Informatik Klasse 13, Foliensatz 2
Vererbung und Operatorfunktionen

Prof. G. Kemnitz

Institut für Informatik, Technische Universität Clausthal

19. August 2009

Vererbung

Vererbung bedeutet, dass aus einer oder mehreren Elternklassen eine neue Klasse erzeugt wird, die alle Attribute und Methoden der Elternklassen übernimmt.

```
class abgeleitete_Klasse(Elternklasse):
 {Vereinbarung_von_Argumenten}
 {Vereinbarung_von_Methoden}
```

Beispiel

```
class Schwein():
 Name = ''
 Gewicht = 0
 Hunger = 0
```

Frage: Was bedeuten die geschweiften Klammern?

```
 def __init__(self, n, g, h):
  self.Name   = n
  self.Gewicht = g
  self.Hunger  = h
 def __str__(self):
  s =  'Schwein: Name = ' + self.Name
  s += ', Gewicht = '+ str(self.Gewicht)
  s += ', Hunger = ' + str(self.Hunger)
  return s

class Superschwein(Schwein):
 def __add__(self, x):
  return Superschwein(self.Name + x.Name,
    self.Gewicht + x.Gewicht, self.Hunger + x.Hunger)
a = Superschwein('Felix', 10, 20)
b = Superschwein('Sepp', 20, 14)
print a+b
```

Mehrfachvererbung

Eine Klasse kann auch die Attribute und Methoden von mehreren Klassen erben.

```
class abgeleitete_Klasse(EK1, EK2, ...):
 {Vereinbarung_von_Argumenten}
 {Vereinbarung_von_Methoden}
```

(EK – Elternklasse)

- Auswahl bei namensgleichen Attributen und Methoden: »Tiefensuche von links nach rechts«

  Suchreihenfolge für das Beispiel: EK1, deren Elternklassen von links nach rechts etc. bis zu den Baumblättern, EK2, deren Elternklassen von links nach rechts ...

Operatorfunktionen

- die vordefinierten Operatoren (+, -, *, /, ** etc., Konvertierfunktionen str(), hex() etc.) können für jede Klasse mit einer Spezialmethode implementiert werden

Operator	Spezialmethode
self + other	__add__(self, other)
self - other	__sub__(self, other)
...	...

Kontrollfragen

- Wie lauten die Spezialmethoden für »self * other«, »str(self)« und »self > other«?
  Nachschlagen unter: Python Reference ▷ Operator Redefinition
- Welche der folgenden (Operator-) Funktionen sollte einen Rückgabewert liefern? Welchen Typ sollte der Rückgabewert haben?

```
self < other
str(self)
-self
self += other
len(self)
del self (self löschen)
```

Aufgabe 2.1: Vererbung

Schreiben Sie eine Klasse »Notiz« mit einer Notiz als Zeichenkette und einer Liste von Stichworten. Methoden:

- Konstruktor zur Erzeugung einer Notiz; Übergabeparameter: Notiz als Text und Stichworte als Liste
- Anhängen eines zusätzlichen Stichworts an die Liste
- Addition von zwei Notizen: Spezialmethode für den »+«-Operator, Rückgabewert sei eine Notiz mit den aneinandergehängten Texten und Stichwortlisten der Summanden
- Vergleichsoperatoren für $>$, $<$ und $=$; zu vergleichen sind die Längen der Texte
- Spezielalmethode für die str()-Funktion zur Erzeugung einer Textdarstellung.

Testen Sie die Klasse durch Erzeugung von Instanzen und Methodenaufrufe.

Schreiben Sie eine Klasse »Sortierschema« mit den Attributen Unterichtsfach (Zeichenkette) und Wichtigkeit (Zahl im Bereich von 0 bis 10) mit den Methoden:

- Konstruktor: Übergabe des Unterrichtsfachs als Parameter; Setzen der Wichtigkeit auf »0«
- Spezielmethode für den Operator »self += Zahl« zur Erhöhung der Wichtigkeit um den Wert von Zahl, aber maximal auf 10
- Spezialmethode für den Operator »self -= Zahl« zur Verringerung der Wichtigkeit um den Wert von Zahl, aber nicht unter »0«
- Spezielalmethode für die str()-Funktion zur Erzeugung einer Textdarstellung.

