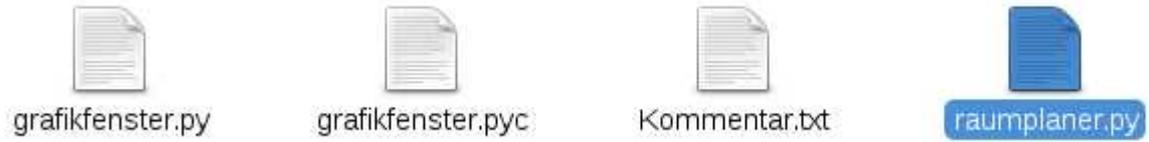


Einstieg in das Projekt Raumplaner (Einrichtungsplaner)

OO mit Python



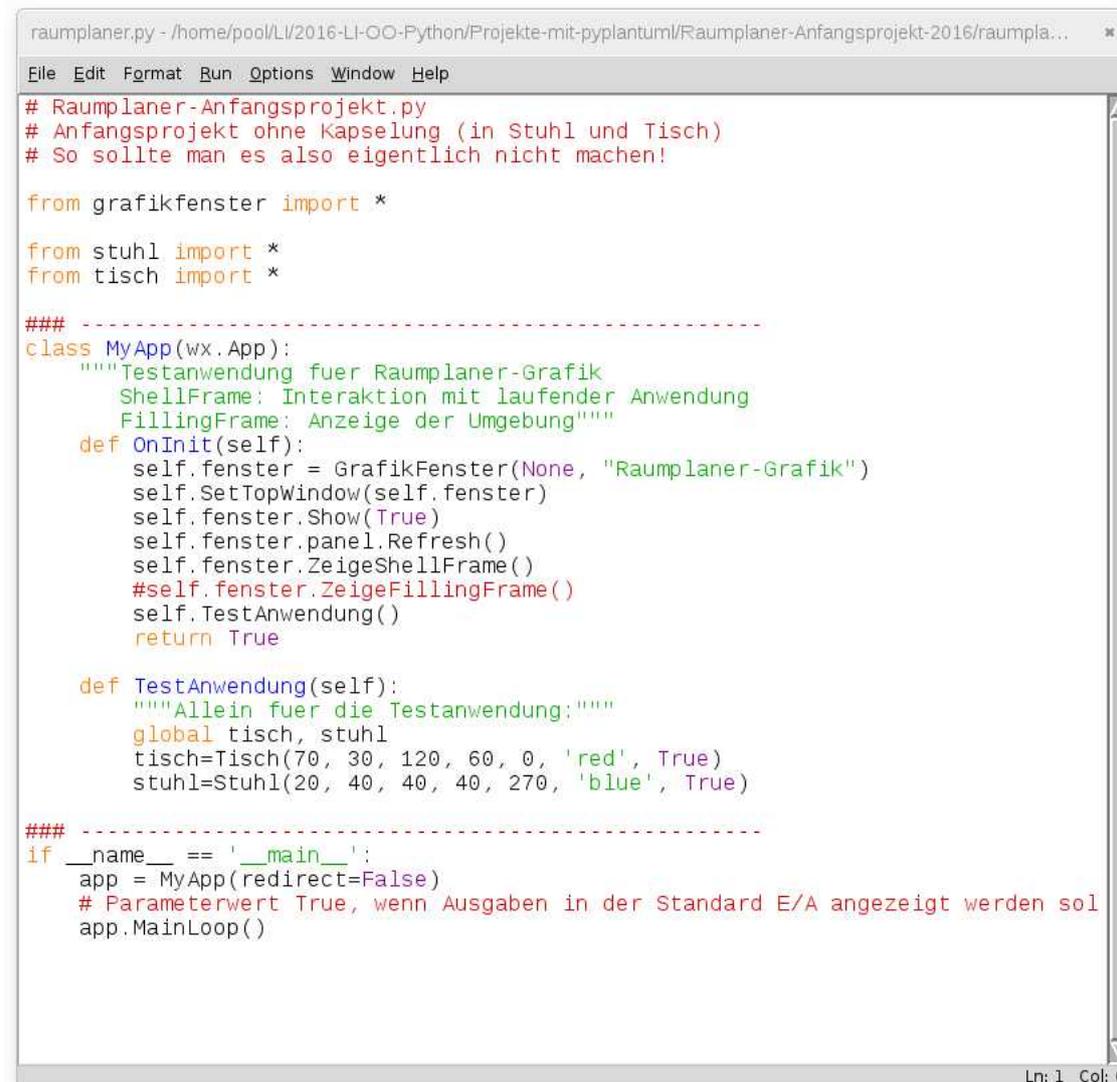
- Öffnen Sie das Projekt "Raumplaner-Anfangsprojekt"
- Wählen Sie "raumplaner.py" aus und öffnen Sie die Datei mit idle [oder einem anderen Python-Editor].

