SoSe 2018

Prof. Dr. Margarita Esponda

Objektorientierte Programmierung

3. Übungsblatt

Ziel: Auseinandersetzung mit Rekursion vs. Iteration.

1. Aufgabe (6 Punkte)

Schreiben Sie ein Python-Programm, das mit Hilfe einer Dictionary-Datenstruktur die Häufigkeit der einzelnen Buchstaben eines Texts aufzählt und ausgibt.

Anwendungsbeispiel:

Eingabe: "Hi, Python dictionaries are cool"

Ausgabe: { 'a': 2, ' ': 4, 'c': 2, 'e': 2, 'd': 1, 'i': 4, 'H': 1, 'l': 1, ',': 1,
'o': 4, 'n': 2, 'p': 1, 's': 1, 'r': 2, 't': 2, 'h': 1, 'y': 1}

2. Aufgabe (12 Punkte)

Nehmen Sie an, Sie befinden sich auf einer Party mit **n** Personen, bei der einer der Partygäste ein Gerücht über einen anderen Partygast startet.

1) Jede Person, die das Gerücht zu hören bekommt, erzählt dieses nur an eine andere Person weiter und vermeidet natürlich, das Gerücht der Person, von der die Rede ist, zu erzählen.

2) Wenn jemand das Gerücht bereits kannte und dieses wieder erzählt bekommt, verbreitet diese Person das Gerücht nicht weiter.

a) Schreiben Sie zunächst einmal eine Funktion **gossip**, die die Verbreitung des Gerüchts beim Eingabe einer Gästezahl **m**, simuliert.

b) Programmieren Sie eine Funktion **sim_gossip** die bei Eingabe der Gästezahl **m** und einer ganze Zahl **n** die Simulation **n**-Mal wiederholt und berechnet, wie viele Personen im Durchschnitt das Gerücht erfahren. Testen Sie Ihre Funktion mit n = 1000000.

c) Wie groß ist die Wahrscheinlichkeit, dass alle **m** Personen, mit Ausnahme der Person, über die geredet wird, das Gerücht erfahren? Schreiben Sie eine Funktion **all_get_gossip**, die die Berechnung macht. Testen Sie Ihre Funktion mit **10** Personen.

3. Aufgabe (12 Punkte)

In dieser Aufgabe möchten wir einige Funktionen definieren, die später für ein Spiel verwendet werden können. Das Spiel soll analog zum *Minesweeper*-Spiel funktionieren. Wir haben statt Bomben Löcher. Das Spielfeld soll mit Hilfe einer Matrix modelliert werden.

a) Schreiben Sie eine Funktion, die nach Eingabe der Argumente **n, m** die **n**x**m** Matrix initialisiert.

b) Definieren Sie eine **printSpielFeld** Funktion, die das Feld ausgibt.

c) Schreiben Sie eine Funktion **newSpiel**, die nach Eingabe von **p** und einer **n**x**m** Matrix das Spielfeld (Matrix) initialisiert, in dem mit Wahrscheinlichkeit **p** in jeder beliebigen (x, y)-Position ein Loch vorkommen kann. Sie können dabei mit einem '**.**'-Zeichen die Positionen ohne Löcher und mit einem '**O**'-Zeichen die Positionen mit Löchern markieren.

d) Schreiben Sie eine Funktion **generateSolution**, mit der in jeder Position des Feldes, an der kein Loch existiert, die Anzahl der benachbarten Löcher ausgegeben wird. Wenn keine Nachbarn vorhanden sind, soll das Punkt-Zeichen hinterlassen werden.

Das SpielFeld (Matrix) soll am Ende beispielsweise folgendermaßen aussehen:

```
. 1 1 1 2 2 1 . .

1 O 1 2 O O 1 . .

1 1 1 2 O 3 1 . .

. . . 1 1 1 . . .
```

Wichtige Hinweise:

1) Verwenden Sie geeignete Namen für Ihre Variablen und Funktionsnamen, die den semantischen Inhalt der Variablen oder die Funktionalität der Funktionen darstellen.

2) Verwenden Sie vorgegebene Funktionsnamen, falls diese angegeben werden.

3) Kommentieren Sie Ihre Programme und schreiben Sie Hilfsdokumentation.

4) Verwenden Sie geeignete Hilfsvariablen und Hilfsfunktionen in Ihren Programmen.

5) Löschen Sie alle Programmzeilen und Variablen, die nicht verwendet werden.

6) Schreiben Sie getrennte Test-Funktionen für alle 4 Aufgaben für Ihren Tutor.