Testen Sie die Klasse durch Erzeugung von Instanzen und Methodenaufrufe.

Schreiben Sie eine Klasse »WNotiz« mit den Elternklassen »Notiz« und »Wichtigkeit«.

- Schreiben Sie ein Konstruktor, dem der Name des Unterrichtsfach und der Notiztext übergeben wird, die Stichwortliste sei leer und die Wichtigkeit »0«
- Testen Sie mit

```
a = WNotiz('Engisch', 'This is stupid')
print a
```

  von welcher der beiden Elternklassen die __str__(sef)-Methode aufgerufen wird?
- Definieren sie anschließend die __str__(self)-Methode der abgeleiteten Klasse neu, so dass alle Attribute der Klasse in der Textdarstellung enthalten sind.
- Testen Sie alle Methoden der abgeleiteten Klasse

Aufgabe 2.2: Roulette-Eltenklasse

In Aufgabe 12.5, Klassse 12 stand:

Wenn man beim Roulette-Spiel auf »rot« setzt, bekommt man mit einer Wahrscheinlichkeit von $p_\mathrm{r} = \frac{18}{37}$ den doppelten Einsatz zurück und verliert mit einer Wahrscheinlichkeit von $1 - p_\mathrm{r} = \frac{19}{37}$ seinen Einsatz. Ein Spieler X geht jeden Abend mit einem Startguthaben von 100 EUR ins Casino und setzt in jedem Spiel 10 EUR auf rot. An wie vielen Spielen kann er im Mittel pro Abend teilnehmen, bis er sein Geld verloren hat.

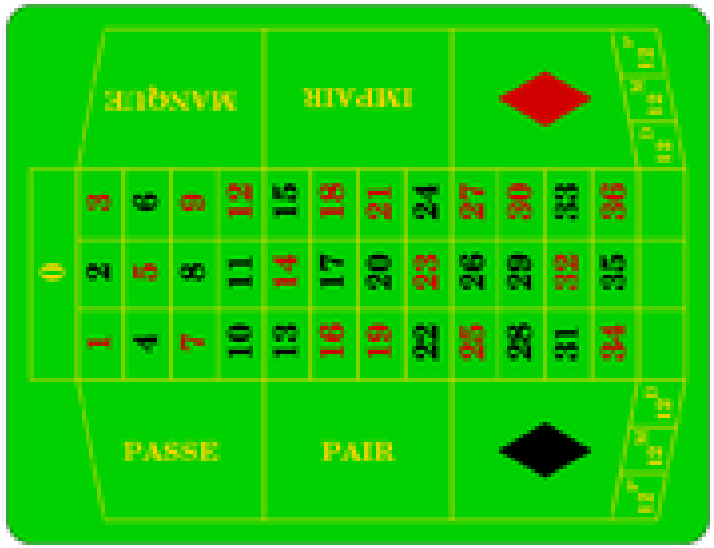
Schreiben Sie eine Klasse Roulett

- mit dem Attributen Guthaben für das aktuelle Guthaben des Spielers
- einem Konstruktor zur Erzeugung einer Instanz mit einem Startguthaben
- einer Methode »Setz_auf_Farbe(self, Betrag), die das Guthaben um den Betrag, der nicht größer als das Guthaben sein darf, mit der entsprechenden Wahrscheinlichkeit verringert oder vergrößert
- der __str__(self)-Methode zur Darstellung des Betrags

Benutzen Sie die Klasse zur Simulation des Spieverlaufs für einen einzelnen Abend.

Aufgabe 2.3: Abgeleitete Roulette-Klasse

Ergänzen Sie in einer von der Basisklasse abgeleiteten Roulettklasse Methoden für andere Arten von Einsätzen, z.B. setze auf eine Reihe und setze auf Zahl, und simulieren Sie zum Test unterschieliche Spielverläufe.



Hinweis: Wenn der Spieler im Mittel gewinnt, haben Sie die Spielmethoden falsch programmiert.