Desktop GUIs

with wxPython

Why Desktop GUIs?

- Command line foreign to most users
- Lots of widgets
- Logical layout
- Self contained
  - No server required
  - Stable – control over changes
- One language, not four for web
- I like them

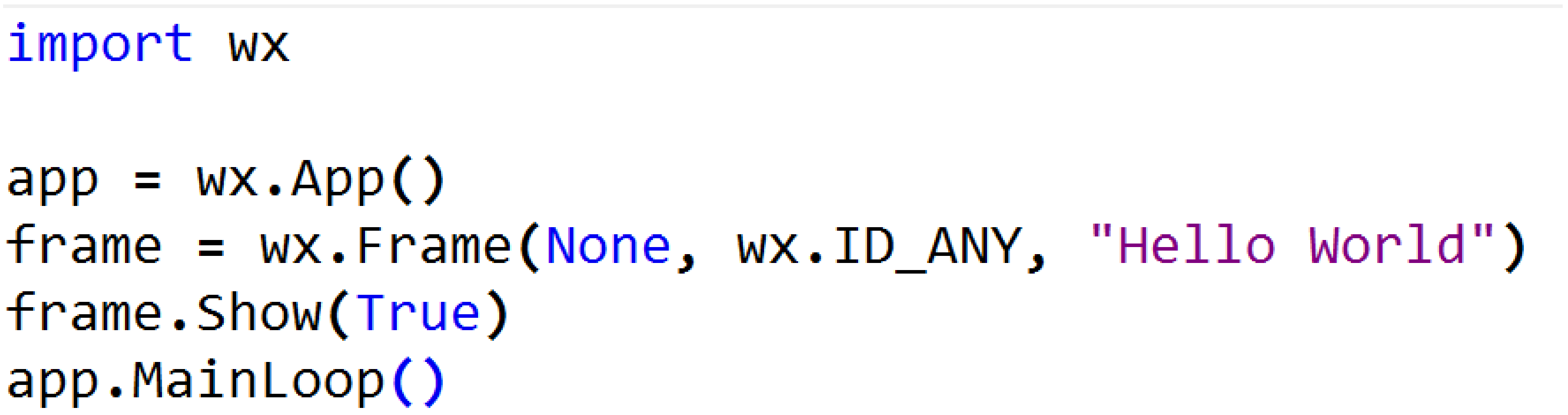
What is WxPython?

- Python bindings for cross-platform WxWidgets toolkit
- WxWidgets written in C++
- Mature (WxWidgets 1992, wxPython 2003)
- LGPL-like license

Why wxPython?

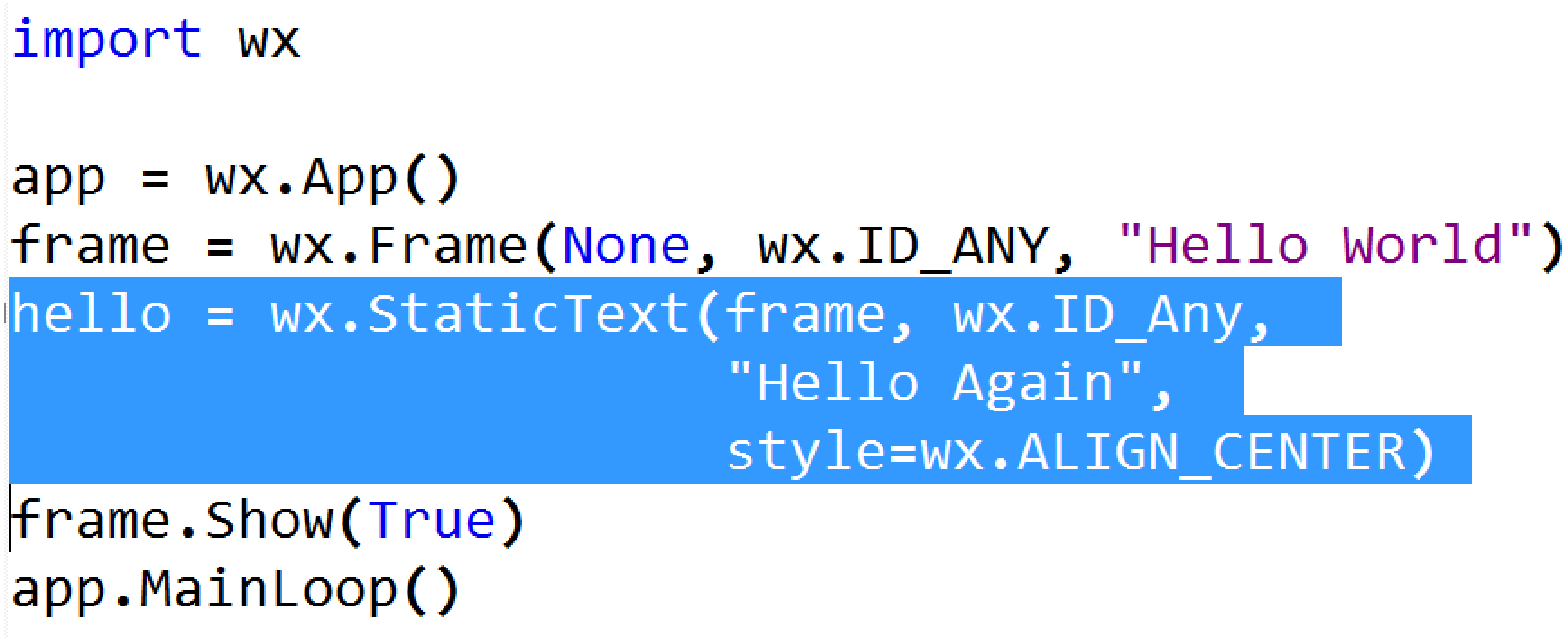
- Uses native rendering for native look
- Large library of widgets
- No license issues
- Good support, demos, tutorials, etc.
- Designer programs available
- Python 2.x with experimental 3.x support

