# Übungsblatt 6

Willkommen zum Praktikum zu Programmieren in Python. Dies sind Aufgaben, die so in ähnlicher Form durchaus in Testaten vorkommen könnten. Man kann diese Aufgaben mit brutaler Gewalt lösen oder elegant. Elegant ist einfacher, schneller und besser. Sie können mit den Aufgaben selbstständig üben.

**Aufgabe 1.** Schreiben Sie ein Python-Skript `zaehle.py`, das zählt, wie häufig verschiedene Wörter als Kommandozeilenparameter vorkommen. Geben Sie alle Wörter mit den häufigsten Vorkommen in alphabetischer Reihenfolge aus. In Beispiel sind zwei Wörter am häufigsten (zwei mal) vorgekommen.

```
$ ./zaehle.py ein Text wird ein Beispiel das danach weggeworfen wird
2:ein
2:wird
```

**Aufgabe 2.** Implementieren Sie ein einfaches Verfahren zur Komprimierung von Texten. Für jedes Zeichen, das hintereinander mehr als einmal vorkommt ersetzen wir alle diese Zeichen durch einmal dieses Zeichen gefolgt von der Häufigkeit des Vorkommens als Dezimalzahl. Gehen Sie davon aus, dass im Text nur die Zeichen des Alphabets (Groß- und Kleinbuchstaben) vorkommen. Schreiben Sie ein Python-Programm `llk.py`, das alle Kommandozeilenparameter komprimiert und zeilenweise ausgibt.

```
$ ./llk.py abbccc aaabbc aaaabbbccdefg aaaaaaaaaaaaaaaaaaaabb
ab2c3
a3b2c
a4b3c2defg
a20b2
```

**Aufgabe 3.** Schreiben Sie einen Parser, der Dateien im `obj`-Format einliest und ausgibt. Das `obj`-Format besteht aus Zeilen der Form

```
v <x> <y> <z>
```

wobei `<x>`, `<y>`, `<z>` Gleitkommazahlen sind. Es können beliebig viele Whitespaces (`\t\ `) vor und nach den vier Teilen einer Zeile vorkommen. Leerzeilen werden ignoriert, Zeilen mit einer # als erstes Zeichen sind Kommentarzeilen und werden ebenfalls ignoriert. Jede Zeile repräsentiert einen Punkt (Vertice) in einem dreidimensionalen Raum. Schreiben Sie eine Funktion `objread(dateiname)`, die eine `obj`-Datei einliest und eine Liste von 3-Tupeln zurückgibt. Zum Beispiel soll die Datei `simple.obj`

```
v  1.0   1.0   1.0
v  3.0   1.0   1.0
v  1.0   5.0   1.0
v  1.0   1.0   7.0
```

die folgende Liste von Punkten ergeben.

```
[(1.0, 1.0, 1.0), (3.0, 1.0, 1.0), (1.0, 5.0, 1.0), (1.0, 1.0, 7.0)]
```

Schreiben Sie ein Skript `readobj.py`, das als Kommandozeilenparameter einen Dateinamen erwartet und die Liste von Punkten ausgibt. Sie können davon ausgehen, dass das Format der `obj`-Datei korrekt ist.

**Aufgabe 4.** Wir wollen Bilder der Größe 400x400 aus den eingelesenen Punkten erzeugen und anschauen. Bilder erzeugen Sie mit der Python Imaging Library im Modul `PIL.Image`. Mit `im = Image.new("1", (400,400))` erstellen Sie ein neues Bild. Sie setzen einen Pixel an den Punkt $(x, y)$ mit `im.putpixel((x,y),1)`. Sie können ein Bild speichern mit `im.save(dateiname)`. Verwenden Sie für den Dateinamen die Endung `.png`.

Da Bilder nur zweidimensional sind, müssen wir uns auf zwei der drei Zahlen beschränken. Es gibt drei Möglichkeiten: x und y, x und z oder y und z. Schreiben Sie eine Funktion

```
machebild(punkte, name="bild", mode="xy")
```

die eine Bilddatei `bild_xy.png` aus den Punkten in `punkte` erzeugt und speichert. Falls `mode` den Wert `"xz"` oder `"yz"` hat, wird das entsprechende Bild erzeugt und unter dementsprechenden Namen gespeichert. Falls `mode` den Wert `"all"` hat, werden alle drei Bilder erzeugt und gespeichert.

Erstellen Sie ein Skript `genpbild.py`, das als Kommandozeilenparameter einen Dateinamen erwartet und drei Bilder mit passender Endung generiert. Für die Datei `p.obj` sollen zum Beispiel die Bilder `p_xy.png`, `p_xz.png` und `p_yz.png` generiert werden.

**Aufgabe 5.** Das `obj`-Format kann zusätzlich noch Zeilen der Form

```
f <p1> <p2> <p3>
```

haben, wobei `<p1>`, `<p2>`, `<p3>` Integerzahlen sind und je eine Zeile ein Dreieck repräsentiert. Jede Zeile mit einem `f` repräsentiert ein Dreieck mit den Eckpunkten `punkte[p1-1]`, `punkte[p2-1]` und `punkte[p3-1]`. Die `pi` sind also die Indizes in die Liste von Punkten, wobei der erste Punkte den Index 1 und nicht 0 hat.

```
v 100 120 160
v 210 170 180
v 150 210 190
v 170 100 130
f 1 2 4
f 2 3 4
```

Zum Beispiel seien zusätzlich zu vier Zeilen mit `v` (wie vertices) noch zwei Zeilen mit `f` (wie faces) wie in dem links stehenden Beispiel. Dann sind damit zwei Dreiecke gemeint, wobei das erste Dreieck die Eckpunkte (100, 120, 160), (210, 170, 180) und (170, 100, 130) hat.

Schreiben Sie ein Skript `gendbild.py`, in der die Funktion `objread` ein Tupel aus der Liste von Punkten (`v`) und aus der Liste der Dreiecke (`f`) zurückgibt `(punkte, dreiecke)`. Um ein Dreieck in ein Bild zu zeichnen, verwenden Sie das Modul `PIL.ImageDraw`. Das folgende Beispiel zeichnet ein rechtwinkliges Dreieck vom Ursprung.

```
draw = ImageDraw.Draw(im)
draw.polygon([(0,0), (0,10), (10,0)], outline=128)
```

Erweitern Sie die Funktion `machebild` der letzten Aufgabe, so dass nicht die Punkte, sondern die Dreiecke gezeichnet werden. Es sollen wieder jeweils 3 Bilder erzeugt werden.

**Hinweis 1.** `import this` – lesen, verstehen, beherzigen.

moodle Python_Barth