Probleme bei der Initialisierung

Probleme
bei der Klasse
MoebelGruppe

Probleme bei der Initialisierung

- Beachten Sie die Reihenfolge im Konstruktor der Klasse MoebelGruppe:

```
def __init__(self,
                 xPos=50,
                 yPos=150,
                 winkel=0,
                 farbe=""):
        self.__moebel=[]
        Moebel.__init__(self,xPos,yPos,0,0,
                        winkel, farbe, True)
```

- Anderenfalls gibt es eine Fehlermeldung beim Erzeugen einer Instanz von MoebelGruppe

Probleme bei der Initialisierung

- Der Grund liegt in der Reihenfolge der Initialisierung:
- Der Konstruktor der Oberklasse Moebel wird anderenfalls aufgerufen, bevor das Attribut angelegt wird.
- In der letzten Zeile des Konstruktors von Moebel wird die Methode `Update()` aufgerufen.
- Das ist aber die `Update()`-Methode von Moebelgruppe, in der auf `self.__moebel` zugegriffen wird.

Probleme bei der Initialisierung

- Das ist aber die Update()-Methode von Moebelgruppe, in der auf `self.__moebel` zugegriffen wird:

```
def Update(self):
    for moebel in self.__moebel:
        moebel.Update()
```

Probleme bei der Initialisierung

Probleme
bei der Klasse
RaumplanerApp

beim Umsetzen der
Persistenz

Probleme bei der Initialisierung

- Das entsprechende Problem ist bei meiner Bearbeitung der RaumplanerApp mit der Ergänzung zur Umsetzung des Speicherns und Ladens der erzeugten Möbelobjekte aufgetreten.
- Bei der Reihenfolge in der OnInit()-Methode

  ```
  self.TestAnwendung()
  self.__alleMoebel=[]
  ```

  wurden die Objekte der Testanwendung zwar dargestellt, aber nicht in der Liste angelegt, wie der folgende Dialog in der Shell zeigt:

```
>>> app._RaumplanerApp__alleMoebel
[]
```

Probleme bei der Initialisierung

- Notwendig ist also die Reihenfolge bei der zunächst das Attribut Initialisiert wird und dann erst die Testanwendung aufgerufen wird.

```
self.__alleMoebel=[]
self.TestAnwendung()
```