ncurses in Python

Dominik Pataky

15. Januar 2018

Python-Kurs 2017

Gliederung

1. Das Modul `curses`

2. Erstellen eines simplen Fensters
   - Grundlegende Funktionen
   - Darstellen von Symbolen
   - Allgemeine Hinweise

3. Arbeiten mit `curses.textpad`

Das Modul `curses`

Das Modul `curses`

`curses` bietet ein Interface für die „curses“-Bibliothek, den de-facto Standard für fortgeschrittenes Terminal Handling.

Das Modul `curses`

- curses ist vor allem in Unix-Umgebungen weit verbreitet
- dieses Modul wurde auf die API von *ncurses*, einer Open-Source Library für curses, zugeschnitten und funktioniert nur unter Linux und BSD Varianten von Unix

Mitgelieferte Submodule

`curses.ascii` zur Arbeit mit ASCII-kodierten Zeichen

`curses.panel` Support für Panels, also mehrere Fenster übereinander (Fenster mit Ebenen)

`curses.textpad` Widget zum editieren von Text mit *emacs*-artigen Key-Bindings

Erstellen eines simplen Fensters

Grundlegende Funktionen

Zum Erstellen eines `curses`-Fensters benutzt man den Befehl

```
1 curses.initscr()
```

- instanziiert die Bibliothek, gibt ein `WindowObject` zurück, welches den gesamten Screen repräsentiert
- Fehler beim Terminal öffnen können zum Beenden des Interpreters führen!

Grundlegende Funktionen

```
1 curses.endwin()
```

- deinitialisiert Bibliothek
- führt Terminal in normalen Zustand zurück

`curses` liefert einen eigenen Wrapper zum Starten und Beenden:

```
1 curses.wrapper(func, ...)
```

- setzt nützliche Standards, initialisiert curses und ruft *func* auf
- Wrapper fängt Exceptions, stellt Terminal wieder her und wirft Exception erneut
- *func* muss `stdscr` als erstes Argument akzeptieren (für das `WindowObject`)

Darstellen von Symbolen

Zur Darstellung einzelner Symbole verwendet man `addch`:

```
1 window.addch(ch[, attr])
2 window.addch(y, x, ch[, attr])
```

`ch` zu schreibendes ASCII-Zeichen (im ASCII-Format!)
Zum konvertieren ggf. auf `ord()` zurück greifen.

`y, x` Koordinaten, an die das Zeichen geschrieben wird.
Beachte invertierte Reihenfolge!

`attr` zu setzende Attribute

Darstellen von Symbolen

Zum Darstellen eines Strings existiert die Funktion `addstr()`:

```
1 window.addstr(str[, attr])
2 window.addstr(y, x, str[, attr])
```

- fügt ab `(y, x)` den gegebenen String ein
- überschreibt alle vorher vorhandenen Zeichen

Zeichen entfernen

```
window.delch([y, x])
window.deleteln()
window.erase()
```

- Funktionen zum löschen einzelner Zeichen, einer ganzen Zeile oder dem gesamten Fenster

Allgemeine Hinweise

- Werden bei Funktionen `x` und `y` weg gelassen und sind optional, wird die Funktion an der aktuellen Cursorstelle ausgeführt.
- Beachte die invertierte Zählweise!
  - Punkt (0, 0) liegt oben links in der Ecke vom Terminal
  - x zählt von da nach rechts aufwärts
  - y zählt von da nach unten aufwärts

Arbeiten mit `curses.textpad`

rectangle

```
1 curses.textpad.rectangle(win, uly, ulx, lry, lrx)
```

- erzeugt im WindowObject **win** ein Rechteck
- Anfangspunkt ist (**ulx**, **uly**) (ul ≙ upper left corner)
- Endpunkt ist (**lrx**, **lry**) (lr ≙ lower right corner)

Textbox

```
curses.textpad.Textbox(win)

Textbox().edit([validator])
Textbox().do_command(ch)
Textbox().gather()
```

- Eine `Textbox` wird mit einem WindowObject erstellt und hat diese drei Funktionen
- `edit()` „Editormodus“ wobei `validator` eine Funktion ist
- `do_command()` führt einen Befehl aus (Link hier)
- `gather()` gibt den Inhalt der Textbox zurück