Hello World



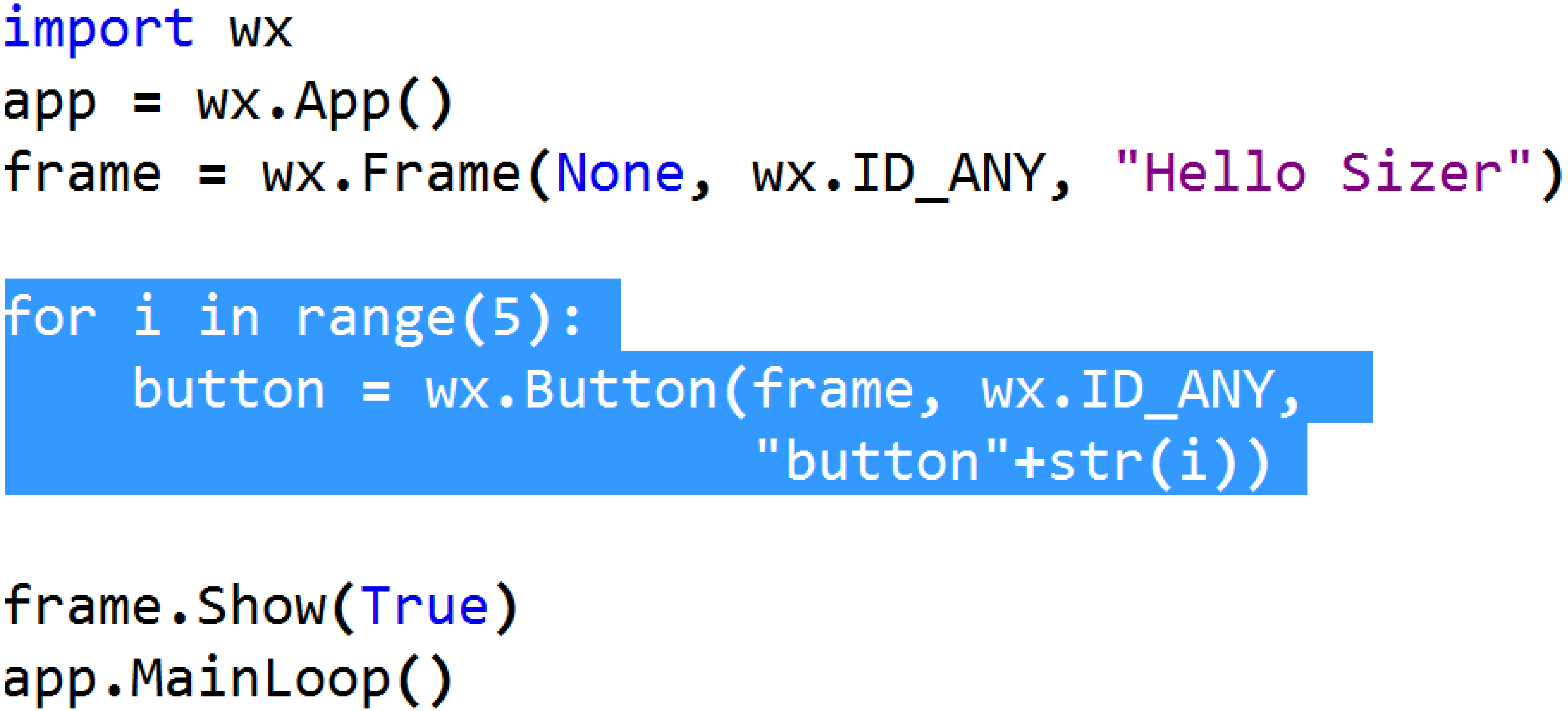
```
import wx

app = wx.App()
frame = wx.Frame(None, wx.ID_ANY, "Hello World")
frame.Show(True)
app.MainLoop()
```

Hello Again



```
import wx

app = wx.App()
frame = wx.Frame(None, wx.ID_ANY, "Hello World")
hello = wx.StaticText(frame, wx.ID_Any,
                      "Hello Again",
                      style=wx.ALIGN_CENTER)
frame.Show(True)
app.MainLoop()
```

