Persistenz

Eine Lösung des Persistenzproblems, also der Sicherung des Zustandes der Objekte eines OO Systems, liefert Python standardmäßig im Modul **`pickle`**. Aus der Dokumentation:
'The pickle module implements a fundamental, but powerful algorithm for serializing[1] and de-serializing a Python object structure. “Pickling” is the process whereby a Python object hierarchy is converted into a byte stream, and “unpickling” is the inverse operation, whereby a byte stream is converted back into an object hierarchy. Pickling (and unpickling) is alternatively known as “serialization”, “marshalling,” or “flattening”, however, to avoid confusion, the terms used here are “pickling” and “unpickling”.'

Sichern und Wiederherstellen der Objekte

Die beiden wesentlichen Methoden sind **`pickle.dump(...)`** zum Sichern in eine Datei und **`pickle.load(...)`** zum Laden aus einer Datei. Da wir hierbei eben auf Dateien zugreifen, müssen wir uns um die Fehlerbehandlung für fehlerhafte Zugriffe kümmern. Hierzu habe ich mich an den Vorschlägen zur Persistenz im Kapitel 4 des Buches[2] "Python von Kopf bis Fuß" orientiert, habe allerdings nicht mit Strings, sondern gleich mit Objekten gearbeitet.
Um das erste Beispiel nicht zu kompliziert werden zu lassen, habe ich zunächst eine einfache Klasse, zwei Instanzen und eine Liste zum Speichern definiert.

```
class TestObjekte:
    def __init__(self, vorname, name):
        self.vorname = vorname
        self.name = name

    def ZeigeDaten(self):
        return (self.vorname, self.name)

obj1 = TestObjekte('Hans', 'Meier')
obj2 = TestObjekte('Caroline', 'Klinge')
alleObjekte = [obj1, obj2]
```

Die weiteren Abschnitte:

```
import pickle

##### ------- schreiben ------------ #####
print "Objektdaten werden in Datei geschrieben"
try:
    with open('daten.dat', 'wb') as objekt_datei:                    [3]
        pickle.dump(obj1, objekt_datei)
        pickle.dump(obj2, objekt_datei)
```

1 Bei Java wird das Problem dadurch gelöst, dass eine Klasse das Interface Serializable implementiert. Interessanterweise ist dafür keine Methode notwendig. Der Programmtext für das Speichern und Lesen ist dann allerdings deutlich komplizierter bei Java.
2 Paul Barry: Python von Kopf bis Fuß; ISBN 978-3-89721-318-0
3 Der erste Parameter ist klar: Hier wird der Dateiname verlangt. Der zweite Parameter kann bei normalem Lesen aus einer Textdatei weggelassen werden. r kennzeichnet also das Öffnen zum Lesen, w kennzeichnet das Öffnen zum Schreiben. Das nachfolgende b kennzeichnet den Zugriff im Binärformat, also auf eine Binärdatei.
with erzeugt einen eigenen Kontextbereich für die folgenden Anweisungen und gewährleistet ein ordnungsgemäßes Schließen der geöffneten Datei auch nach einem Fehler.

```
except IOError as ioerr:
    print('Dateifehler: ' + str(ioerr))
except pickle.PicklingError as perr:
    print('Pickle-Fehler: ' + str(perr))


##### ------- lesen ------------ #####
alleObjekte = []
print "Objektdaten werden aus Datei gelesen"

try:
    with open('daten.dat', 'rb') as objekt_datei:
        alleObjekte.append(pickle.load(objekt_datei))
        alleObjekte.append(pickle.load(objekt_datei))

except IOError:
    print('Datei nicht auffindbar!')
except AttributeError as ae:
    print ('AttributeError: ' + str(ae))                    [1]
except pickle.UnpicklingError as perr:
    print('Pickle-Fehler: ' + str(perr))
```

Sichern und Wiederherstellen der Objekte

Der entscheidende Aspekt für unsere Anwendung ist, dass mit diesem Verfahren direkt die Objekte gespeichert und wieder geladen werden können. Etwas präziser formuliert muss es heißen *"vollständige Informationen über die Objekte"*, denn es werden natürlich nicht die Objekte selbst gespeichert. Diese Informationen sind in einer Datei abgelegt, können also auch über das Netz oder per Datenträger übertragen werden.

Beteiligte Klassen müssen bekannt sein

Für eine Wiederherstellung des Zustandes ist dabei nur wichtig, dass die beteiligten Klassen in der wiederherstellenden Umgebung bekannt sind.
Da dies ein ernstes Problem sein kann, muss ein dabei möglicherweise auftretender Fehler abgefangen werden.

Die Datei

Liest man die Datei mit den gesicherten Daten mit einem einfachen Editor, erkennt man die Prinzipien wieder, die man selbst auch bei der Sicherung verwenden würde. Klassennamen und Attributnamen sind zu finden und irgendwo verstecken sich die Werte[2].

Lösungen im Programm

Hier noch einmal eine Lösung für den Raumplaner gesondert anzugeben lohnt sich nicht. Interessanter wird eher die Einbindung in die GUI-Struktur mit dem Auslösen der Ereignisse und den Dialogen für Dateiname und Pfad.

1 Diese Fehlerbehandlung habe ich eingefügt, da die nachfolgende nicht wie erwartet den Fehler abgefangen hat, dass die notwendige Klasse nicht vorhanden ist. In einem Anwendungsfall der Raumplaneranwendung war es ein ImportError.
2 Herauszufinden, wie die Werte abgespeichert werden, könnte interessant sein, ist hier aber nicht weiter untersucht.