Einführung in Python — Übung 3

Rebecca Breu

Februar 2011

wxPython

Aufgabe 1 (Hello World) Programmieren Sie die Hello-World-Anwendung aus der Vorlesung, welche ein leeres Frame erzeugt. Probieren Sie verschiedene Parameter für Titel, Größe und Position des Frames.

Aufgabe 2 (Statischer Text)

2.1 Erweitern Sie das Programm aus Aufgabe 1, sodass es Text im Frame darstellt (`wx.StaticText`). Stellen Sie mehrzeiligen Text zentriert, linsbündig, rechtsbündig dar.

2.2 Fügen Sie mehrere `wx.StaticText`-Widgets in das Frame ein. Was passiert, wenn Sie keine Positionsangaben machen? Ordnen Sie die Widgets mit Positionsangaben an. Beobachten Sie das Verhalten, wenn das Frame in seiner Größe verändert wird.

Aufgabe 3 (Buttons)

3.1 Erweitern Sie das Programm aus Aufgabe 1, sodass es einen Button darstellt. Wird auf diesen Button geclickt, soll sich das Programm beenden. (Dazu das MainFrame schließen: `self.Close(True)`)

3.2 Fügen Sie weitere Buttons hinzu. Die Buttons sollen verschiedene Aktionen ausfüren (z.B. verschiedenen Text im Terminal ausgeben), solabld sie geclickt werden.

Aufgabe 4 (Sizer) Erstellen Sie ein Frame mit einem BoxSizer und Buttons und probieren Sie verschiedene Möglichkeiten des BoxSizers aus. Z.B.:

- Ein Button, welcher stets so groß wie das gesamte Frame ist
- Mehrere Buttons nebeneinander
- Mehrere Buttons untereinander

Sie brauchen die Buttons nicht mit Funktionen belegen, hier geht es allein um das Layout.

Aufgabe 5 (RadioBox) Erstellen Sie ein Frame, welches eine RadioBox darstellt. Sobald der Anwender ein Item auswählt, soll die Auswahl in der Konsole ausgegeben werden.

Hinweis: Im Gegensatz zu den bisherigen Aufgaben ist es nun notwendig, das RadioBox-Objekt zu einem Objekt-Attriut des MainFrames zu machen, damit sie auch außerhalb der init-Methode des Frames darauf zugreifen können: `self.radiobox = ...`

Aufgabe 6 (Text-Editor) Schreiben Sie einen einfachen Text-Editor. Gehen Sie dabei wie folgt vor:

6.1 (Texteingabe-Feld) Erstellen Sie ein Frame mit einem mehrzeiligen TextCtrl-Widget. Das Textfeld sollte das ganze Frame ausfüllen, vgl. Aufgabe 4. Probieren Sie die Nutzung des Widgets aus!

- Was passiert, wenn Sie sehr viele Zeilen in das Widget einfügen, oder sehr lange Zeilen?
- Was bewirken Tastenkombinatioen wie Ctrl-C, Ctrl-V, Ctrl-X, Einfg, Entf, ...?

6.2 (Menüs) Fügen Sie eine Menüzeile hinzu und ein Menü „File“. Das File-Menü soll den Eintrag „Quit“ enthalten.

6.3 (Statuszeile) Fügen Sie eine Statuszeile hinzu. Beobachten Sie, wir dort die Hilfetexte der Menüeinträge angezeigt werden.

6.4 (Neue Datei) Fügen Sie dem File-Menü einen Eintrag „New“ hinzu, welcher den Inhalt des Texteingabe-Feldes löscht.

Hinweis: Analog zu Aufgabe 5 ist es nun notwendig, das Textfeld zu einem Objektattribut des MainFrame-Objekts zu machen.

6.5 (Datei öffnen) Fügen Sie dem File-Menü einen Eintrag „Open...“ hinzu. Über einen FileDialog soll der Anwender eine Datei auswählen können, welche im Textfeld angezeigt wird.

6.6 (Datei speichern als) Fügen Sie dem File-Menü einen Eintrag „Save as...“ hinzu. Über einen FileDialog soll der Anwender eine Datei auswählen können, in welcher der Inhalt des Textfelds gespeichert wird.

6.7 (Datei speichern) Fügen Sie dem File-Menü einen Eintrag „Save“ hinzu, der die Datei unter dem Namen speichert, unter dem sie geöffnet bzw. zuletzt gespeichert wurde. Geben Sie dafür dem MainFrame z.B. ein Objekt-Attribut `filename`, welches bei den Aktionen „Save as...“ und „Open ...“ entsprechend gesetzt wird.

Achtung, es gibt nicht immer einen Dateinamen, wenn der Anwender speichern will! (Wann?) In einem solchen Fall sollte das `filename`-Attribut `None` sein; idealerweise wird dann bei „Save“ ebenfalls ein FileDialog angeboten.

Des weiteren kann man den Dateinamen in der Titelzeile des Frames anzeigen lassen (`SetTitle`).

6.8 (Anzeige in der Statuszeile) Lassen Sie die Aktionen, die das Programm ausführt, in der Statuszeile anzeigen, z.B. „File bla.txt saved.“

6.9 (Speicherstatus) Speichern Sie den Status, ob eine Datei seit dem Öffnen oder dem letzten Speichern geändert wurde, in einem Attribut des MainFrame-Objekets. Lassen Sie den Status in geeigneter Weise in der Titelleiste des Frames anzeigen. Das Event, welches das TextCtrl beim Ändern des Textes ausgelöst, ist `wx.EVT_TEXT`.

Zusätzlich könnte man Warnungen (z.B. MessageDialog) einfügen, wenn der Anwender bei einer ungespeicherten Datei „Open...“, „Quit“ oder „New“ ausführen möchte.

Zusätzliche Aufgaben

Aufgabe 7 (Refactoring) Oft bestehen Teile einer GUI aus vielen gleichen Widgets mit ähnlichen Funktionalitäten, was zu vielen Wiederholungen im Programmcode führt. Für Menüs beispielsweise müssen viele einzelne Menü-Items erzeugt und deren Events an Event-Handler gebunden werden. Um den Programmcode zu vereinfachen, kann man für solche sich wiederholende Aufgaben generische Funktionen oder Klassen schreiben.

Verstehen Sie folgenden Code und benutzen Sie ihn in Ihrem Text-Editor:

```
class OneMenu(wx.Menu):
    """
    One popup menu.
    """

    def __init__(self, parent, items):
        """
        items: List of menu items, where one item is a list:
                  [name, status bar text, method to bind, enable flag]
        """

        wx.Menu.__init__(self)

        for item in items:
            if item:
                menu_item = self.Append(-1, item[0], item[1])
                menu_item.Enable(item[3])
                parent.Bind(wx.EVT_MENU, item[2], menu_item)
            else:
                self.AppendSeparator()
```

War das nicht genug? Weitere Aufgaben:

http://www.pythonchallenge.com