Reguläre Ausdrücke


```
import re
```

```
m = re.search('abc', '123abc45')
if m:
    print m.start(), m.end()
```

```
for w in ('Mai', 'Juni', 'Juli', 'August'):
    m = re.search('Ju.i', w)
    if m:
        print w
```

```
r = re.compile('Ju.i')
for w in ('Mai', 'Juni', 'Juli', 'August'):
    m = r.search(w)
    if m:
        print w
```


Reguläre Ausdrücke

Reservierte Zeichen

`.^$*+?{[\|()`

Zeichenauswahl

`[abc]`	a oder b oder c
`[^abc]`	beliebiges Zeichen außer a, b, c
`[^\dA-F]`	alles außer Hexadezimalziffern

spezielle Zeichen

`.`	beliebiges Zeichen.
`^`	Anfang
`$`	Ende
`\d`	Dezimalziffer [0-9]
`\D`	keine Dezimalziffer [0-9]
`\s`	Whitespace [\t\n\r\f\v]
`\S`	kein Whitespace [^ \t\n\r\f\v]
`\w`	alphanumerisch [a-zA-Z0-9_]
`\W`	nicht alphanumerisch [^a-zA-Z0-9_]

Wiederholungen
des davor stehenden Zeichens

`*`	0 oder mehr mal
`+`	1 oder mehr mal
`?`	0 oder 1 mal
`{n}`	genau n mal
`{n,m}`	n bis n mal
`{n,}`	mindestens n mal
`{,m}`	höchstens m mal

"ungierige" Varianten

`*?`	0 oder mehr mal
`+?`	1 oder mehr mal
`??`	0 oder 1 mal
`{n,m}?`	

Alternativen

`xxx|yyy` xxx oder yyy

Gruppen
werden mit () geklammert

Methoden von regulären Ausdrücken

Diese Methoden können alternativ, als Funktionen des Moduls **re** aufgerufen werden. Dann wird ein zusätzliches erstes Argument (String) für das Suchmuster übergeben.

match(*string*[, *pos*[, *endpos*]])
sucht Übereinstimmung am Anfang des Strings
bzw. an Stelle pos. Gibt ein Match-Objekt zurück

search(*string*[, *pos*[, *endpos*]])
sucht erste Übereinstimmung ab Anfang des Strings
bzw. ab Stelle pos. Gibt ein Match-Objekt zurück

split(*string*[, *maxsplit* = 0])
zerlegt den String in Wörter, die durch Fundstellen getrennt werden

sub(*repl*, *string*[, *count* = 0])
ersetzt die Fundstellen durch repl.
(a) Wenn `repl` ein String ist, dient er zur Ersetzung
(b) Wenn `repl` eine Funktion ist, wird ihr jeweils das Match-Objekt übergeben.

subn(*repl*, *string*[, *count* = 0])
dto., gibt ein Tupel aus ersetztem String und Anzahl der Ersetzungen zurück

Methoden von Match-Objekten

group([*group1*, ...])
gefundener String

start([*group*])
Anfangsposition

end([*group*])
Endposition

span([*group*])
(Anfangsposition, Endposition) als Tupel

Nummerierte Gruppen

```
m = re.match(r'(\d+)\.(\d*)', '3.14')

m.group(0)     # '3.14'
m.group(1)     # '3'
m.group(2)     # '14'
```

Benannte Gruppen (?P<Name>...)

```
m = re.match(r'(?P<Tag>\d{1,2})\.\s*(?P<Monat>\w+)\s*(?P<Jahr>\d{4})',
             '26. Juli 2006')

m.group(1)
m.group('Tag')
```


Ersetzung von Gruppen

Funktion des Moduls `re`:

```
sub(pattern, repl, string[, count = 0])
```

Methode eines regulären Ausdrucks:

```
sub(repl, string[, count = 0])
```

ersetzt die Fundstellen durch repl.
(a) Wenn `repl` ein String ist, dient er zur Ersetzung
(b) Wenn `repl` eine Funktion ist, wird ihr jeweils das Match-Objekt übergeben.

Beispiel

```
def repl(m):
    return '%s-%s-%02d' % (m.group(3), m.group(2), int(m.group(1)))

re.sub(r'(?P<Tag>\d{1,2})\.\s*(?P<Monat>\w+)\s*(?P<Jahr>\d{4})',
       repl,
       'Wintersemester: 16. Oktober 2006 bis 9. Februar 2007')
```

Reguläre Ausdrücke: Übungsaufgabe

- Schreiben Sie ein Programm, das
  - Eine HTML-Datei einliest,
  - Alle Elementnamen in Großbuchstaben umwandelt,
  - Leerraum in Tags der Form <name /> entfernt,
  - Die geänderte Datei wieder abspeichert.

