (110,   110),
style=wx.SUNKEN_BORDER)
        self.panel.SetBackgroundColour("black")

        self.Bind(wx.EVT_TOGGLEBUTTON, self.SchimbaInRosu, id=1)
        self.Bind(wx.EVT_TOGGLEBUTTON, self.SchimbaInVerde, id=2)
        self.Bind(wx.EVT_TOGGLEBUTTON, self.SchimbaInAlbastru, id=3)

        self.Centre()
        self.ShowModal()
        self.Destroy()

    def SchimbaInRosu(self, event):
        culoareVerde = self.culoare.Green()
        culoareAlbastra = self.culoare.Blue()
        if  self.culoare.Red():
            self.culoare.Set(0, culoareVerde, culoareAlbastra)
        else:
            self.culoare.Set(255, culoareVerde, culoareAlbastra)
        self.panel.SetBackgroundColour(self.culoare)
        self.panel.Refresh()


    def SchimbaInVerde(self, event):
        culoareRosie = self.culoare.Red()
        culoareAlbastra = self.culoare.Blue()
        if  self.culoare.Green():
```

```
            self.culoare.Set(culoareRosie, 0, culoareAlbastra)
        else:
            self.culoare.Set(culoareRosie, 255, culoareAlbastra)
        self.panel.SetBackgroundColour(self.culoare)
        self.panel.Refresh()

    def SchimbaInAlbastru(self, event):
        culoareRosie = self.culoare.Red()
        culoareVerde = self.culoare.Green()
        if  self.culoare.Blue():
            self.culoare.Set(culoareRosie, culoareVerde, 0)
        else:
            self.culoare.Set(culoareRosie, culoareVerde, 255)
        self.panel.SetBackgroundColour(self.culoare)
        self.panel.Refresh()


app = wx.App(0)
ToggleButtons(None, -1, 'Combinatii intre culori')
app.MainLoop()
```

6.3 wx.StaticBox

wx.StaticBox este un control pentru decorarea frame-urilor, dar este folosit şi pentru a grupa diferite controale. El trebuie creat înainte de a crea obiectele pe care le conţine. Controalele grupate într-un **StaticBox** nu trebuie să fie controale de tip child.

```
#ex15.py

import wx

class MyDialog(wx.Dialog):
    def __init__(self, parent, id, title):
        wx.Dialog.__init__(self, parent, id, title, size=(270, 270))

        wx.StaticBox(self, -1, 'Informatii personale', (5, 5), size=(240, 170))
        wx.CheckBox(self, -1 ,'Somer', (15, 30))
        wx.CheckBox(self, -1 ,'Casatorit', (15, 55))
        wx.StaticText(self, -1, 'Varsta', (15, 95))
        wx.SpinCtrl(self, -1, '1', (55, 90), (60, -1), min=1, max=120)
        wx.Button(self, 1, 'Ok', (90, 185), (60, -1))

        self.Bind(wx.EVT_BUTTON, self.OnClose, id=1)

        self.Centre()
        self.ShowModal()
        self.Destroy()

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

app = wx.App(0)
MyDialog(None, -1, 'StaticBox')
app.MainLoop()
```

6.4 wx.ComboBox

Controlul **wx.ComboBox** reprezintă o combinaţie între un buton, un câmp de tip text şi un listbox. Utilizatorii pot selecta o singură opţiune dintr-o listă de tip string.

Constructorul clasei **wx.ComboBox** este:

```
wx.ComboBox(int id, string value='', wx.Point pos=wx.DefaultPosition, wx.Size
size=wx.DefaultSize,   wx.List choices=wx.EmptyList, int style=0, wx.Validator
validator=wx.DefaultValidator, string name=wx.ComboBoxNameStr)
```

wx.ComboBox poate avea următoarele stiluri:

- wx.CB_DROPDOWN
- wx.CB_READONLY
- wx.CB_SORT

