**Programmierkurs Python I – WS 09/10**

# Übung 4

---

## 1 Rationale Zahlen (3 + 2 Punkte)

Definiere eine Klasse, die Bruchzahlen darstellt. Der Konstruktor soll Zähler und Nenner als Argumente nehmen. Die Bruchzahl-Objekte sollen mit den Standard-Python-Operatoren `+`,`-`,`*` und `/` funktionieren:

```
>>> Rat(2,5) + Rat(3,4)
23/20
```

Um das zu erreichen, muss die Klasse die Methoden `__add__(self,b)`, `__sub__(self,b)`, `__mul__(self,b)` und `__truediv__(self,b)` implementieren, die ihrerseits wieder `Rat`-Objekte zurückgeben. Damit die String-Ausgabe für die Objekte wie im Beispiel aussieht, müsst ihr die Methode `__repr__(self)` implementieren, die einen String zurück gibt.

Bonusaufgabe (2 P.): Implementiere den Euklidischen Algorithmus zum Errechnen des größten gemeinsamen Teilers und benutze ihn für eine Methode, die die `Rat`-Objekte kürzt. Wie im Beispiel oben sollen bei allen Rechenoperationen gekürzte Brüche zurückgegeben werden.

## 2 Tic Tac Toe (6 Punkte)

Ziel der Übung ist es, Tic Tac Toe als Python-Spiel für zwei Spielparteien zu implementieren; die Interaktion soll auf der Kommandozeile gesteuert werden. Tic Tac Toe wird auf einem Spielfeld von 3x3 Feldern gespielt, mit zwei Spielern die unterschiedliche Symbole benutzen. Gezogen wird abwechselnd; gewonnen hat, wer zuerst eine Reihe, Spalte oder Diagonale ganz mit seinen eigenen Symbolen füllt.
(Siehe auch: `http://de.wikipedia.org/wiki/Tic_Tac_Toe`)

Wir geben eine Klasse `GameBoard` vor, die verschiedene Funktionalitäten eines Spielbrettes implementiert: Initialisiert werden kann die Klasse mit dem Konstruktor und den Seitenlängen des Brettes (für Tic Tac Toe sind Höhe und Breite 3). Für Eure Methoden braucht ihr hauptsächlich die Methoden zum Setzen eines Symbols auf ein Feld und zum Abfragen, welches Symbol auf welchem Feld steht. Felder werden mit

ihren Koordinaten benannt, bei 0 beginnend. (`(1 1)` ist z.B. das Feld in der Mitte des Tic-Tac-Toe-Brettes) Wir geben Euch die Symbole für die Spieler im Template for. `self.white` ist das Symbol für den Spieler, der anfangen darf, das andere ist `self.black`

Außerdem geben wir Euch ein `TicTacToe`-Template vor, dessen (noch leere) Methoden ihr implementieren müsst:

- Der Konstruktor initialisiert zusätzlich zu unseren Vorgaben u.a. das Spielbrett und alle Hilfsstrukturen, die ihr benutzt.
- `isLegalMove(self ,x,y,s)` soll `True` zurückgeben, wenn man ein Symbol s auf das Feld (`x,y,s`) setzen darf, und sonst `False` (wenn das Feld besetzt ist oder nicht existiert).
- `move(self ,x,y,s)` setzt ein Symbol `s` auf das Feld `(x,y)`
- `evaluateBoard(self)` evaluiert das aktuelle Spielbrett. Die Methode soll -1 zurückgeben, wenn das Spiel noch nicht zu Ende ist; 0 zurückgeben, wenn der erste Spieler gewinnt; 1 für unentschieden; 2 wenn der zweite Spieler gewinnt.

Die Methode `play()` geben wir vor; sie nimmt Spielzüge von abwechselnden Spielern entgegen und verwertet sie, und gibt das aktuelle Brett aus so wie den Gewinner, falls das Spiel zu Ende ist. Sie benutzt die oben angegebenen Methoden. Ziehen kann man durch Eingabe des nächsten Feldes, wobei wir hier die Koordinaten in menschenlesbare konvertieren. (Wenn der Benutzer (2 2) eingibt, ist das die Bret-Mitte, intern ist das mittlere Feld wie gesagt (1 1). Darum braucht ihr Euch nur dann zu kümmern, wenn ihr das Spiel spielt!) Aussehen kann das so:

```
+ ---- + ---- + ---- +
+      +      +  o   +
+ ---- + ---- + ---- +
+      +  x   +  x   +
+ ---- + ---- + ---- +
+  o   +      +  o   +
+ ---- + ---- + ---- +
x's Zug?
1 2
+ ---- + ---- + ---- +
+      +      +  o   +
+ ---- + ---- + ---- +
+  x   +  x   +  x   +
+ ---- + ---- + ---- +
+  o   +      +  o   +
+ ---- + ---- + ---- +
x gewinnt!
```

# 3 Default-Dictionaries (Bonus, 3 Punkte)

In der Vorlesung wurde die Klasse `Defaultdict` vorgestellt. Leider hat die Implementierung eine wichtige Einschränkung: Sie funktioniert nur dann wirklich sinnvoll, wenn der Default-Wert ein unveränderlicher Typ (z.B. eine Zahl) ist. Wie müsste man die Implementierung anpassen, um beispielsweise Dictionaries als Default-Wert zuzulassen?

```
>>> d = Defaultdict(...)
>>> d[47][11] = 23
>>> d[47][11]
23
>>> d[17][4] = 99
>>> d[17][4]
99
>>> d[47][11]
23
```

Versuche, den Lösungsansatz möglichst allgemein zu halten.

---

**Abgabe bis Donnerstag, 26.11.09, 14:00 Uhr** per Mail an
`regneri@coli.uni-sb.de`
`lcarolyn@coli.uni-sb.de`