OO mit Python

- Starten Sie es mit "Run"
→ "Run Module"

oder

F5.



```
raumplaner.py - /home/pool/LI/2016-LI-OO-Python/Projekte-mit-pyplantuml/Raumplaner-Anfangsprojekt-2016/raumpla...
File Edit Format Run Options Window Help
# Raumplaner-Anfangsprojekt.py
# Anfangsprojekt ohne Kapselung (in Stuhl und Tisch)
# So sollte man es also eigentlich nicht machen!

from grafikfenster import *

from stuhl import *
from tisch import *

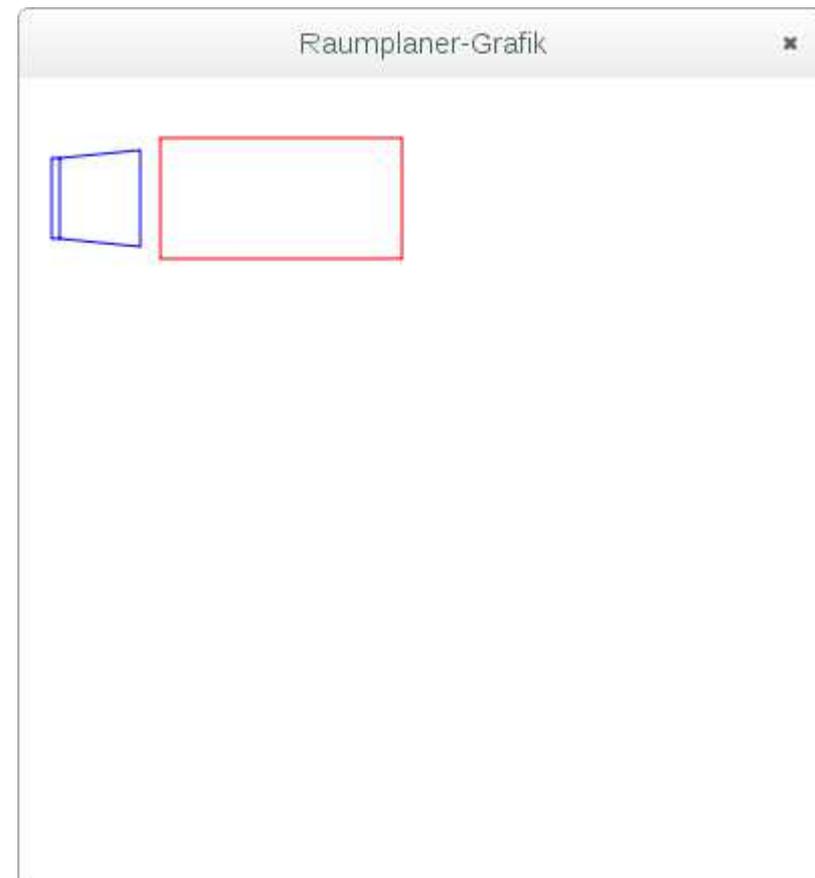
### -----
class MyApp(wx.App):
    """Testanwendung fuer Raumplaner-Grafik
    ShellFrame: Interaktion mit laufender Anwendung
    FillingFrame: Anzeige der Umgebung"""
    def OnInit(self):
        self.fenster = GrafikFenster(None, "Raumplaner-Grafik")
        self.SetTopWindow(self.fenster)
        self.fenster.Show(True)
        self.fenster.panel.Refresh()
        self.fenster.ZeigeShellFrame()
        #self.fenster.ZeigeFillingFrame()
        self.TestAnwendung()
        return True

    def TestAnwendung(self):
        """Allein fuer die Testanwendung:"""
        global tisch, stuhl
        tisch=Tisch(70, 30, 120, 60, 0, 'red', True)
        stuhl=Stuhl(20, 40, 40, 40, 270, 'blue', True)

### -----
if __name__ == '__main__':
    app = MyApp(redirect=False)
    # Parameterwert True, wenn Ausgaben in der Standard E/A angezeigt werden sol
    app.MainLoop()
```