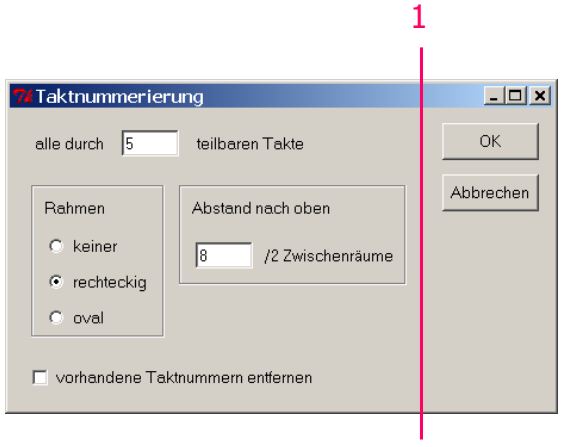
Beispiele:	
<?xml ...> <html> <body> 	<?xml ...> <HTML> <HR/> <BODY> <IMG src="img/c-sharp.jpg"/>

GUI-Programmierung mit Tkinter

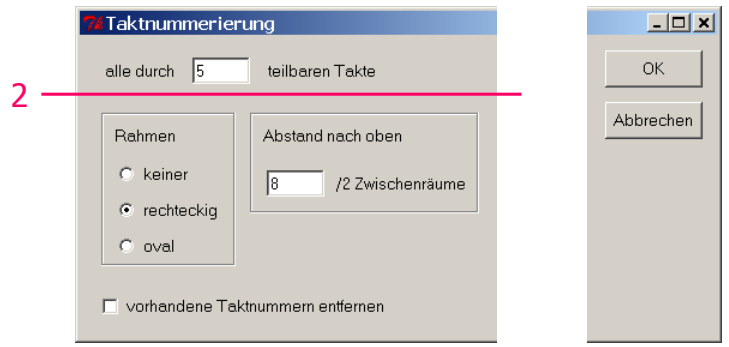
Erster Test


```
from Tkinter import *

frame = Tk()

frame.title("Test")

Button(frame, text="1. TOP").pack(side=TOP, fill=X)
Button(frame, text="2. LEFT").pack(side=LEFT, fill=Y)
Button(frame, text="3. TOP").pack(side=TOP, fill=X)
Button(frame, text="4. BOTTOM").pack(side=BOTTOM, fill=X)
Button(frame, text="5. RIGHT").pack(side=RIGHT, fill=Y)

frame.mainloop()
```