More Widgets

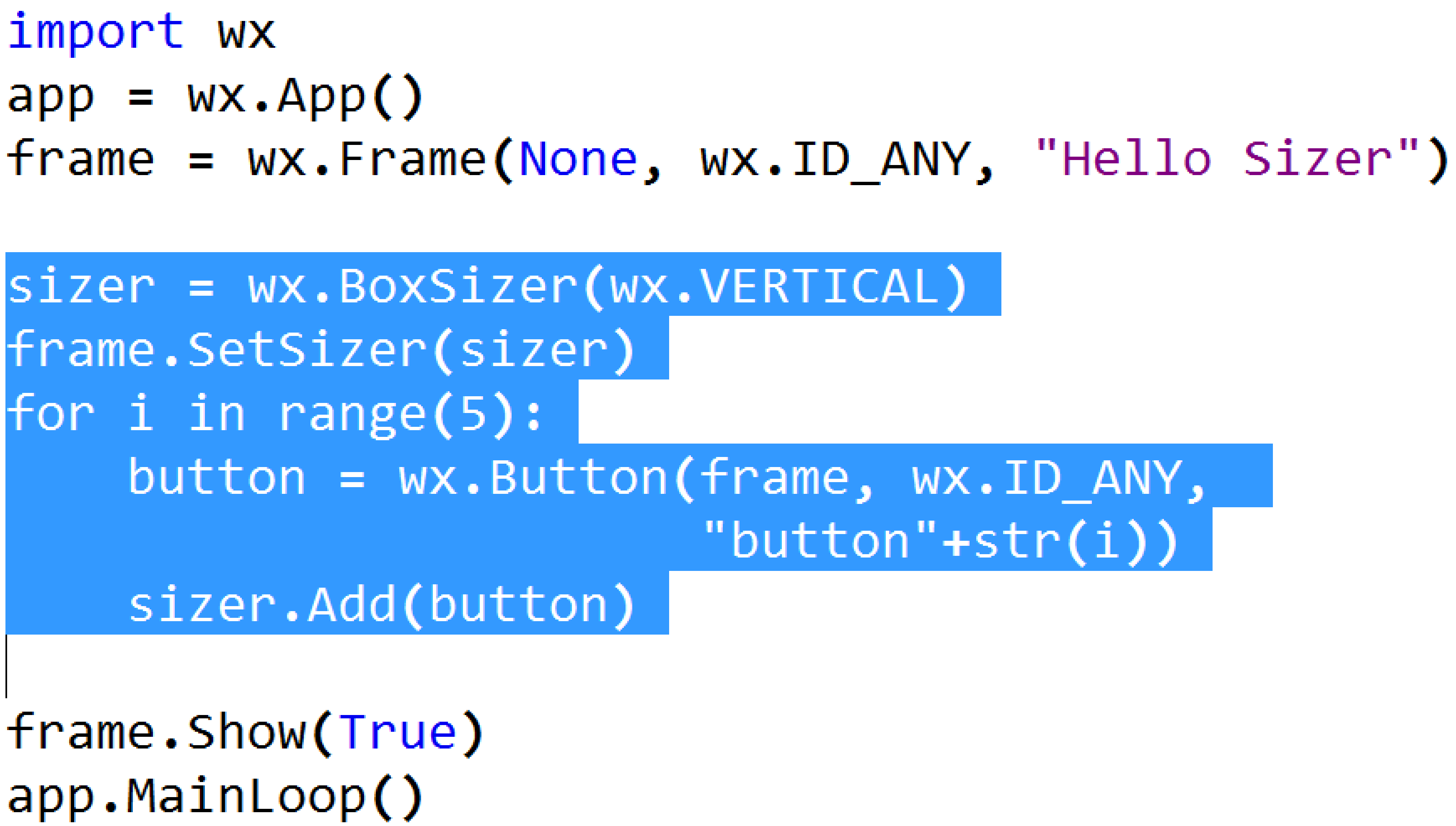


```
import wx
app = wx.App()
frame = wx.Frame(None, wx.ID_ANY, "Hello Sizer")

for i in range(5):
    button = wx.Button(frame, wx.ID_ANY,
                       "button"+str(i))

frame.Show(True)
app.MainLoop()
```

Sizers

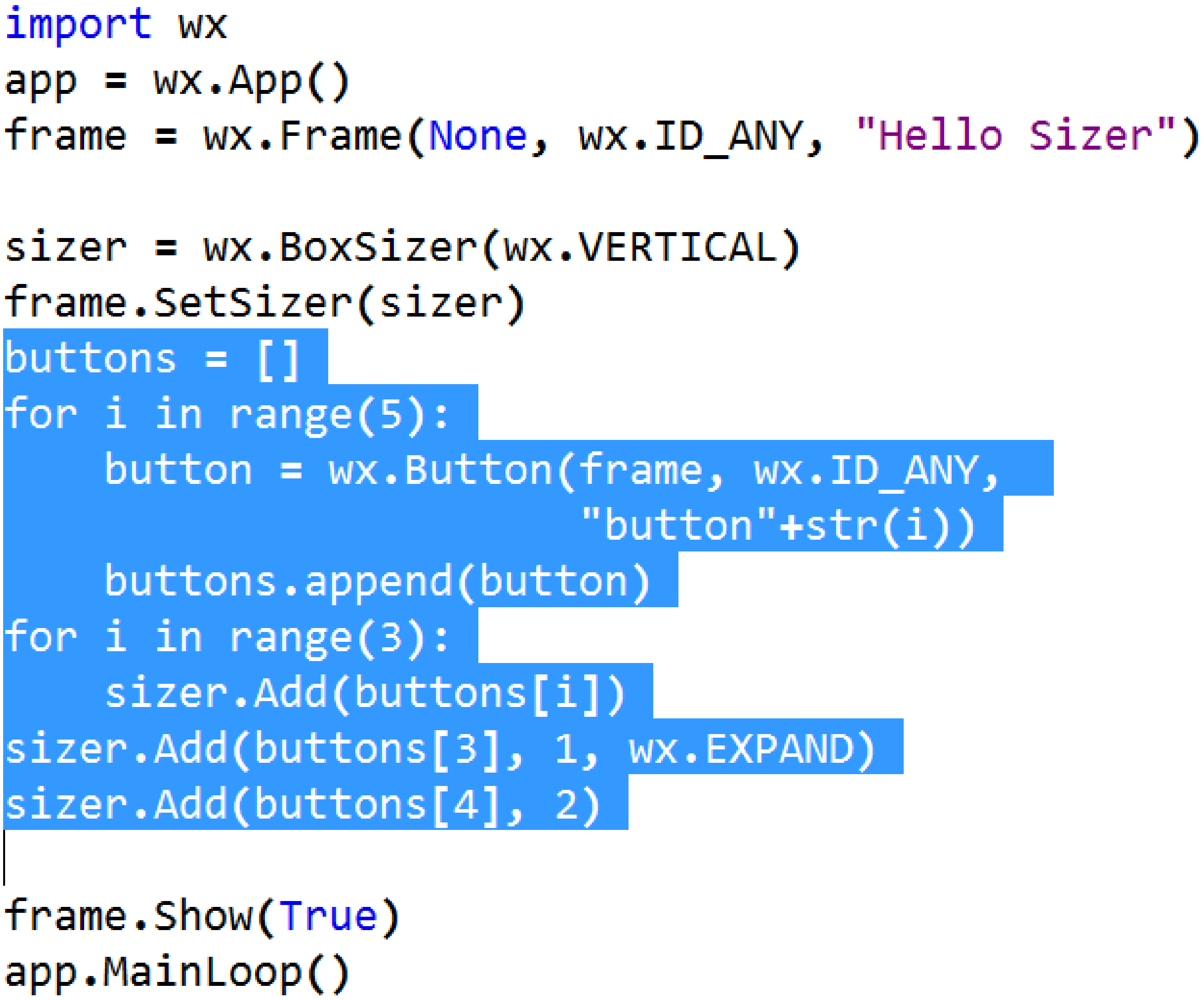
- Central to controlling layout
- Controls widget relationships on window creation or re-size
- Add sizer to parent, widgets to sizer
- Main types
  - Wrap (flow as needed)
  - Box (horz/vert line of widgets)
  - StaticBox (add border/caption)
  - Grid (fixed table)
  - FlexGrid (adjustable table)
  - GridBag (row/col spanning)