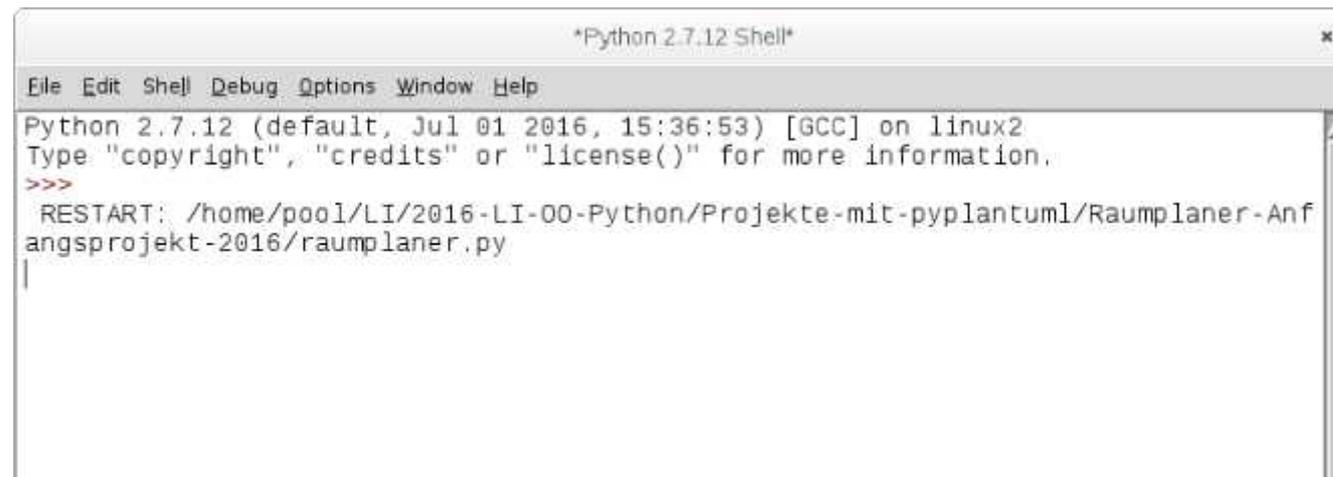
OO mit Python

- Eines der geöffneten Fenster zeigt eine symbolische Darstellung eines Stuhls und eines Tisches.



OO mit Python

- Ein weiteres zeigt die *Standardkonsole* von Python, ...

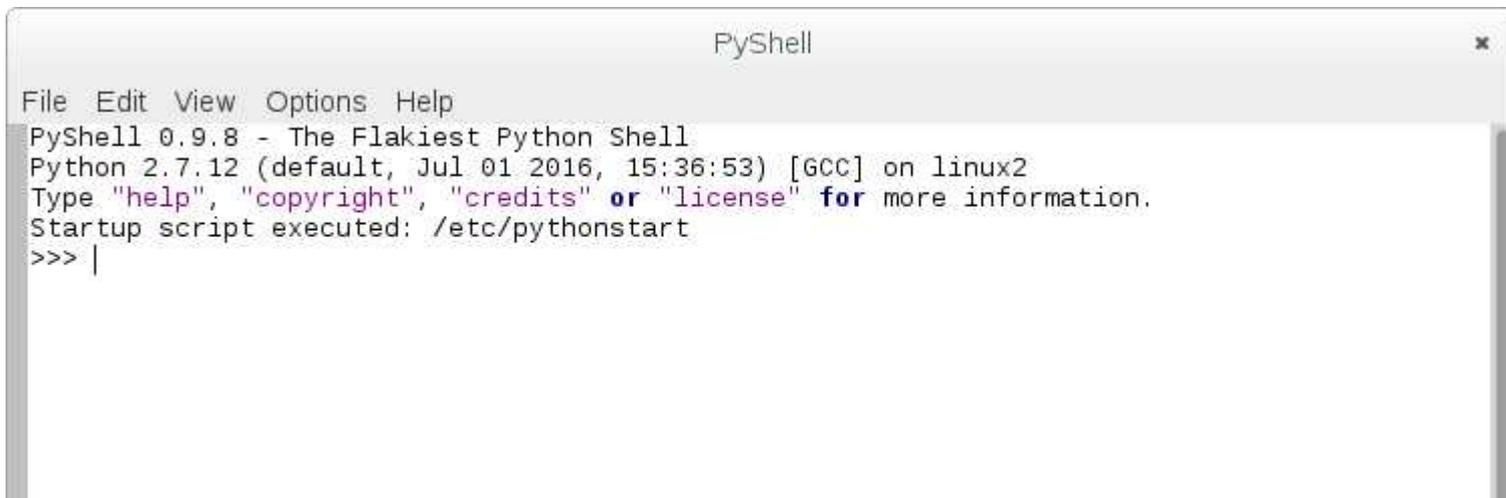


```
*Python 2.7.12 Shell*
File Edit Shell Debug Options Window Help
Python 2.7.12 (default, Jul 01 2016, 15:36:53) [GCC] on linux2
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: /home/pool/LI/2016-LI-00-Python/Projekte-mit-pyplantuml/Raumplaner-Anfangsprojekt-2016/raumplaner.py
|
```