```
#ex16.py

import wx

class MyDialog(wx.Dialog):
    def __init__(self, parent, id, title):
        wx.Dialog.__init__(self, parent, id, title, size=(250, 250))

        lista = ['', 'Ianuarie', 'Februarie', 'Martie', 'Aprilie', 'Mai', 'Iunie',
'Iulie', 'August', 'Septembrie', 'Octombrie','Noiembrie', 'Decembrie' ]


        wx.ComboBox(self,  -1,  pos=(50,  150),  size=(150,  -1),  choices=lista,
style=wx.CB_READONLY)
        wx.Button(self, 1, 'Close', (80, 220))

        self.Bind(wx.EVT_BUTTON, self.OnClose, id=1)
        self.Bind(wx.EVT_COMBOBOX, self.OnSelect)

        self.Centre()

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

    def OnSelect(self, event):
        self.txt=wx.StaticText(self,            -1,            event.GetString(),
style=wx.ALIGN_CENTRE)


class MyApp(wx.App):
    def OnInit(self):
        dlg = MyDialog(None, -1, 'Lunile anului')
        dlg.ShowModal()
        dlg.Destroy()
        return True

app = MyApp(0)
app.MainLoop()
```

6.5 wx.CheckBox

Controlul **wx.CheckBox** poate avea două stări: **On** şi **Off**.

```
#ex17.py

import wx

class MyCheckBox(wx.Frame):
    def __init__(self, parent, id, title):
        wx.Frame.__init__(self, parent, id, title, size=(300, 170))

        panel = wx.Panel(self, -1)
        self.cb = wx.CheckBox(panel, -1, 'Afiseaza titlul', (10, 10))
```

```
        self.cb.SetValue(True)

        wx.EVT_CHECKBOX(self, self.cb.GetId(), self.ShowTitle)

        self.Show()
        self.Centre()

    def ShowTitle(self, event):
        if self.cb.GetValue():
            self.SetTitle('Ati bifat casuta!')
        else: self.SetTitle('')


app = wx.App(0)
MyCheckBox(None, -1, 'Acesta este titlul ferestrei')
app.MainLoop()
```

6.6 wx.RadioButton

Controlul **wx.RadioButton** permite ca utilizatorul să aleagă o singură opţiune dintr-un grup de opţiuni. Un grup de buoane radio este definit prin aplicarea stilului **wx.RB_GROUP primului buton** radio din cadrul grupului. Toate celelalte botoane radio din cadrul grupului se definesc după acesta. Prin declararea unui nou buton cu stilul **wx.RB_GROUP** se va crea un nou grup de butoane radio.

```
#ex18.py

import wx

class MyFrame(wx.Frame):
    def __init__(self, parent, id, title):
        wx.Frame.__init__(self,   parent,   id,   title,   wx.DefaultPosition,
wx.Size(200, 150))
        panel = wx.Panel(self, -1)
        self.rb1   =   wx.RadioButton(panel,   -1,   'Valoare   A',   (10,   10),
style=wx.RB_GROUP)
        self.rb2 = wx.RadioButton(panel, -1, 'Valoare B', (10, 30))
        self.rb3 = wx.RadioButton(panel, -1, 'Valoare C', (10, 50))
        self.Bind(wx.EVT_RADIOBUTTON, self.SetVal, id=self.rb1.GetId())
        self.Bind(wx.EVT_RADIOBUTTON, self.SetVal, id=self.rb2.GetId())
        self.Bind(wx.EVT_RADIOBUTTON, self.SetVal, id=self.rb3.GetId())
        self.statusbar = self.CreateStatusBar(3)
        self.SetVal(True)

    def SetVal(self, event):
        state1 = str(self.rb1.GetValue())
        state2 = str(self.rb2.GetValue())
        state3 = str(self.rb3.GetValue())
        self.statusbar.SetStatusText(state1,0)
        self.statusbar.SetStatusText(state2,1)
        self.statusbar.SetStatusText(state3,2)

class MyApp(wx.App):
    def OnInit(self):
        frame = MyFrame(None, -1, 'Butoane Radio')
        frame.Show(True)
        frame.Center()
        return True

app = MyApp(0)
app.MainLoop()
```

6.7 wx.ListBox

Controlul **wx.Listbox** poate fi creat în două moduri: cu o singură selecţie sau cu selecţie multiplă.