Hello Sizer



```
import wx
app = wx.App()
frame = wx.Frame(None, wx.ID_ANY, "Hello Sizer")

sizer = wx.BoxSizer(wx.VERTICAL)
frame.SetSizer(sizer)
for i in range(5):
    button = wx.Button(frame, wx.ID_ANY,
                       "button"+str(i))
    sizer.Add(button)

frame.Show(True)
app.MainLoop()
```

Stretchy Widgets



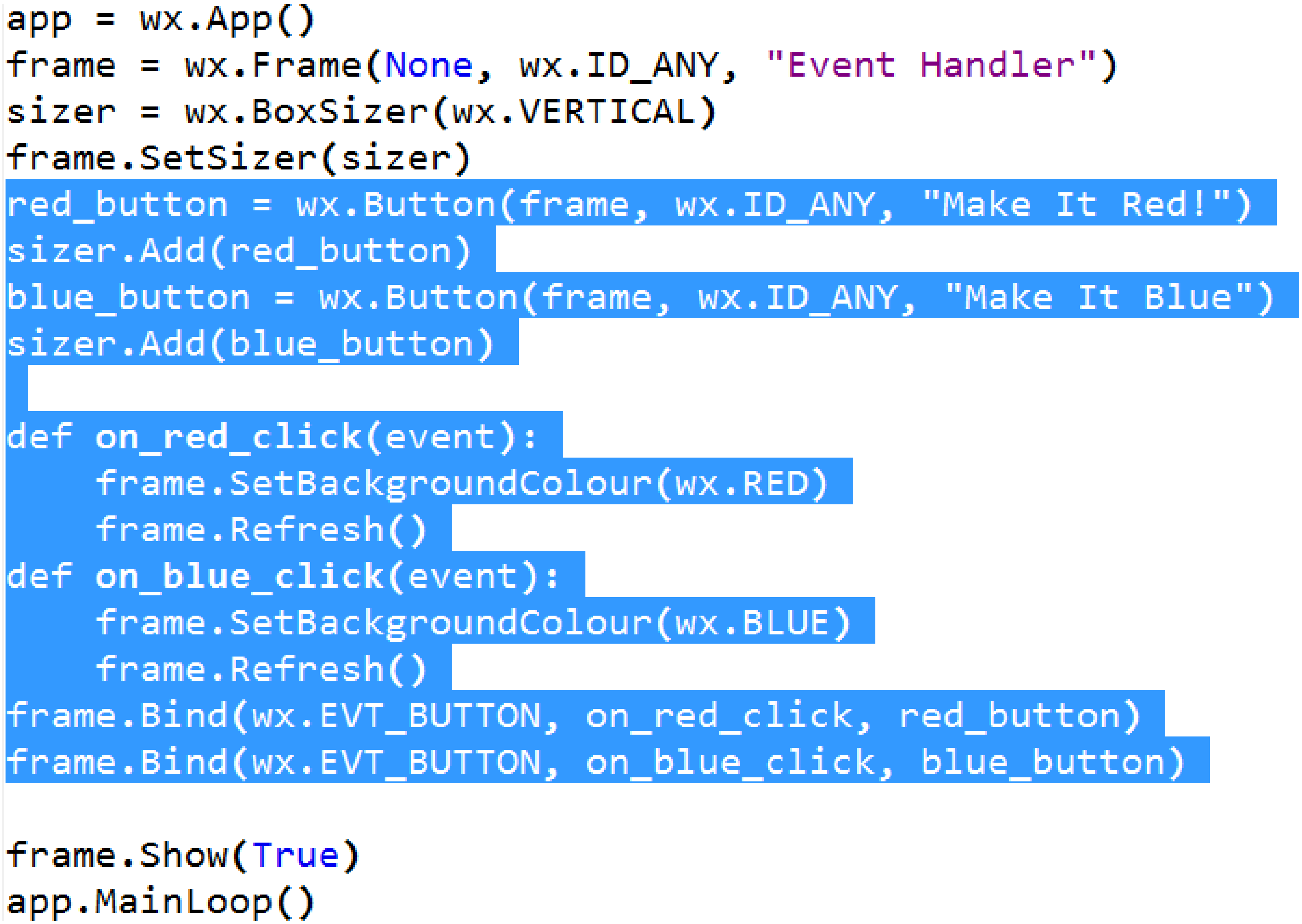
```
import wx
app = wx.App()
frame = wx.Frame(None, wx.ID_ANY, "Hello Sizer")

sizer = wx.BoxSizer(wx.VERTICAL)
frame.SetSizer(sizer)
buttons = []
for i in range(5):
    button = wx.Button(frame, wx.ID_ANY,
                       "button"+str(i))
    buttons.append(button)
for i in range(3):
    sizer.Add(buttons[i])
sizer.Add(buttons[3], 1, wx.EXPAND)
sizer.Add(buttons[4], 2)

frame.Show(True)
app.MainLoop()
```