- ... die allerdings während des laufenden Programms inaktiv ist.

OO mit Python

- Ein drittes zeigt (bei entsprechender Einstellung im Projekt) die aktive *PyShell*.

A screenshot of a PyShell terminal window. The window title is "PyShell" with a close button (x) in the top right corner. The menu bar contains "File", "Edit", "View", "Options", and "Help". The main text area displays the following information: "PyShell 0.9.8 - The Flakiest Python Shell", "Python 2.7.12 (default, Jul 01 2016, 15:36:53) [GCC] on linux2", "Type 'help', 'copyright', 'credits' or 'license' for more information.", and "Startup script executed: /etc/pythonstart". The prompt ">>>|" is visible at the end of the last line.

```
PyShell
File Edit View Options Help
PyShell 0.9.8 - The Flakiest Python Shell
Python 2.7.12 (default, Jul 01 2016, 15:36:53) [GCC] on linux2
Type "help", "copyright", "credits" or "license" for more information.
Startup script executed: /etc/pythonstart
>>> |
```

OO mit Python

- Die aktive *PyShell* wollen wir nun nutzen, um unsere beiden Objekte zu manipulieren.
- Um herauszufinden, was wir beim derzeitigen Stand des Projekts machen können, gibt es mehrere Möglichkeiten.

OO mit Python

- Eine Möglichkeit bietet eine Hilfe durch Klassenkarten, die uns die Attribute und Methoden zeigen, welche die Klassen Stuhl und Tisch bereitstellen.

tisch.py

C Tisch

x
y
b
t
f
w
s

```
• __init__(self, xPos=20, yPos=20, breite=40, tiefe=40, winkel=0, farbe='blue', sichtbar=False)
• BewegeHorizontal(self, weite)
• BewegeVertikal(self, weite)
• Drehe(self, winkel)
• GibFarbe(self)
• GibFigur(self)
• GibSichtbar(self)
• Verberge(self)
• Zeige(self)
```

stuhl.py

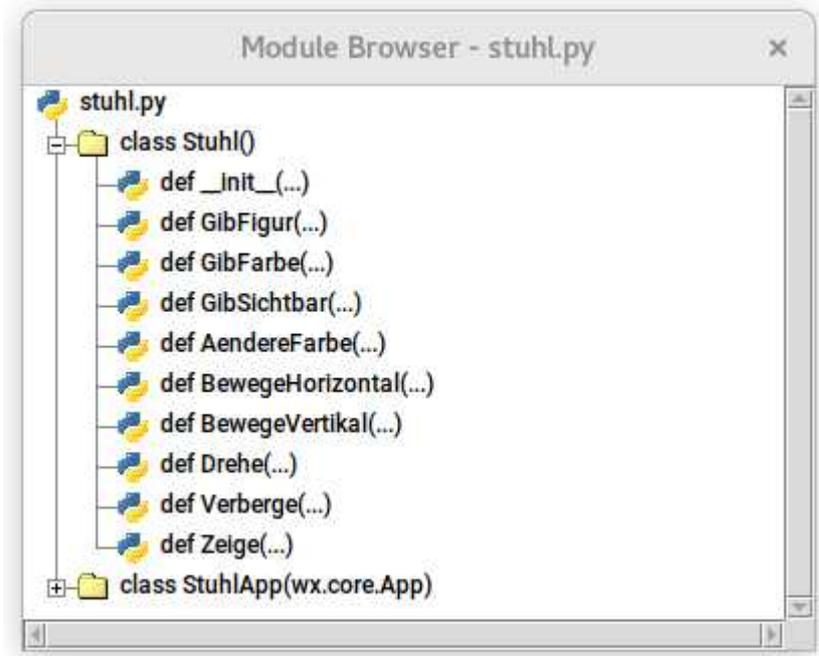
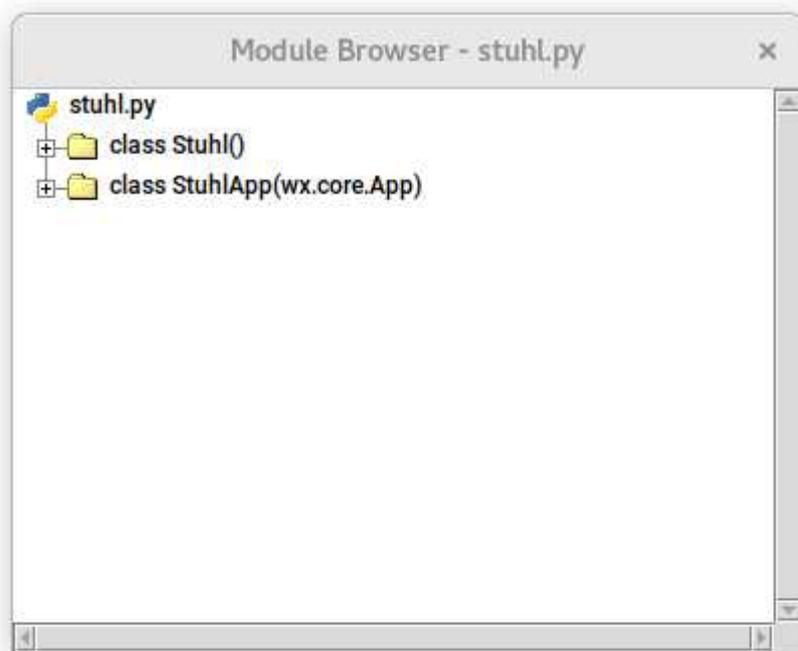
C Stuhl