```
#ex18.py

import wx

from time import *

class MyFrame(wx.Frame):
    def __init__(self, parent, id, title):
        wx.Frame.__init__(self, parent, id, title, wx.DefaultPosition, (550,
350))

        zone_list = ['CET', 'GMT', 'MSK', 'EST', 'PST', 'EDT']
        self.full_list = {
            'CET': 'Europa Centrala',
            'GMT': 'Londra',
            'MSK': 'Moscova',
            'EST': 'Orient',
            'PST': 'Pacific',
            'EDT': 'Daylight Time'
        }
        self.time_diff = {
            'CET' : 1,
            'GMT' : 0,
            'MSK': 3,
            'EST': -5,
            'PST': -8,
            'EDT': -4
        }

        vbox = wx.BoxSizer(wx.VERTICAL)
        hbox1 = wx.BoxSizer(wx.HORIZONTAL)
        hbox2 = wx.BoxSizer(wx.HORIZONTAL)
        hbox3 = wx.BoxSizer(wx.HORIZONTAL)

        self.timer = wx.Timer(self, 1)
        self.diff = 0
        panel = wx.Panel(self, -1)
        self.time_zones = wx.ListBox(panel, 26, wx.DefaultPosition, (170, 130),
zone_list, wx.LB_SINGLE)
        self.time_zones.SetSelection(0)
        self.text = wx.TextCtrl(panel, -1, 'Europa centrala', size=(200, 130),
style=wx.TE_MULTILINE)
        self.time = wx.StaticText(panel, -1, '')
        btn = wx.Button(panel, wx.ID_CLOSE, 'Inchide')
        hbox1.Add(self.time_zones, 0, wx.TOP, 40)
        hbox1.Add(self.text, 1, wx.LEFT | wx.TOP, 40)
        hbox2.Add(self.time, 1, wx.ALIGN_CENTRE)
        hbox3.Add(btn, 0, wx.ALIGN_CENTRE)
        vbox.Add(hbox1, 0, wx.ALIGN_CENTRE)
        vbox.Add(hbox2, 1, wx.ALIGN_CENTRE)
        vbox.Add(hbox3, 1, wx.ALIGN_CENTRE)
        panel.SetSizer(vbox)
        self.timer.Start(100)

        self.Bind(wx.EVT_BUTTON, self.OnClose, id=wx.ID_CLOSE)
        self.Bind(wx.EVT_LISTBOX, self.OnSelect, id=26)
        self.Bind(wx.EVT_TIMER, self.OnTimer, id=1)
```

```
    def OnClose(self, event):
        self.Close()

    def OnSelect(self, event):
        index = event.GetSelection()
        time_zone = self.time_zones.GetString(index)
        self.diff = self.time_diff[time_zone]
        self.text.SetValue(self.full_list[time_zone])

    def OnTimer(self, event):
        ct = gmtime()
        print_time = (ct[0], ct[1], ct[2], ct[3]+self.diff, ct[4], ct[5], ct[6],
ct[7], -1)
        self.time.SetLabel(strftime("%H:%M:%S", print_time))

class MyApp(wx.App):
    def OnInit(self):
        frame = MyFrame(None, -1, 'Ora exacta pe diverse meridiane')
        frame.Centre()
        frame.Show(True)
        return True

app = MyApp(0)
app.MainLoop()
```

6.8 wx.ListCtrl

wx.ListCtrl se foloseşte pentru crearea listelor în următoarele formate:

- report view
- list view
- icon view