Event Handlers

- GUIs are event driven
- Most widgets trigger events of various types
- When event is triggered by a widget, a function is called to handle it
- Need to 'Bind' a function to an event from a widget:

```
frame.Bind(wx.EVT_BUTTON, function, widget)
```

Click Event

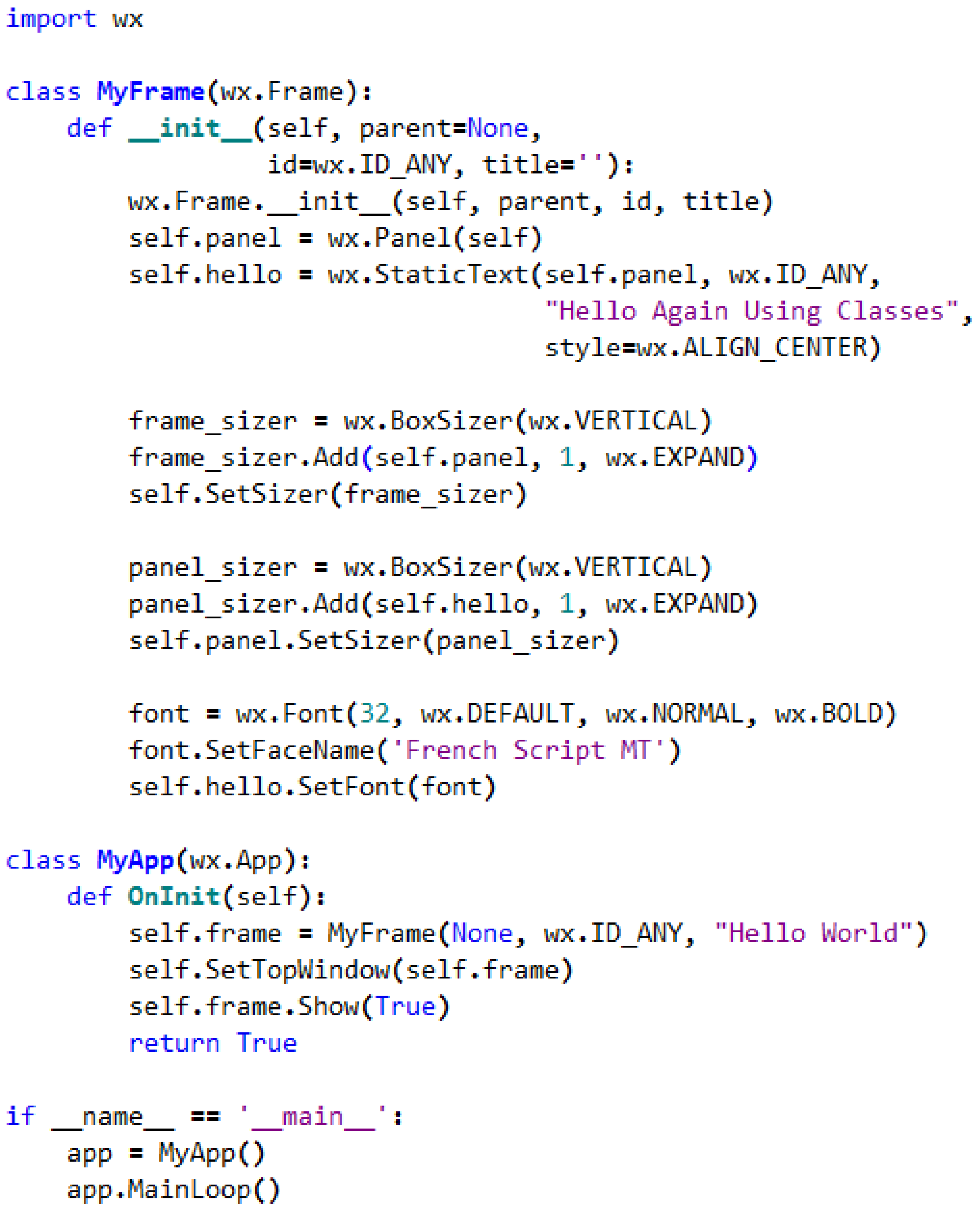


```python
app = wx.App()
frame = wx.Frame(None, wx.ID_ANY, "Event Handler")
sizer = wx.BoxSizer(wx.VERTICAL)
frame.SetSizer(sizer)
red_button = wx.Button(frame, wx.ID_ANY, "Make It Red!")
sizer.Add(red_button)
blue_button = wx.Button(frame, wx.ID_ANY, "Make It Blue")
sizer.Add(blue_button)

def on_red_click(event):
    frame.SetBackgroundColour(wx.RED)
    frame.Refresh()
def on_blue_click(event):
    frame.SetBackgroundColour(wx.BLUE)
    frame.Refresh()
frame.Bind(wx.EVT_BUTTON, on_red_click, red_button)
frame.Bind(wx.EVT_BUTTON, on_blue_click, blue_button)

frame.Show(True)
app.MainLoop()
```

