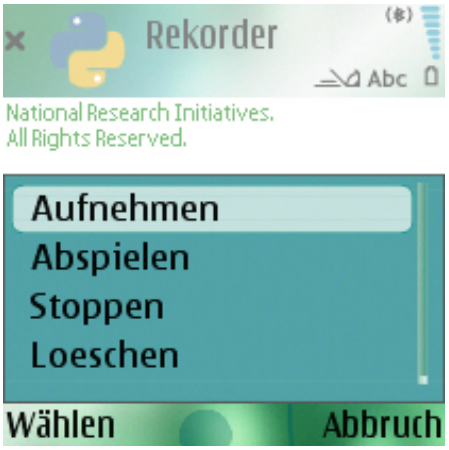
Matthias Heming 25. November 2009

Arbeitsblatt zum Thema »Listen, Tupel und GUI«

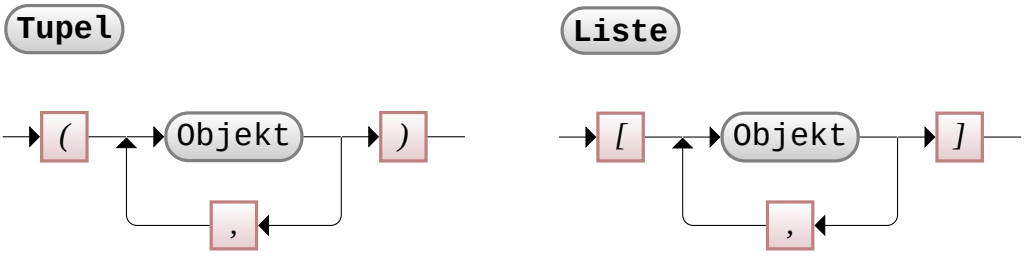
Wieder einmal wird das Beispiel des Aufnahmegerätes genutzt. Das Ergebnis der letzten aufwendigen Verbesserungsmaßnahmen war eine Aufnahmegeräte-Klasse, deren Methoden zumindest ein kleines bisschen intelligent erschienen. Diese Flexibilität ist dann wichtig, wenn beim Verfassen des Programms noch nicht klar ist, was genau mit den Objekten geschehen soll!

Auf der rechten Seite wird ein Bildschirmfoto gezeigt, auf dem die Steuerung des (bisher einzigen) Aufnahmegeräteobjektes durch ein Menü möglich ist. Ein **Menüeintrag** ist gekennzeichnet, durch eine **Beschriftung** und eine Anweisung, was getan werden soll, falls der Menüpunkt ausgewählt wird. Es wird also keine Anfrage, sondern ein **Auftrag** benötigt. Gehören zwei Dinge **(Beschriftung, Auftrag)** so eng zueinander, kann man sie als **Tupel** schreiben.



Nichts anderes, als beide Dinge in Klammern und mit Komma getrennt. So etwas kennen Sie bereits als Punkt in einem Koordinatensystem, ein Pärchen aus x-Wert und y-Wert wie $(3, 4)$, anstelle des Kommas wird hier häufig ein Schrägstrich verwendet.

Nun wurde geklärt, was ein Menüeintrag ist. Was ist nun ein **Menü**? Nichts anderes als eine **Liste** von Menüeinträgen. Passenderweise können mit Python sehr einfach Listen gemacht werden. Anstelle der runden Klammern bei Tupeln, nimmt man einfach eckige. Im folgenden Railroad-Diagramm wird deutlich, dass in einer Liste oder einem Tupel irgendwelche Objekte stehen (Es könnten Tupel in einer Liste stehen, Listen in einem Tupel stehen oder sogar Tupel in einem Tupel, das wiederum in einer Liste steht . . .).



Aufgabe 1

Formulieren Sie das auf dem Bildschirmfoto gezeigte Menü in der Syntax der Programmiersprache Python. Nehmen Sie dazu an, dass ein Objekt `a` der Klasse `Aufnahmegeraet` bereits erzeugt wurde und die einzelnen Aufträge daher mit `a.nimmAuf` etc. verfügbar sind.

Aufgabe 2

Passen Sie den Quelltext Ihres Diktiergerätprogramms so an, dass nicht mehr beim Programmstart festgelegt ist, was das Programm machen wird, sondern der Nutzer bzw. die Nutzerin das Programm über ein Menü steuern kann. Dazu benötigen Sie das im Modul `appuifw` enthaltene Objekt der Klasse `Application`, deren Details im beiliegenden Dokumentationsausschnitt erläutert sind. Probieren Sie neben dem Menü auch Modifikationen des Titels und der Bildschirmgröße aus.

Hinweis: Wird nur das Aussehen der GUI festgelegt, so wird das Programm nach dem Programmstart zwar passend aussehen, aber direkt beendet werden, da keine weiteren Aufträge durchgeführt werden sollen. Mit Hilfe der folgenden Zeilen, auch im QR-Code enthalten, wird dem Programm sinngemäß mitgeteilt: *Warte auf Eingaben der Nutzerin bzw. des Nutzers. Falls diese diese bzw. dieser Exit-Taste drücken, beende dich.*.

Fügen Sie diese Zeilen am Schluss Ihres Programmquelltextes hinzu.

```
import e32
ao_lock = e32.Ao_lock()
def exithandler():
    ao_lock.signal()
appuifw.app.exit_key_handler = exithandler
ao_lock.wait()
```

Aufgabe 3

Was bedeutet eigentlich die Abkürzung GUI? Was ist mit `appuifw` gemeint?

Zusatzaufgaben

- Überlegen Sie, wie ein Menü aussehen könnte, über das mehrere Objekte der Klasse Aufnahmegerät gesteuert werden können.
- Fügen Sie einen Menüeintrag hinzufügen, der das Programm beendet.

Application Type

A single implicit instance of this type always exists when *appuifw* module is present and can be referred to with the name *app*. New instances cannot be created by a Python program.
Instances of Application type have the following attributes:

body: The UI control that is visible in the application's main window. Currently either Text, a *Listbox* object, *Canvas*, or *None*.

exit_key_handler: A callable object that is called when the user presses the *Exit* softkey. Setting *exit_key_handler* to *None* sets it back to the default value.

menu This is a list of the following kinds of items:

- (title, callback) which creates a regular menu item
- (title, ((title, callback)[...])) which creates a submenu

title (Unicode) is the name of the item and *callback* the associated callable object. The maximum allowed number of items in a menu, or items in a submenu, or submenus in a menu is 30.

Example:

```
appuifw.app.menu = [(u"Item␣1", item1),
(u"Submenu␣1",
((u"Subitem␣1", subitem1),
(u"Subitem␣2", subitem2)))]
```

screen The screen area used by an application. See Figure 5.3 for example screens. The appearance of the application on the screen can be affected by setting one of the following values: 'normal', 'large', and 'full'.

Examples:

```
appuifw.app.screen='normal' # (a normal screen with
                            # title pane and softkeys)
appuifw.app.screen='large' # (only softkeys visible)
appuifw.app.screen='full' # (a full screen)
```