x
y
b
t
f
w
s

```
• __init__(self, xPos=20, yPos=20, breite=40, tiefe=40, winkel=0, farbe='blue', sichtbar=False)
• BewegeHorizontal(self, weite)
• BewegeVertikal(self, weite)
• Drehe(self, winkel)
• GibFarbe(self)
• GibFigur(self)
• GibSichtbar(self)
• Verberge(self)
• Zeige(self)
```

OO mit Python

- <Strg> - B
beim aktiven Fenster eingeben, dann wird ein eigenes Fenster mit einem *Modulbrowser* angezeigt.
- Anklicken von + zeigt in den Unterpunkten die zur Verfügung stehenden Methoden



OO mit Python

- Anklicken einer Methode zeigt sie im Programmtext

```
def BewegeHorizontal(self, weite):  
    """Veraendernde Methode fuer die x-Position"""  
    self.Verberge()  
    self.x += weite  
    self.Zeige()  
  
def BewegeVertikal(self, weite):  
    """Veraendernde Methode fuer die y-Position"""  
    self.Verberge()  
    self.y += weite  
    self.Zeige()  
  
def Drehe(self, winkel):  
    """Veraendernde Methode fuer die Winkel"""  
    self.Verberge()  
    self.w += winkel  
    self.Zeige()  
  
def Verberge(self):  
    """Veraendernde Methode fuer die Sichtbarkeit"""  
    self.sichtbar = not self.sichtbar
```