Use Classes

- Have shown a procedural approach. Don't.
- Normally use Classes, particularly with more complex apps.
- Gives clean, separated namespaces
- Typically create own App class as subclass of wx.App
- Windows as subclasses of wx.Frame
- When using builder software, import generated code and subclass/override in your own module

Class Example



```
import wx

class MyFrame(wx.Frame):
    def __init__(self, parent=None,
                 id=wx.ID_ANY, title=''):
        wx.Frame.__init__(self, parent, id, title)
        self.panel = wx.Panel(self)
        self.hello = wx.StaticText(self.panel, wx.ID_ANY,
                                   "Hello Again Using Classes",
                                   style=wx.ALIGN_CENTER)

        frame_sizer = wx.BoxSizer(wx.VERTICAL)
        frame_sizer.Add(self.panel, 1, wx.EXPAND)
        self.SetSizer(frame_sizer)

        panel_sizer = wx.BoxSizer(wx.VERTICAL)
        panel_sizer.Add(self.hello, 1, wx.EXPAND)
        self.panel.SetSizer(panel_sizer)

        font = wx.Font(32, wx.DEFAULT, wx.NORMAL, wx.BOLD)
        font.SetFaceName('French Script MT')
        self.hello.SetFont(font)

class MyApp(wx.App):
    def OnInit(self):
        self.frame = MyFrame(None, wx.ID_ANY, "Hello World")
        self.SetTopWindow(self.frame)
        self.frame.Show(True)
        return True

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

wxPython Demo

