Creative Commons Namensnennung - Nicht kommerziell - Keine Bearbeitungen 4.0 International Lizenz

Unit 13

Dr. Günter Kolousek

21. Juli 2015

Lege wiederum ein Verzeichnis an. Nennes es `24_unit13`! In diesem Verzeichnis sollen alle Dateien der jeweiligen Einheit abgelegt werden.

1 Schulübungen

In diesem Beispielblock werden geübt:

- Operationen auf mehrdimensionalen Listen
- Wiederholung der linearen Suche
- Implementierung der binären Suche
- Vergleich der beiden Varianten bzgl. der Laufzeit
- Verarbeiten von Kommandozeilenparameter

Hier sind die Beispielangaben:

1. Schreibe eine Funktion `linear_search(cities, name)`, die in der Liste `cities` aller Städte, die Stadt mit dem Namen `name` linear durchsucht und diese Stadt zurückliefert.

   Um die Städte zu bekommen, verwende die Funktion `get_cities()` aus dem Modul `cities`. Diese liefert eine Liste von Listen zurück. Jede enthaltene Liste in zurückgelieferten Liste enthält die Daten einer Stadt. Schaue dir dazu die Datei `cities.tab` an und probiere die Funktion `get_cities()` in der Shell aus.

   D.h. diese Funktion soll die Liste der Daten der gesuchten (und gefundenden) Stadt zurückliefern oder `None`, wenn keine gefunden wurde.

2. Schreibe nun eine Funktion `sorted_cities(cities)`, die wiederum die Liste alle Städte bekommt (wie beim vorherigen Punkt) und diese nach dem Städtenamen aufsteigend sortiert zurückliefert. An welcher Position befindet sich der Städtenamen?

3. Schreibe eine Funktion `binary_search(cities, name)`, die wiederum eine Liste aller Städte als erstes Argument bekommt, diesmal jedoch davon ausgeht, dass diese Liste aufsteigend sortiert ist und nach der Stadt mit dem Algorithmus "binary search" sucht.

   Es ist wiederum der gesuchte Datensatz (als Liste) zurückzuliefern, anderenfalls `None`

   Zu Testzwecken baue `print` Anweisungen ein, die die gerade durchsuchten Datensätze anzeigen, damit du den Ablauf besser verfolgen kannst.

4. Schreibe eine Funktion `timing(cities, name)`, die die beiden Algorithmen aufruft und auch jeweils die Zeitdauer misst, die diese jeweils für die Abarbeitung benötigen. Die Ausgabe dieser Funktion soll ca. folgendermaßen aussehen:

   ```
   Linear search benötigt: 0.00334811210632s
   Binary search benötigt: 8.39233398438e-05s

   Binary search benötigt nur 2.50658691163% der Zeit von linear search
   D.h., binary search ist 39.8948863636-Mal schneller!
   ```

   Aber wie misst man die Zeit? Im Modul `time` befindet sich die Funktion `time`, die eine Anzahl von Sekunden als `float` zurückliefert. Ruft man diese Funktion vorher auf, speichert den ersten Wert und ruft die Funktion danach wieder auf, dann kann man diese beiden Werte subtrahieren und man erhält die Zeitdauer.

   Information: In der Informatik hat das Datum 1.1.1970 00:00 UTC (Coordinated Universal Time) eine besondere Bedeutung. Ab diesem Zeitpunktpunkt ("Unix Epoche" oder nur "The Epoch") werden die Anzahl der Sekunden gezählt, wobei die Schaltsekunden mitgezählt werden.

5. Baue das Modul zu einem "Hauptprogramm" um und rufe die Funktion `timing` auf. Verwende eine Stadt aus dem Datensatz, z.B. Neunkirchen.

6. Kommentiere den bisherigen Teil des Hauptprogrammes aus und programmiere das Programm so um, dass der Benutzer zwei Städtenamen als Parameter auf der Kommandozeile mitgeben muss und die Entfernung zwischen diesen beiden Städten ausgegeben wird (verwende dazu die Datei `cities.tab` und das Modul `cities`).

   Die Ausgabe sollte so aussehen:

   ```
   % python3 unit13.py
   usage: python3 unit13.py city1 city2
   % python3 unit13.py Neunkirchen
   usage: python3 unit13.py city1 city2
   % python3 unit13.py Neunkirchen XXX
   Stadt XXX wurde leider nicht gefunden!
   ```

```
% python3 unit13.py Neunkirchen "Wiener Neustadt"
Die Entfernung von Neunkirchen zu Wiener Neustadt beträgt 15.53km
```

Beachte die "usage" Meldungen und die Formatierung der Entfernungsangabe.

Hmm, wie wird das gemacht? Im Modul `sys` gibt es eine Variable `argv`, die den Dateinamen (Programmnamen) und alle Parameter und Optionen in einer Liste enthält. Schreibe ein kleines Programm und probiere aus:

```
import sys
print(sys.argv)
```

7. Baue das Programm nun so um, dass auch mehr als 2 Städtenamen eingegeben werden können. Es soll dann die Gesamtentfernung zwischen allen Städten ausgegeben werden, wenn man diese in der angegebenen Reihenfolge durchreisen will:

```
% python3 unit13.py Neunkirchen "Wiener Neustadt" "Wien, Innere Stadt"\
  "Krems an der Donau"
Die Entfernung von Neunkirchen zu Krems an der Donau beträgt 122.49km
```