**Programmierkurs Python II – SS 12**

# Übung 2

---

## 1 Gerichtete Graphen (4 Punkte)

Implementiere eine Datenstruktur für gerichtete Graphen. Du kannst hierfür die in der Vorlesung vorgestellte Klasse `Graph` erweitern. Die folgenden Methoden sollen unterstützt werden:

1. `add_edge(x, y)`
   Eine neue Kante erzeugen
2. `has_edge(x, y)`
   Existiert eine Kante von `x` nach `y`?
3. `indeg(x)` und `outdeg(x)` Eingangs- und Ausgangsgrad eines Knotens
4. `nodes()` und `edges()`
   Listen aller Knoten und Kanten. Alternativ kann auch ein Iterator statt einer Liste zurückgegeben werden.
5. `dfs(x)` und `bfs(x)` Ein Iterator über alle vom Knoten `x` aus per Tiefen- bzw. Breitensuche aus erreichbaren Knoten. (Die in der Vorlesung notierte `dfs`-Methode muss zur Lösung der Aufgabe leicht angepasst werden.)

## 2 Ungerichtete Graphen (2 Punkte)

Implementiere eine Methode `getUndirectedGraph` für die `Graph`-Klasse. Diese Methode soll ein `Graph`-Objekt zurückgeben, in dem die Kanten-Richtungen „entfernt“ wurden. Denkbar wäre folgender Code in einem Skript:

```
graph = Graph()
#...[Knoten und Kanten werden hinzugefügt]
undirected = graph.getUndirectedGraph()
```

Zwischen allen Knoten, die in `graph` adjazent sind, soll auch Adjazenz in `undirected` bestehen, allerdings soll die ursprüngliche Richtung der Kanten nicht mehr erkennbar sein, wenn man z.B. `has_edge(x,y)` aufruft.

Hinweis: Du kannst Dir hierfür eine Klasse `UndirectedGraph` von `Graph` ableiten, die entweder `add_node` und `__init__` geeignet überschreibt, oder `add_node` und `has_edge`. (Wenn die Implementierung der anderen Methoden auf die übrige Datenstruktur zurückgreift, und nicht nur auf diese Methoden, muss sie evtl. auch überschrieben werden.)

# 3 Zyklustest (2 Punkte)

Implementiere eine Methode `hasCycle` für die `Graph`-Klasse. Diese Methode soll testen, ob ein Graph einen Zyklus enthält. Bei korrekter Implementierung liefert diese Methode für die Standard-Graphobjekte als auch die ungerichteten Versionen (siehe Aufgabe 2) das korrekte Ergebnis (also ein Test auf gerichtete und ungerichtete Zyklen gleichermaßen.)

# 4 Graphen Objektorientiert (Bonus, 6 Punkte)

*4 Punkte:* Implementiere eine objektorientierte Variante gerichteter Graphen. Definiere eine Klasse für Knoten, die eine Liste ihrer adjazenten Knoten verwaltet, und eine Klasse für Graphen, die die Knoten verwalten. Die Knoten-Klasse funktioniert analog zu der `Tree`-Klasse aus der vorherigen Vorlesung.

Implementiere für diese Klasse alle Methoden aus Aufgabe 1.

*2 Punkte:* Implementiere Aufgabe 2 und 3 für die objektorientierte Graph-Klasse.

---

**Abgabe bis Donnerstag, 10.05.2012, 08:30 Uhr**