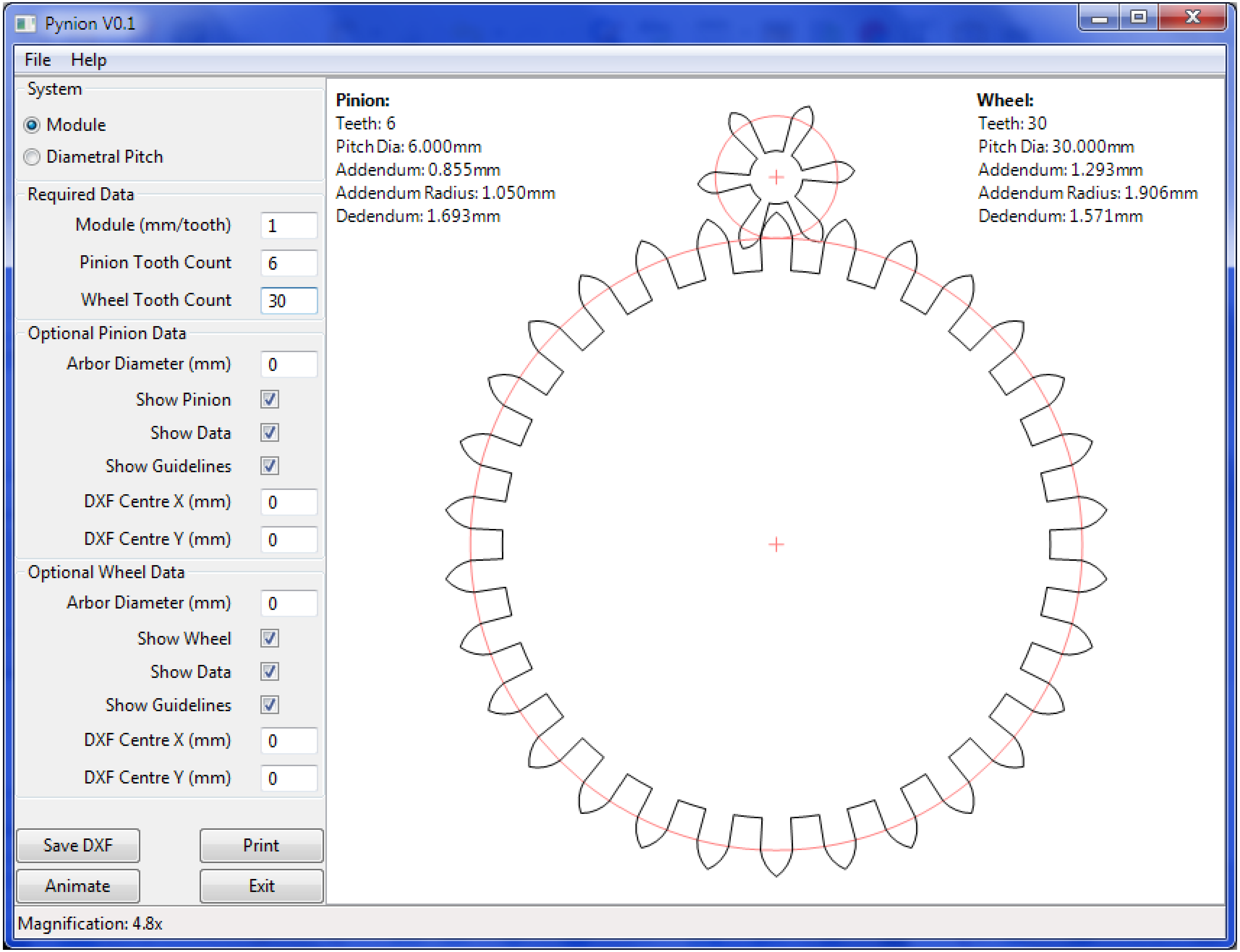
- wxPython comes with a comprehensive demo program
- Most widgets demoed with modifiable code and overview explanation

wxGlade GUI Builder

- Modelled after the Glade GTK+/GNOME GUI builder in Linux
- Handles the basic widgets directly
- Can use any widget with some additional input
- Written in Python so easy to expand if desired

Windows 7

Example Application (Windows)



System

● Module

○ Diametral Pitch

Required Data

Module (mm/tooth) 1

Pinion Tooth Count 6

Wheel Tooth Count 30

Optional Pinion Data

Arbor Diameter (mm) 0

Show Pinion ☑

Show Data ☑

Show Guidelines ☑

DXF Centre X (mm) 0

DXF Centre Y (mm) 0

Optional Wheel Data

Arbor Diameter (mm) 0

Show Wheel ☑

Show Data ☑

Show Guidelines ☑

DXF Centre X (mm) 0

DXF Centre Y (mm) 0

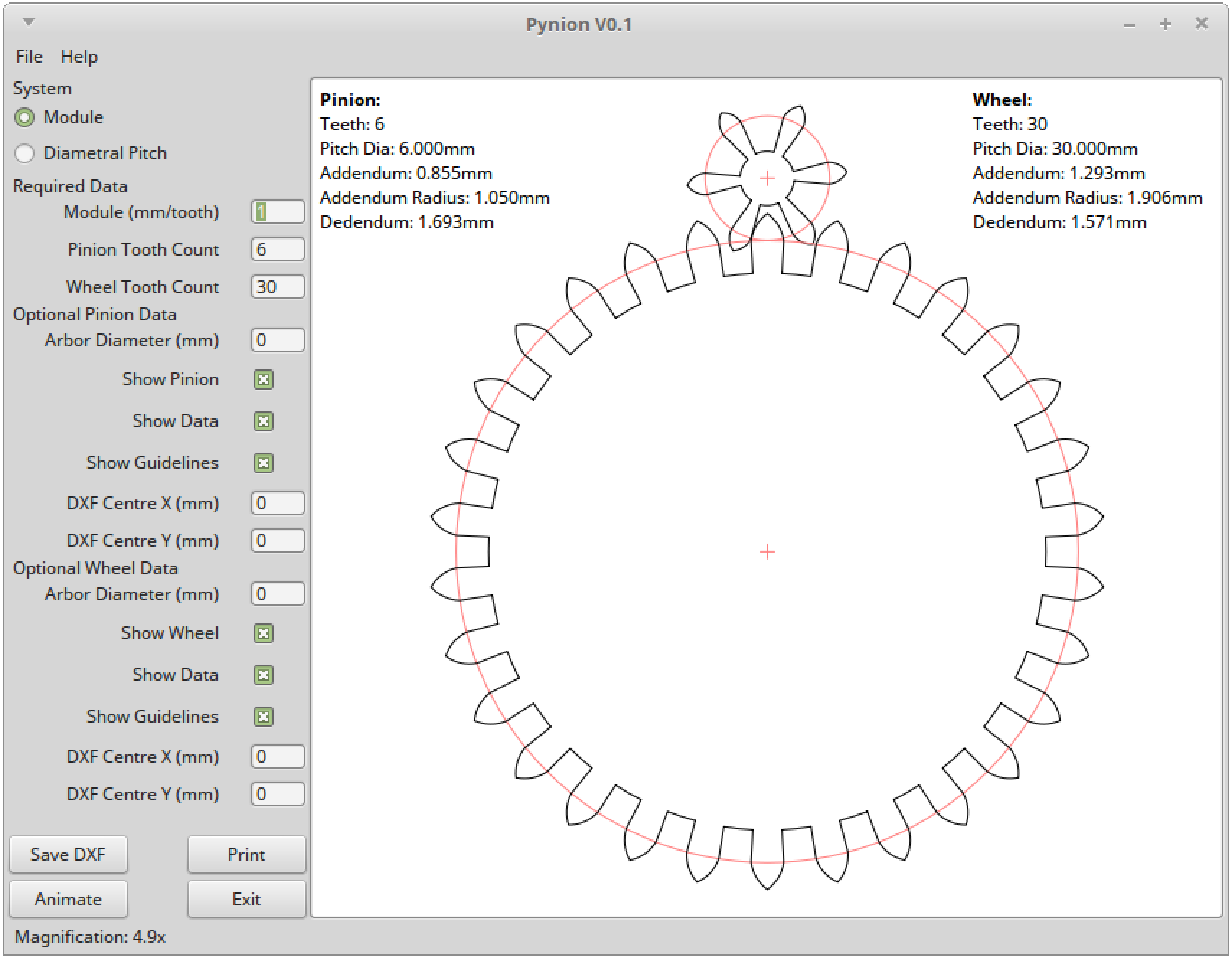
Pinion:
Teeth: 6
Pitch Dia: 6.000mm
Addendum: 0.855mm
Addendum Radius: 1.050mm
Dedendum: 1.693mm

Wheel:
Teeth: 30
Pitch Dia: 30.000mm
Addendum: 1.293mm
Addendum Radius: 1.906mm
Dedendum: 1.571mm

Magnification: 4.8x

Mint X

Example Application (Linux)



Pynion V0.1

System

● Module

○ Diametral Pitch

Required Data

Module (mm/tooth) 1

Pinion Tooth Count 6

Wheel Tooth Count 30

Optional Pinion Data

Arbor Diameter (mm) 0

Show Pinion ☒

Show Data ☒

Show Guidelines ☒

DXF Centre X (mm) 0

DXF Centre Y (mm) 0

Optional Wheel Data

Arbor Diameter (mm) 0

Show Wheel ☒