Ein Beispieldialog

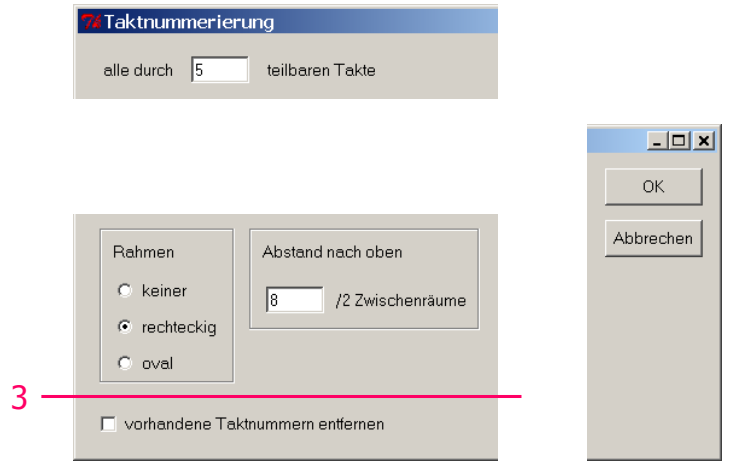


Zerlegung durch gerade Schnitte



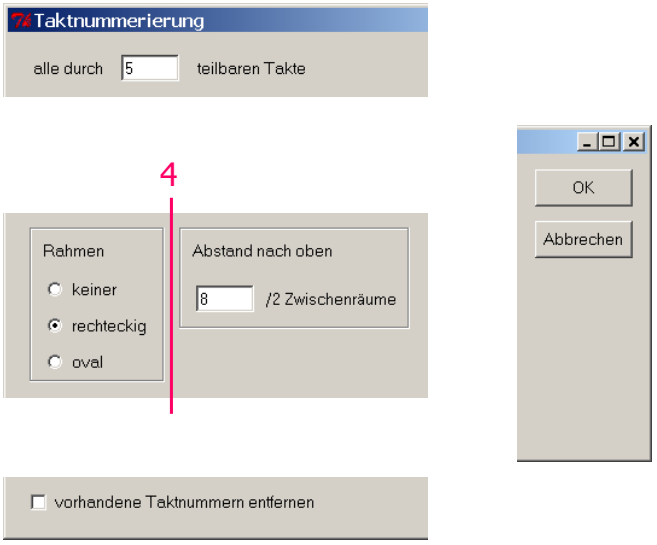
2

Zerlegung durch gerade Schnitte

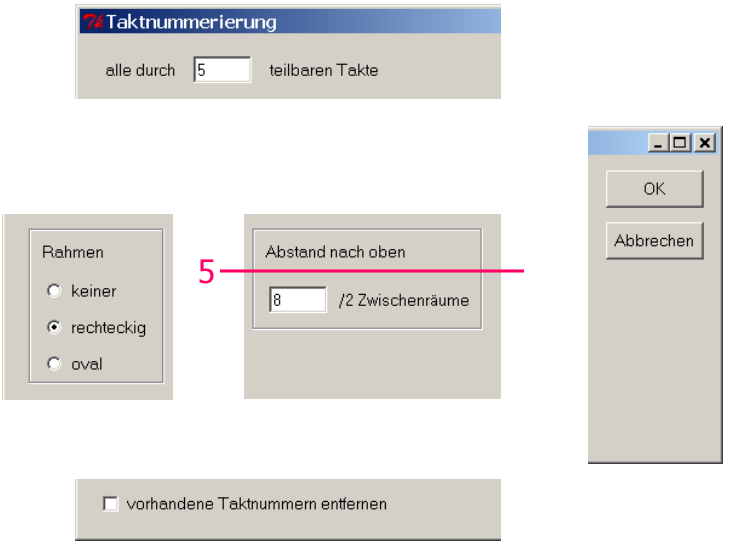


3

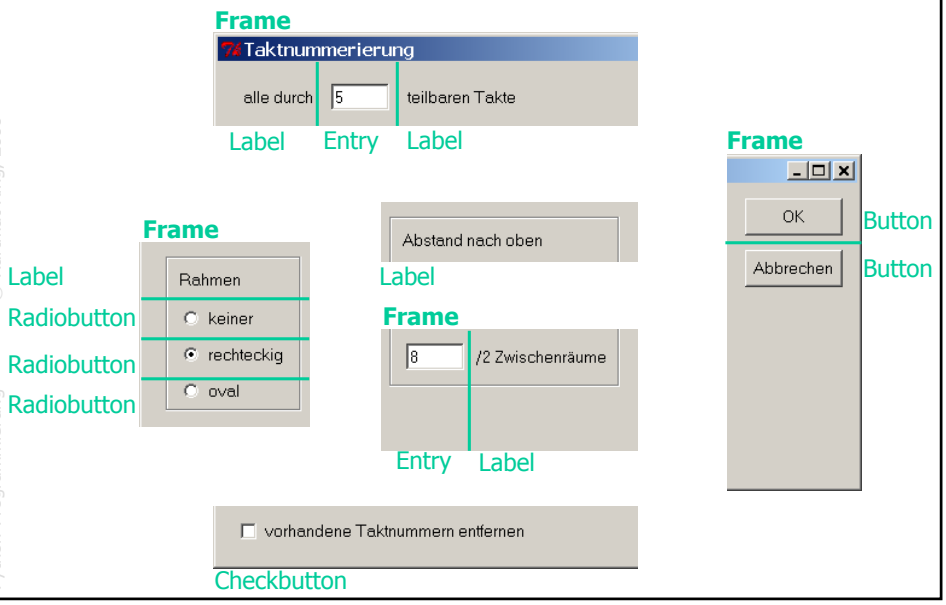
Zerlegung durch gerade Schnitte



Zerlegung durch gerade Schnitte

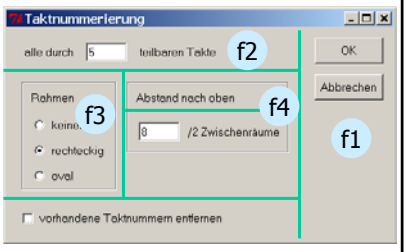


Zerlegung durch gerade Schnitte

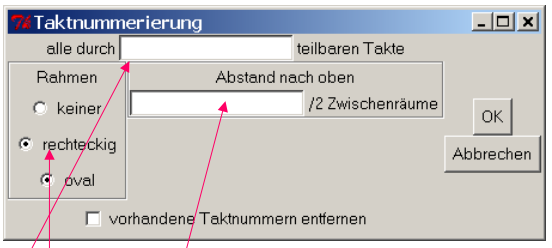


Zerlegung durch gerade Schnitte


```
from Tkinter import *

class TestDialog:
    def __init__(self):
        self.frame = Tk()
        self.frame.title("Taktnummerierung")
        self.createWidgets()
        self.frame.mainloop()

    def createWidgets(self):
        f1 = Frame(self.frame)
        f1.pack(side=RIGHT)
f1      Button(f1, text="OK").pack(side=TOP)
        Button(f1, text="Abbrechen").pack(side=TOP)

        f2 = Frame(self.frame)
        f2.pack (side=TOP)
f2      Label(f2, text="alle durch").pack(side=LEFT)
        Entry(f2).pack(side=LEFT)
        Label(f2, text="teilbaren Takte").pack(side=LEFT)

        Checkbutton (self.frame, text="vorhandene Nummern entfernen").pack(side=BOTTOM)

        f3 = Frame(self.frame, borderwidth=2, relief=GROOVE)
        f3.pack (side=LEFT)
        Label (f3, text="Rahmen").pack (side=TOP)
f3      Radiobutton(f3, text="keiner").pack(side=TOP)
        Radiobutton(f3, text="rechteckig").pack(side=TOP)
        Radiobutton(f3, text="oval").pack(side=TOP)

        f4 = Frame(self.frame, borderwidth=2, relief=GROOVE)
        f4.pack (side=TOP)
f4      Label(f4, text="Abstand nach oben").pack(side=TOP)
        Entry(f4).pack(side=LEFT)
        Label(f4, text="/2 Zwischenräume").pack(side=LEFT)

dlg = TestDialog()
```

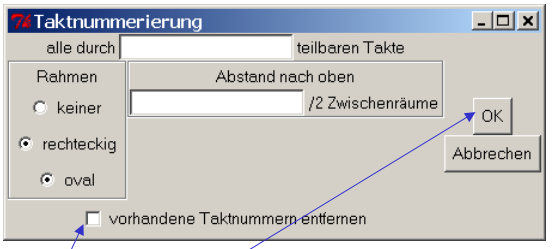


Mängel



- zu enges Layout
- alle Elemente zentriert
- Eingabefelder sind zu breit

Abhilfe




```
x.pack(side=TOP,
       fill=X,
       anchor=W,
       padx=8,
       pady=8)
```

LEFT, RIGHT, TOP, BOTTOM
X, Y
N, NE, E, SE, S, SW, W, NW
Abstand links und rechts
Abstand oben und unten

Button-Aktion

alt:

```
def createWidgets(self):
    ...
    Button (f1, text="OK").pack(side=TOP)
```

neu:

```
def onOK(self):
    self.ok = 1
    self.frame.destroy()

def createWidgets(self):
    self.ok = 0
