Python - Übungsblatt 6

Gerald Senarclens de Grancy

March 5, 2007

Übung 1 - Spezielle Funktionen in Klassen

Um ein besseres Bild von Klassen zu bekommen, erstellen wir eine Klasse `Human`, die einige Eigenschaften eines Menschen nachbilden soll.

__init__()

Oft will man beim Erstellen einer Klasse der neu erstellten Instanz von Anfang an ein paar Eigenschaften zuweisen, manchmal auch Eigenschaften, die nachher nicht mehr verändert werden können. Solche könnten in unserem Fall Vorname, Nachname, Geschlecht und Augenfarbe sein. Um bei der Erstellung einer Instanz Argumente übergeben zu können, benötigt man eine `__init__()` Funktion mit den dementsprechenden Argumenten. Wird die Klasse mit
`myInstance = Human()`
instanziert, so müssen in den Klammern nach dem Namen der Klasse die gewünschten Argumente angegeben werden. Unsere init-Funktion soll folgende Signatur haben:
`__init__(first_name, name, sex, eye_color)`

__getattr__()

Mit dieser Funktion kann man Attribute einer Instanz auslesen. Dies geschieht, indem man einen Schlüssel für das gewünschte Attribut übergibt und danach mittels einer Abfrage das gefrage Attribut retourniert. Zum Beispiel:

```
def __getattr__(self, attribute):
   if attribute == "name":
      return self.__name
```

Erstelle nun eine Funktion, die für alle bisherigen Attribute deren Inhalt ausliest.

__str__()

Wenn man ein Objekt "einfach so" als Argument in einem `print` Statement verwenden will, so muss man eine die Repräsentation desselben geeignet definieren. Dies erfolgt durch die Rückgabe eines Strings in der `__str__()` Funktion.

__add__(), __mul__(),...

Man kann auch Operatoren wie `+` und `*` für die Verwendung in einer Klasse überladen. Dies geschieht zum Beispiel mit Funktionen wie `__add__()` und `__mul__()`. Da das Addieren und Multiplizieren von Menschen wenig Sinn macht, definiere diese Funktionen so definieren, dass lediglich eine erklärende Fehlermeldung ausgegeben wird.

Übung 2 - Public und Private

Um den Zugriff auf die Funktionen und Eigenschaften eines Objekts steuern zu können, was zum Beispiel nötig ist, um sicherzustellen, dass nur sinnvolle Werte zugewiesen werden, kann man Variablen und Funktionen als "privat" deklarieren. Das heisst, dass das Argument bzw. die Funktion nicht von ausserhalb aufgerufen oder verändert werden kann. In vielen objektorientierten Programmiersprachen wird für diesen Zweck ein eigenes Schlüsselwort definiert, in Python genügt es jedoch, wenn man __ als Prefix eines Variablen- oder Funktionsnamen anführt. Beachte auch, dass folglich alle in dieser Übung bisher definierten Eigenschaften und Funktionen privat sind, dass also kein direkter Zugriff darauf möglich ist. Füge jetzt noch folgende "public" Funktionen hinzu:

```
def setTitle(self, title)
def getTitle(self)
def getState(self)
def __setState(self, state)
def eat()
def doSport()
def sleep()
```

Übung 3 - Inheritance (Vererbung)

Um viel Arbeit zu ersparen, gib es in objektorientierten Programmiersprachen verschiedene Konzepte wie Inheritance, multiple Inheritance und Interfaces. Python implementiert davon Inheritance und multiple Inheritance, das heisst, das eine Klasse quasi als Tochterklasse einer oder mehrerer anderer Klassen deklariert werden kann. Dabei übernimmt die Tochterklasse alle Eigenschaften und Methoden der Mutterklasse, kann diese aber je nach Bedarf und zur Spezialisierung erweitern und verändern. Die Syntax sieht dabei wie folgt aus:

```
class SubClass(SuperClass):
```

Interfaces sind in Python nicht direkt vorgesehen (Java verwendet diese zum Beispiel, da bei Java nur einfache Inheritance möglich ist, das heisst, jede Klasse kann nur eine Superklasse haben). Wir wollen nur zu unserer Menschenklasse zwei Tochterklassen, `class Man(Human):` und `class Woman(Human):` definieren, die nur die nötigen Funktionen anpasst (zB: ist jede Instanz von Man von männlichem geschlecht).