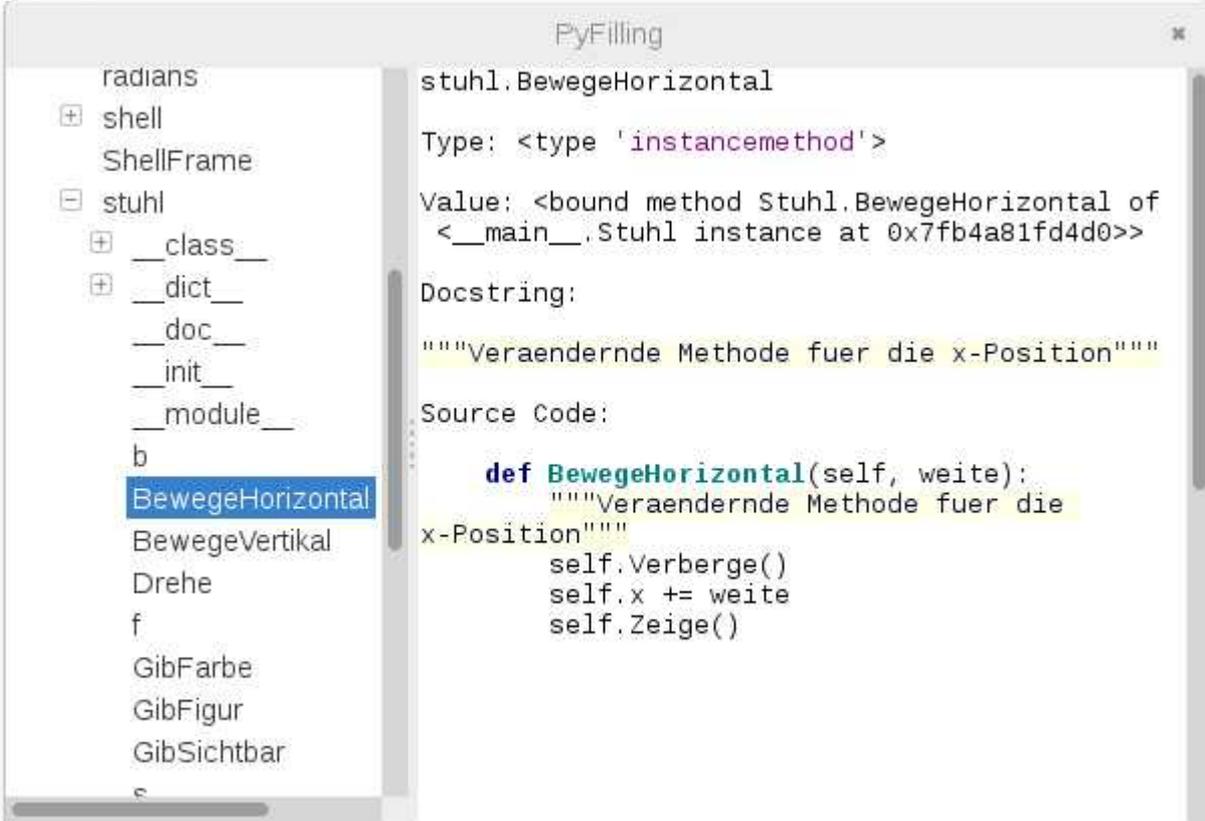


OO mit Python

Eine weitere einfache Möglichkeit:
Im Programmtext der App den Aufruf

```
self.fenster.ZeigeFillingFrame()
```

"einkommentieren".

