Ereignisse, Beobachter und Handler

Beginnen wir bei einem einfachen Button auf einer Oberfläche. Wenn Sie ihn anklicken, erwarten Sie, dass etwas passiert – vielleicht im Hintergrund, möglichst aber so, dass Sie erkennen können, dass etwas passiert ist.

Beenden-Button

Die einfachste Anwendung wäre ein Button auf einer sonst leeren Oberfläche, der zum Beenden der Anwendung dient. Das hier gezeigte Fenster sieht zwar grässlich aus, immerhin ist der Button aber schon so beschriftet, wie wir es wollen.

Der Programmtext zu dieser einfachen Anwendung benötigt natürlich ein **`application`**-Objekt, das eine Instanz einer von wxFrame abgeleiteten Klasse erzeugt, anzeigt und dann in die MainLoop wechselt.

In der MainLoop wartet die Application auf ein Ereignis.

Ohne Ereignis, irgendeine Aktion, passiert also nichts.

Der bisherige Programmtext bietet dafür (fast) nichts, was zum Button gehört.

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

        wx.Frame.__init__(self, parent, ID, title, pos, size, style)
        panel = wx.Panel(self, -1)

        button = wx.Button(panel, 1003, "Beenden")
```

Allerdings gibt es bereits Ereignisse, die das Fenster betreffen. Diese werden standard-mäßig von wxFrame mit geliefert, die drei Symbole rechts oben in der Fensterleiste haben nämlich bereits ihre Funktion. Man kann das Fenster also minimieren, maximieren, auf Normalmaß bringen und schließen. Sie können sogar auch mit der Maus die Größe ändern.
Auch der Button hat minimale Funktionalität: Er ändert seine Darstellung, wenn die Maus über ihn gezogen wird und wenn man ihn anklickt.
Er macht aber - - - nichts!

Funktionalität definieren

Wie soll er das auch. Ohne Vorgaben durch den Programmierer soll der Button selbst-verständlich keine Aktion ausführen. Wir schreiben daher jetzt eine Methode, in der beschrieben wird, was zum Beenden geschehen soll. Sie ist sehr einfach:

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

Die Methode ist durch den notwendigen zweiten Parameter für ein Event-Objekt deutlich als Methode zur Ereignisbehandlung gekennzeichnet.
Baut man diese Methode ein und testet die Anwendung passiert nichts Neues, also nichts.

Bind

Der Grund ist, dass wir dem wxFrame-Objekt erst noch mitteilen müssen, dass bei laufender Anwendung die Methode *`OnCloseMe()`* darauf wartet, vom Button-Objekt angesprochen zu werden. Dazu dient die Bind-Methode, deren Aufruf mit in den Konstruktor aufgenommen werden muss. Die Zeile lautet:

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

Die Bind – Methode verknüpft also das Ereignis mit der Ereignisbehandlungsmethode, deren Name als zweiter Parameter anzugeben ist. Da die Quelle des EVT_BUTTON-Ereignisse nicht klar ist, muss sie als dritter Parameter angegeben werden, weil nur dann das Ereignis für genau den Button mit genau der Methode verknüpft wird.