Show Data ☒

Show Guidelines ☒

DXF Centre X (mm) 0

DXF Centre Y (mm) 0

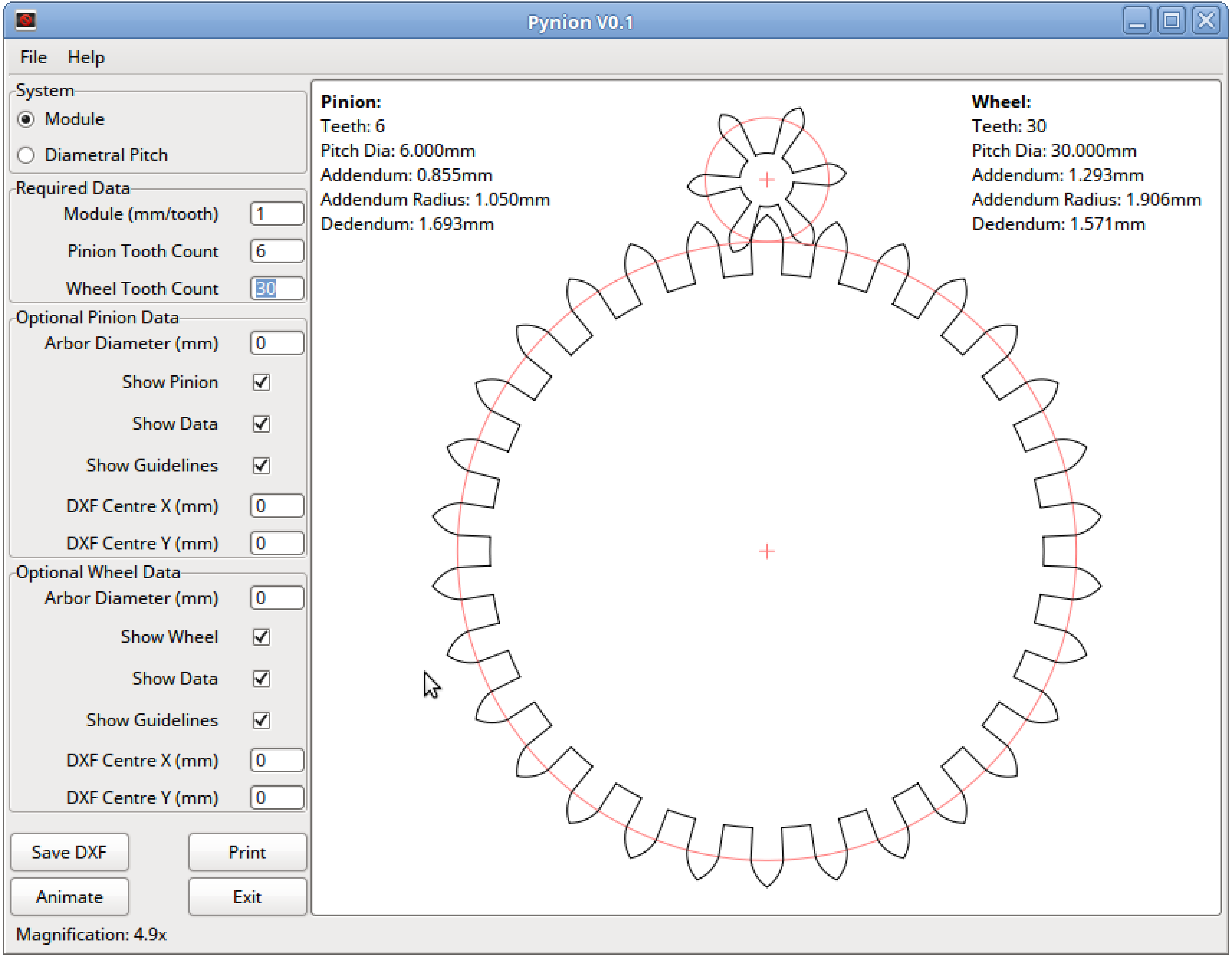
Magnification: 4.9x

Pinion:
Teeth: 6
Pitch Dia: 6.000mm
Addendum: 0.855mm
Addendum Radius: 1.050mm
Dedendum: 1.693mm

Wheel:
Teeth: 30
Pitch Dia: 30.000mm
Addendum: 1.293mm
Addendum Radius: 1.906mm
Dedendum: 1.571mm

Mint Clearlooks

Example Application (Linux)



System

● Module

○ Diametral Pitch

Required Data

Module (mm/tooth) 1

Pinion Tooth Count 6

Wheel Tooth Count 30

Optional Pinion Data

Arbor Diameter (mm) 0

Show Pinion ☑

Show Data ☑

Show Guidelines ☑

DXF Centre X (mm) 0

DXF Centre Y (mm) 0

Optional Wheel Data

Arbor Diameter (mm) 0

Show Wheel ☑

Show Data ☑

Show Guidelines ☑

DXF Centre X (mm) 0

DXF Centre Y (mm) 0

Magnification: 4.9x

Pinion:
Teeth: 6
Pitch Dia: 6.000mm
Addendum: 0.855mm
Addendum Radius: 1.050mm
Dedendum: 1.693mm

Wheel:
Teeth: 30
Pitch Dia: 30.000mm
Addendum: 1.293mm
Addendum Radius: 1.906mm
Dedendum: 1.571mm

Support Material

- Books
  - wxPython In Action
  - wxPython 2.8 Application Development Cookbook

- On-line
  - www.wxpython.org/docs/api/wx-module.html
  - www.wxpython.org/Phoenix/docs/html/main.html
  - groups.google.com/forum/#!forum/wxpython-users
  - wiki.wxpython.org
  - www.blog.pythonlibrary.org/tag/wxpython/

- Designer Tools
  - WxGlade - wxglade.sourceforge.net
  - wxFormBuilder - sourceforge.net/projects/wxformbuilder