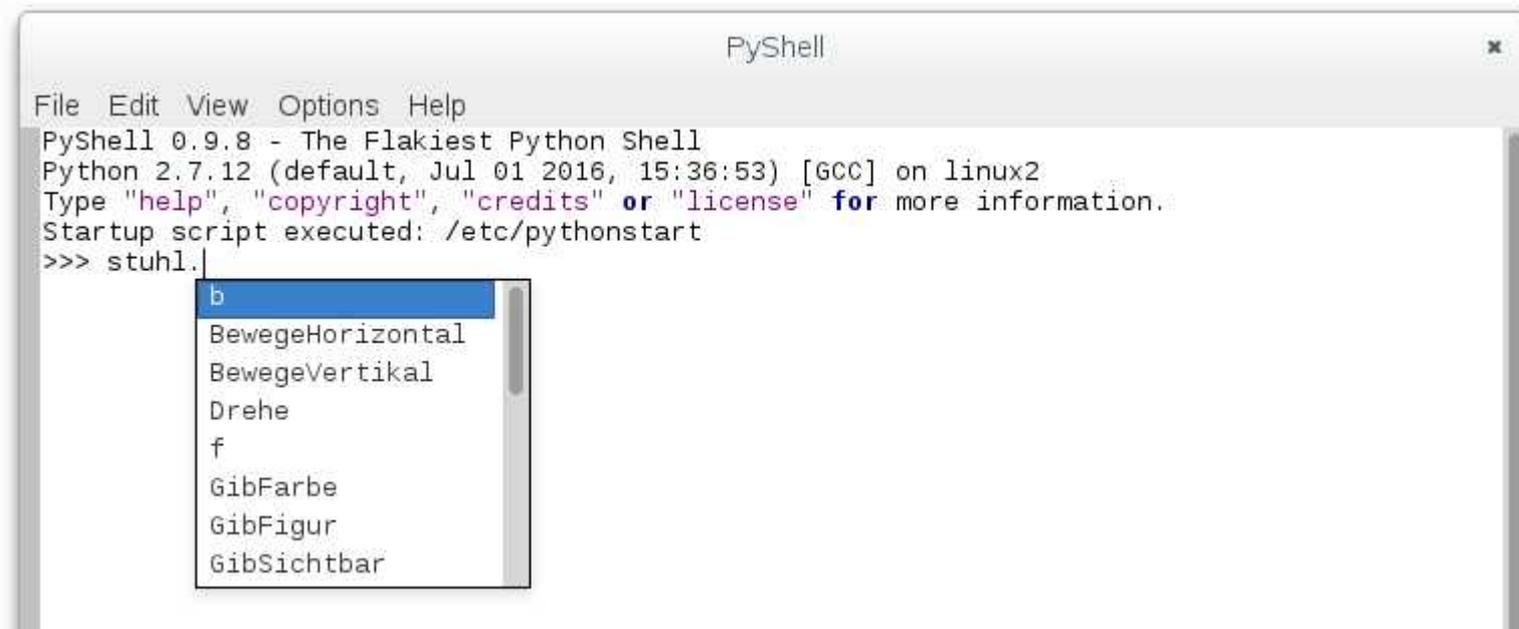


The screenshot shows the PyFilling namespace inspector. On the left, a tree view shows the namespace structure: 'radians', 'shell' (containing 'ShellFrame'), and 'stuhl' (containing 'BewegeHorizontal', 'BewegeVertikal', 'Drehe', 'f', 'GibFarbe', 'GibFigur', 'GibSichtbar', and 's'). The 'BewegeHorizontal' method is selected and highlighted in blue. On the right, the details for this method are shown: 'stuhl.BewegeHorizontal', Type: '<type 'instancemethod'>', Value: '<bound method Stuhl.BewegeHorizontal of <__main__.Stuhl instance at 0x7fb4a81fd4d0>>', and Docstring: '"""Veraendernde Methode fuer die x-Position"""'. The Source Code section shows the following code: 'def BewegeHorizontal(self, weite):', '"""Veraendernde Methode fuer die x-Position"""', 'self.Verberge()', 'self.x += weite', and 'self.Zeige()'. The window title is 'PyFilling' and the footer text is 'PyFilling - The Tastiest Namespace Inspector'.

OO mit Python

- Die einfachste Möglichkeit aber bietet die aktive *PyShell* selbst:

Bei entsprechender Einstellung im Projekt erhalten wir nach der Eingabe von *stuhl* ein Kontexthilfe.



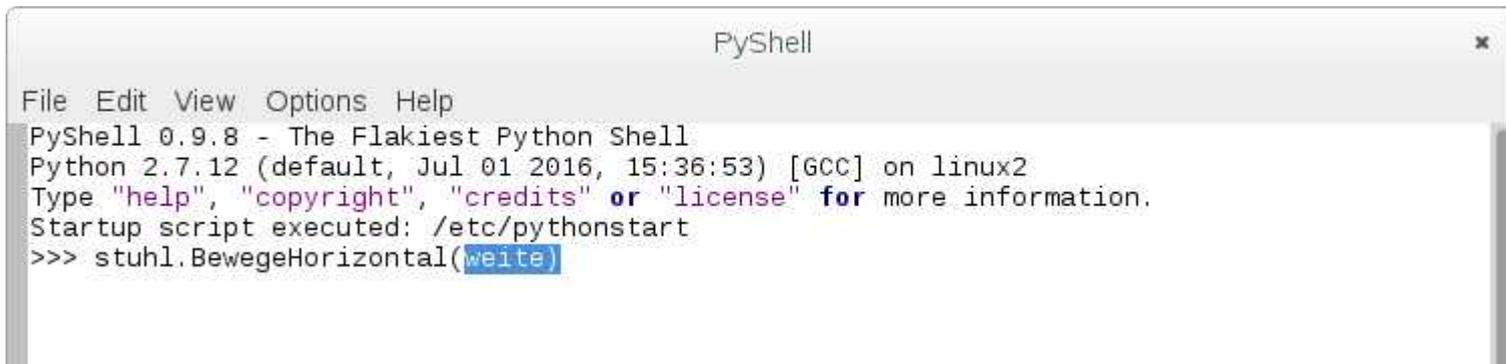
```
PyShell
File Edit View Options Help
PyShell 0.9.8 - The Flakiest Python Shell
Python 2.7.12 (default, Jul 01 2016, 15:36:53) [GCC] on linux2
Type "help", "copyright", "credits" or "license" for more information.
Startup script executed: /etc/pythonstart
>>> stuhl.
```

The screenshot shows a context help menu for the variable 'stuhl'. The menu items are:

- b
- BewegeHorizontal
- BewegeVertikal
- Drehe
- f
- GibFarbe
- GibFigur
- GibSichtbar

OO mit Python

- Wir wählen beispielsweise *BewegeHorizontal* aus und setzen die Eingabe mit einer öffnenden Klammer fort.
- Wir bekommen wieder eine Hilfe angeboten und geben eine Weite ein, beispielsweise 190.