    b = Button (f1, text="OK", command=self.onOK)
    b.pack(side=TOP, fill=X)
```


Entry abfragbar machen

alt:

```
Entry (f2).pack(side=LEFT)
```

neu:

```
self.text1 = StringVar()
self.text1.set("5")
self.entry1 = Entry(f2, textvariable=self.text1, width=6)
self.entry1.pack(side=LEFT)
```

Radiobutton anschließen

alt:

```
Radiobutton (f3, text="keiner").pack(side=TOP)
Radiobutton (f3, text="rechteckig").pack(side=TOP)
Radiobutton (f3, text="oval").pack(side=TOP)
```

neu:

```
self.radio = IntVar()

b = Radiobutton (f3, text="keiner", variable=self.radio, value=0)
b.pack(side=TOP, anchor=W)

b = Radiobutton (f3, text="rechteckig", variable=self.radio, value=1)
b.pack(side=TOP, anchor=W)

b = Radiobutton (f3, text="oval", variable=self.radio, value=2)
b.pack(side=TOP, anchor=W)

self.radio.set(1)
```


Python erweitern

test.cpp

```
// Beispiel für Python-Erweiterung

#include "Python.h"
#include <stdio.h>

static PyObject* square(PyObject* self, PyObject* args) {
  int n = 0;
  PyArg_ParseTuple(args, "i", &n);
  return Py_BuildValue("i", n*n);
}

static PyMethodDef test_functions[] = {
  {"square", square, METH_VARARGS, "berechnet das Quadrat"},
  {NULL, NULL}
};

extern "C"
void inittest() {
  Py_InitModule("square", test_functions);
}
```

Python

```
import test
print test.square(7)
```

test.dll

Python einbetten


```
// Beispiel für Python-Einbettung

#include <Python.h>

int main(int argc, char *argv[]) {
    Py_Initialize();
    PyRun_SimpleString("from time import time, ctime\n"
                       "print 'Today is', ctime(time())\n");
    Py_Finalize();
    return 0;
}
```


IronPython

- IronPython is the code name of the new implementation of the Python programming language running on .NET.
- It supports an interactive console with fully dynamic compilation. It is well integrated with the rest of the .NET Framework and **makes all .NET libraries easily available to Python programmers**, while maintaining full compatibility with the Python language.
- IronPython is currently in Beta status. We are planning on releasing V1.0 in summer 2006.


```
SciTe: python-properties:
#IronPython:
command.go.*.ipy=c:\Uni\Python\IronPython\IronPython\IronPythonConsole.exe "$(FileNameExt)"
command.go.subsystem.*.ipy=1
```