```
#ex19.py

import wx

class MyDialog(wx.Dialog):
    def __init__(self, parent, id, title):
        wx.Dialog.__init__(self,    parent,    id,    title,    size=(600,500),
style=wx.DEFAULT_DIALOG_STYLE)

        hbox  = wx.BoxSizer(wx.HORIZONTAL)
        vbox1 = wx.BoxSizer(wx.VERTICAL)
        vbox2 = wx.BoxSizer(wx.VERTICAL)
        vbox3 = wx.GridSizer(2,2,0,0)
        vbox4 = wx.BoxSizer(wx.VERTICAL)
        pnl1 = wx.Panel(self, -1, style=wx.SIMPLE_BORDER)
        pnl2 = wx.Panel(self, -1, style=wx.SIMPLE_BORDER)
        self.lc = wx.ListCtrl(self, -1, style=wx.LC_REPORT)
        self.lc.InsertColumn(0, 'Nume si prenume')
        self.lc.InsertColumn(1, 'Telefon')
        self.lc.SetColumnWidth(0, 140)
        self.lc.SetColumnWidth(1, 153)
        vbox1.Add(pnl1, 1, wx.EXPAND | wx.ALL, 3)
        vbox1.Add(pnl2, 1, wx.EXPAND | wx.ALL, 3)
        vbox2.Add(self.lc, 1, wx.EXPAND | wx.ALL, 3)
        self.tc1 = wx.TextCtrl(pnl1, -1)
        self.tc2 = wx.TextCtrl(pnl1, -1)
        vbox3.AddMany([   (wx.StaticText(pnl1,   -1,   'Nume   si   prenume'),0,
wx.ALIGN_CENTER),
                          (self.tc1, 0, wx.ALIGN_LEFT|wx.ALIGN_CENTER_VERTICAL),
                          (wx.StaticText(pnl1,          -1,          'Telefon'),0,
wx.ALIGN_CENTER_HORIZONTAL),
```

```
                        (self.tc2,0)])
        pnl1.SetSizer(vbox3)
        vbox4.Add(wx.Button(pnl2, 10, 'Adauga'),    0, wx.ALIGN_CENTER| wx.TOP,
45)
        vbox4.Add(wx.Button(pnl2, 11, 'Sterge'), 0, wx.ALIGN_CENTER|wx.TOP, 15)
        vbox4.Add(wx.Button(pnl2,  12,  'Goleste  lista'),  0,  wx.ALIGN_CENTER|
wx.TOP, 15)
        vbox4.Add(wx.Button(pnl2, 13, 'Iesire'), 0, wx.ALIGN_CENTER| wx.TOP, 15)
        pnl2.SetSizer(vbox4)
        self.Bind (wx.EVT_BUTTON, self.OnAdd, id=10)
        self.Bind (wx.EVT_BUTTON, self.OnRemove, id=11)
        self.Bind (wx.EVT_BUTTON, self.OnClear, id=12)
        self.Bind (wx.EVT_BUTTON, self.OnClose, id=13)
        hbox.Add(vbox1, 1, wx.EXPAND)
        hbox.Add(vbox2, 1, wx.EXPAND)
        self.SetSizer(hbox)

    def OnAdd(self, event):
        if not self.tc1.GetValue() or not self.tc2.GetValue():
            return
        num_items = self.lc.GetItemCount()
        self.lc.InsertStringItem(num_items, self.tc1.GetValue())
        self.lc.SetStringItem(num_items, 1, self.tc2.GetValue())
        self.tc1.Clear()
        self.tc2.Clear()

    def OnRemove(self, event):
        index = self.lc.GetFocusedItem()
        self.lc.DeleteItem(index)

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

    def OnClear(self, event):
        self.lc.DeleteAllItems()

class MyApp(wx.App):
    def OnInit(self):
        dia = MyDialog(None, -1, 'Lista persoane')
        dia.ShowModal()
        dia.Destroy()
        return True

app = MyApp(0)
app.MainLoop()
```