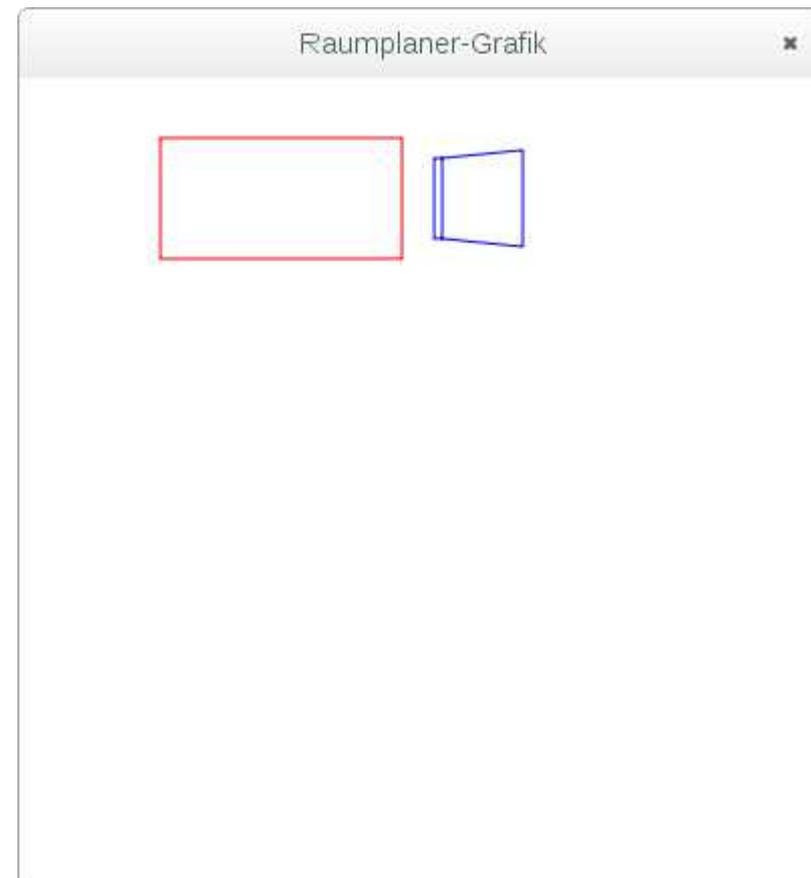


```
PyShell
File Edit View Options Help
PyShell 0.9.8 - The Flakiest Python Shell
Python 2.7.12 (default, Jul 01 2016, 15:36:53) [GCC] on linux2
Type "help", "copyright", "credits" or "license" for more information.
Startup script executed: /etc/pythonstart
>>> stuhl.BewegeHorizontal(weite)
```

- Klammer schließen und <ENTER>.

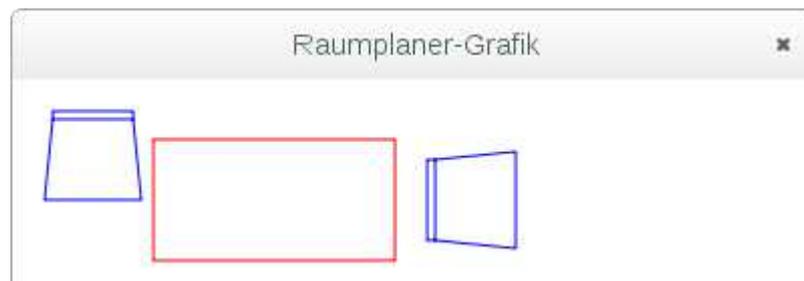
OO mit Python

- Das Stuhlsymbol hat wie erwartet seine Position geändert.



OO mit Python

- Probieren Sie weitere Möglichkeiten zur Manipulation aus.
- Sie können auch weitere Objekte erzeugen, indem Sie beispielsweise
`stuhl_2 = Stuhl()`
eingeben.
- Überlegen Sie, weshalb der neue Stuhl nicht zu sehen ist und wie Sie das ändern können.



OO mit Python

Eine schöne anschließende Aufgabe ist das Erstellen einer

Tischgruppe
aus einem Tisch
mit sechs Stühlen

