Dokumentation

Zum Einsatz von Dokumentation in Programmtext gibt es durchaus ernstzunehmende kritische Stimmen[1]. Nach meiner Erfahrung ist es in der Regel ein schwieriges Unterfangen, Schülerinnen und Schüler dazu zu bewegen, Kommentare in Programmtexte einzufügen. Andererseits sind in Python einige besondere Attribute[2] vordefiniert, die auch für Dokumentationszwecke nützlich sind.

__doc__

Wir können einem Objekt – das kann ein ganzes Modul, eine Klasse, aber auch eine Methode sein – einen Dokumentationsstring zuweisen. Mit dem besonderen Attribut **__doc__** können wir darauf zugreifen.
Im hier beschriebenen Beispiel enthält die Datei mit dem Hauptprogramm, also die zu startende Anwendungsklasse, am Anfang den folgenden mehrzeiligen String:

```
"""Anwendungsklasse zum Gesamtprojekt
   In dieser Version
   - wird mit Vererbung gearbeitet
   - ist das Projekt auf Teilmodule aufgeteilt
   ShellFrame und FillingFrame koennen eingebunden werden.
   Die in der [nicht notwendigen] TestAnwendung definierten
   Moebel-Objekte sind Namen zugewiesen, um im ShellFrame
   einfach auf sie zugreifen zu koennen.
"""
```

Starten wir das Programm, können wir im ShellFrame mit **`print __doc__`** diesen Text ausgegeben bekommen. Gerade bei unterschiedlichen Versionen des selben Programms ist das eine hilfreiche Dokumentation.

Auch die Anwendungsklasse hat in diesem Beispiel einen Doc-String:

```
class MyApp(wx.App):
    """Anwendungsklasse, die
    - ein Grafikfenster fuer die Darstellung der Moebel-Objekte anzeigt
    - standardmaessig eine Testanwendung startet
    - die Moeglichkeit des Einbindens von ShellFrame und FillingFrame bietet
    """
```

Dieser kann ebenso angezeigt werden wie der Doc-String zur TestAnwendung:

```
    def TestAnwendung(self):
        "Allein fuer die Testanwendung:"
```

Was sollte im Projekt noch mit Kommentaren versehen werden?

Sicherlich ist es auch bei der vererbenden Klasse sinnvoll, einen Kommentar anzugeben. Bei den einzelnen Moebel-Klassen ist es weniger sinnvoll, da ihre Namen ausreichend zu ihrer Beschreibung sind. Auch der Name der Methode **`GibFigur()`** reicht sicherlich aus. Grundsätzlich sollte man die Namen von Attributen, Methoden und Klassen so wählen, dass ein zusätzlicher Kommentar unnötig ist. Auch die Namen der Projekte wähle ich möglichst so, dass die Besonderheiten ähnlicher Projekte deutlich werden.

---

1 Siehe dazu der Text aus Clean Code. Robert C. Martin weist sehr eindringlich darauf hin, dass es nur an wenigen Stellen nötig ist, Kommentare einzufügen. Insbesondere rügt er das standardmäßige Einfügen von Kommentaren [wie es z.B. BlueJ in seinen Standarddateien pflegt]. Besser ist es seiner Meinung nach, Programmtexte so zu schreiben, dass sie selbsterklärend sind.

2 Siehe dazu der gesonderte Text.

Weitere dokumentierende Attribute: ***__dict__*** ***__class__*** ***__bases__***

Alle Attribute und Methoden kann man von einem Objekt mit der eingebauten Funktion **`dir(...)`** abfragen. Sie listet dann die Schlüsselworte (keys) aus dem Dictionary aus, das zu dem Objekt gehört. **`__dict__`** kann direkt ausgelesen werden kann, allerdings listet **`__dict__`** nur die eigenen, aber nicht die geerbten Methoden aus. Die in älteren Versionen von Python vorhandenen besonderen Attribute __methods__ und __members__ gibt es nicht mehr, da dir(...) deren Werte liefert.
Das Attribut __class__ liefert zu einem Objekt die Klasse, der es angehört.

```
>>> app.tisch.__class__
<class 'tisch.Tisch'>
```

__bases__ liefert die Oberklassen einer Klasse.

```
>>> Tisch.__bases__
(<class 'moebel.Moebel'>,)
```