Einführung in die Programmierung (Python)

Klausur für den Masterstudiengang Human Factors

8. Mai 2019

1. Eine als PPN bezeichnete Datensatz-Nummer eines Bibliothekskatalogs enthält als letztes Zeichen eine nach dem Modulo-11-Verfahren berechnete Prüfziffer. Konkret gilt:

   - Die PPN ohne Prüfziffer besteht aus einer Folge von **maximal** 10 Ziffern.
   - Den Ziffernstellen werden **von rechts nach links** die Gewichtungsfaktoren 2 bis 11 zugeordnet.
   - Alle Ziffern werden mit dem jeweils zugeordneten Gewichtungsfaktor multipliziert und diese Produkte werden summiert.
   - Aus dieser Summe ergibt sich die Prüfziffer wie folgt, wobei eine 10 als `X` kodiert wird, da nur eine Stelle zur Verfügung steht: $(11 - (\text{Summe} \mod 11)) \mod 11$.

   Beispiel für die PPN 95980479 (ohne Prüfziffer):

   ```
   PPN-Ziffern:  0  0 9 5 9 8 0 4 7 9
   Gewichtung : 11 10 9 8 7 6 5 4 3 2
   Summe      : 0*11 + 0*10 + 9*9 + 5*8 + 9*7 + 8*6 + 0*5 + 4*4 + 7*3 + 9*2 = 287
   Prüfziffer : (11 - (287 mod 11)) mod 11 = 10, kodiert als X
   ```

   Schreiben Sie eine Python-Funktion `pruefziffer(ppn)`, die eine PPN ohne Prüfziffer als String übernimmt (Vornullen können weggelassen werden), die Prüfziffer nach obigem Algorithmus bildet und diese als String zurückgibt. Bei ungültigen Argumenten (z.B. PPNs mit mehr als 10 Ziffern) soll die Funktion eine Ausnahme werfen.

   Implementieren Sie ein Rahmenprogramm, das die Liste der als Kommandozeilenargumente angegebenen PPNs (`sys.argv` ab Index 1) durchläuft und damit jeweils die Funktion `pruefziffer` aufruft. Jede PPN ist zusammen mit ihrer Prüfziffer oder im Falle einer Ausnahme mit einer Fehlermeldung auszugeben, z.B.

   ```
   ./ppn.py 104078376 95980479 11122233345
   104078376 ==> 7
   95980479 ==> X
   11122233345 ==> ungültig
   ```

2. Schreiben Sie ein Python-Programm, das eine **rekursive** Funktion `quersumme(s)` implementiert, die für eine als String (Ziffernfolge) oder Integer-Objekt übergebene Ganzzahl die Quersumme ermittelt und als Integer zurückgibt. Ein negatives Vorzeichen eines Arguments wird vor der Berechnung der Quersumme mittels Betragsfunktion umgekehrt.

   Hinweis: Die Quersumme von 12345 ergibt sich rekursiv z.B. als $5 + \text{Quersumme}(1234)$. Die Ziffern einer Integerzahl lassen sich einzeln ermitteln, indem man die letzte Ziffer mit Modulo 10 abspaltet und sie danach mittels ganzzahliger Division durch 10 entfernt. Alternativ kann man die Integerzahl in einen String konvertieren und dessen Ziffernfolge durchlaufen.

   Rufen Sie die Funktion `quersumme` für jedes Element der Sequenz

   ```
   0, 278, -123, '0004', 0.9, 'auto'
   ```

   und geben Sie die ermittelten Ergebnisse folgendermaßen aus:

```
Quersumme von 0: 0
Quersumme von 278: 17
Quersumme von -123: 6
Quersumme von 0004: 4
ungültige Zahl: 0.9
ungültige Zahl: auto
```

3. Eine Textdatei enthält zeilenweise Kassenbuch-Einträge folgender Art:

```
22.03.2019 0.6
22.03.2018 -20
```

In Spalte 1 steht das Datum einer einzelnen Buchung (Ein-/Auszahlung) im Format TT.MM.JJJJ und in Spalte 2 der zugehörige Geldbetrag im Format einer Float-Zahl. Als Spaltentrenner dient ein Leerzeichen.

Schreiben Sie ein Python-Programm, das alle Zeilen einer solchen Datei sequentiell von der Standardeingabe `sys.stdin` einliest und mit deren Daten die nachfolgend genannten **zwei Dictionaries** passend aktualisiert, wobei generell die **Jahreszahl des Datums als Schlüssel** dient (zur Vereinfachung können Sie den Typ `collections.defaultdict` verwenden):

(a) Dictionary `buchungen`: Pro Schlüssel wird die **Liste** der Buchungszeilen des betreffenden Jahres in Form eines **Tupels (Monat, Tag, Betrag)** gespeichert.

(b) Dictionary `summe`: Als Wert tritt hier in Form eines `float`-Objekts die Summe der Geldbeträge aller Buchungen des jeweiligen Jahres auf.

Hinweis: Zerlegen Sie beim Einlesen jede Zeile in die beiden Spalten und konvertieren Sie den Geldbetrag in ein `float`-Objekt, bevor Sie die beiden Dictionaries aktualisieren. Sie können davon ausgehen, dass alle Zeilen ein gültiges Format haben.

Nach dem Einlesen aller Buchungszeilen sind die Daten der Dictionaries in zwei Blöcken auszugeben:

(a) Ausgabe der Buchungszeilen **aufsteigend sortiert nach dem Jahr** und innerhalb eines Jahres **aufsteigend sortiert nach dem Datum (Monat, Tag)**. Dabei soll jeweils nach allen Buchungszeilen eines Jahres eine Leerzeile folgen.

(b) Ausgabe der Summen pro Jahr **fallend sortiert nach der Summe**. Hinweis: Nutzen Sie den Schlüsselwort-Parameter `key` der Funktion `sorted` zur Angabe einer Funktion, die den Schlüssel (die Jahreszahl) übernimmt und als Wert die zugehörige Summe aus dem Dictionary liefert.

Die Ausgabe soll diese Form haben:

```
Buchungen:
  22.03.2017 10

  11.01.2018 5
  22.03.2018 -20

  11.01.2019 5.5
  22.03.2019 -5.1
  22.03.2019 0.6
  23.03.2019 20

Summen:
  2019 |    21.00
  2017 |    10.00
  2018 |   -15.00
```

Hinweis: Mit `f'{zahl:8.2f}'` lässt sich eine Float-Zahl in der bei den Summen gezeigten Weise formatieren.