Introduction to Python — Exercise 3

Rebecca Breu

Juni 2010

wxPython

Exercise 1 (Hello World) Implement the Hello World application given in the lecture, which creates an empty frame. Try different parameters for title, size and position of the frame.

Exercise 2 (Static text)

2.1 Extend the program from Exercise 1 so that it displays text inside the frame (`wx.StaticText`). Display multi line text centered, as well as flushed left and flushed right.

2.2 Add more `wx.StaticText` widgets to the frame. What happens if you omit position parameters? Position the widgets via parameters and observe the behaviour of the frame on resizing.

Exercise 3 (Buttons)

3.1 Extend the program from Exercise 1 so that it displays a button. The program should exit when the button is pressed. (Close the main frame: `self.Close(True)`.)

3.2 Add more buttons. The buttons should execute different actions (e.g. print different text to the terminal) when clicked.

Exercise 4 (Sizer) Create a frame with a box sizer and buttons and try the different possibilities of box sizers. E.g.:

- A button that fills the whole frame
- A horizontal row of buttons
- A vertical row of buttons

You don't need to bind actions to the buttons; this exercise is all about layout.

Exercise 5 (RadioBox) Create a frame which displays a radio box. As soon as the user selects one item, print the selection to the console.

Hint: Contrary to the exercises above, it is now necessary to add the radio box object as an attribute to the frame object so that you can access it outside of the init method: `self.radiobox = ...`

Exercise 6 (Text editor) Write a simple text editor. Proceed by the following steps:

6.1 (Text entry field) Create a frame with a multiline TextCtrl widget. The text field should fill the whole frame, see Exercise 4. Try using the widget!

- What happens if you enter many lines into the field, or very long lines?
- What do key combinations like Ctrl-C, Ctrl-V, Ctrl-X, Ins, Del, ... do?

6.2 (Menus) Add a menu bar and a "File" menu. The menu should contain the entry "Quit".

6.3 (Status bar) Add a status bar. Observe how the menu entry's help texts are automatically displayed in the status bar.

6.4 (New file) Add a "New" entry to the status bar which deletes the content of the text entry field.

Hint: Analogous to Exercise 5 you need to make the text entry field an attribute of the main frame object now.

6.5 (Open file) Add a "Open ..." entry to the file menu. The user should be able to select a frile from a FileDialog which is then displayed in the text entry field.

6.6 (Save as) Add a "Save as ..." entry to the menu. The user should be able to select a file to wich the content of the text field entry is then saved.

6.7 (Save) Add a "Save" entry to the menu bar, which saves the file under the name it was opend or saved last. In order to do that, add an attribute to the main frame `filename` which is updated accordingly on "Save as" and "Open".

Beware, the filename does not always exist when the user wants to save! (Why?) In this case, the `filename` attribute should be `None`; ideally the action "Save" invokes a file dialog in this case, as "Save as" does.

Additionally, you can display the filename in the frame's status bar (`SetTitle`).

6.8 (Display in the status bar) Display the program's actions in the status bar, e.g. "File bla.txt saved."

6.9 (Save status) Save the status whether the file has been changed since being saved last in an attribute of the main frame. Display this status appropriately in the frame's title bar. The event triggered by the TextCtrl on text changes is `wx.EVT_TEXT`.

Additionally you can add warnings (e.g. MessageDialog) when the user wants to execute "Open", "Quit" or "New" while having an unsaved file open.

Additional Exercises

Exercise 7 (Refactoring) Oftentimes, parts of a GUI consist of many widgets of the same type with similar functionalities, which leads to lots of repetition in the source code. For menus, for example, we have to create a great deal of menu items and have to bind their events to event handler. To simplify the source code on such occasions, we can write generic functions or classes.

Understand the following code and use it in your 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()
```

Not enough? More exercises:

http://www.pythonchallenge.com