Python: Übungsblatt 6 (set, dict)

Programmieraufgaben – Sets und Dictionaries / Beispiel: Lotto-Ziehungen (‚6 aus 49')

1. Erstellen Sie eine neue, nicht veränderbare Menge mit der Bezeichnung `spielkugeln`; die Menge soll die ganzen Zahlen von 1 bis 49 enthalten.

2. Initialisieren Sie zwei neue Mengen: `kugeln_in_trommel` und `kugeln_in_ablage`. Die Trommel soll zunächst alle Elemente aus `spielkugeln` erhalten; die Ablage ist zunächst leer.

3. ‚Ziehen' Sie nun zufällig sechs Zahlen aus der Menge der Trommel, und legen Sie diese in der Ablage ab (nehmen Sie die entsprechenden Werte aus Trommel heraus). Geben Sie über `print` die beiden Mengen sowie eine sortierte Liste mit den gezogenen Zahlen aus.

   Beispielausgabe:
   ```
   gezogene Zahlen: {3, 37, 42, 11, 18, 20}
   verbleibende Kugeln in der Trommel {1, 2, 4, ... , 48, 49}
   Gewinnzahlen: 3 11 18 20 37 42
   ```

   Hinweis: Die ‚Ziehungen' können Sie einzeln (Kugel für Kugel) vornehmen oder über eine entsprechende Funktion des Moduls `random` (s. Dokumentation) über nur einen Aufruf.

4. Im August 2017 finden insgesamt vier Ziehungen statt. Die Ziehungsdaten stehen in einem Tupel zur Verfügung:

   ```
   ziehungsdaten = ("2017-08-05", "2017-08-12", "2017-08-19", "2017-08-26")
   ```

   a) Erweitern Sie das Programm so, dass die vier Ziehungen direkt hintereinander durchgeführt werden. Geben Sie die Zahlen der jeweiligen Ziehung zusammen mit dem Ziehungsdatum auf dem Bildschirm aus.

      Beispielausgabe:
      ```
      Zahlen der Ziehung 2017-08-05: {8, 14, 48, 19, 25, 30}
      Zahlen der Ziehung 2017-08-12: {32, 7, 10, 42, 48, 31}
      Zahlen der Ziehung 2017-08-19: {5, 6, 11, 47, 28, 30}
      Zahlen der Ziehung 2017-08-26: {32, 33, 39, 7, 12, 27}
      ```

   b) Die Ergebnisse der Ziehungen sollen in einem Dictionary abgelegt werden. Das jeweilige Ziehungsdatum soll als Schlüssel für das Dictionary genutzt werden. Die jeweils gezogenen Zahlen sollen als zugehörige Menge gespeichert werden. Geben Sie das Dictionary über `print` auf dem Bildschirm aus.

      Beispielausgabe:
      ```
      Inhalt Dictionary 'ziehungen': {'2017-08-05': {8, 14, 48, 19, 25,
      30}, '2017-08-12': {32, 7, 10, 42, 48, 31}, ...}
      ```

5. Ermitteln Sie alle gezogenen Zahlen und speichern Sie diese in einer Menge `alle_gezogenen_zahlen`. Geben Sie alle gezogenen Zahlen in aufsteigend sortierter Reihenfolge auf dem Bildschirm aus (z. B. als Liste).

Beispielausgabe:
```
Alle gezogenen Zahlen: Alle gezogenen Zahlen: [5, 6, 7, 8, 10, 11, 12,
14, 19, 25, 27, 28, 30, 31, 32, 33, 39, 42, 47, 48]
```

6. Die Ziehungen sollen nun weiter ausgewertet werden.

a) Erzeugen Sie ein Dictionary `zahlen_statistik`, in der jeder mindestens einmal gezogenen Zahl das jeweilige Ziehungsdatum zugewiesen wird; ist eine Zahl bei mehreren Ziehungen aufgetreten, so sollen alle entsprechenden Daten zugewiesen werden (der Schlüssel des Dictionaries ist die gezogene Zahl; der Wert entspricht dem Ziehungsdatum bzw. den Ziehungsdaten). Geben Sie das Dictionary über `print` auf dem Bildschirm aus.

Beispielausgabe (für den Fall, dass die 8 nur am 5.8.17 gezogen wurde und die 48 sowohl am 5.8.17 als auch am 12.8.17) :
```
{8: ['2017-08-05'], 48: ['2017-08-05', '2017-08-12'], ...}
```

b) Geben Sie die Statistik zu den Zahlen (Zahl – Ziehungsdatum) in Form einer Tabelle aus. Die Zahlen sollen dabei in aufsteigender Reihenfolge sortiert sein (je Zahl max. eine Zeile).

Beispielausgabe:
```
Zahl | Ziehung(en)
-----+--------------------------------------------
   5 | 2017-08-01
   6 | 2017-08-01
   7 | 2017-08-12, 2017-08-26
    ...
```

c) Zusatzaufgabe: Im Python-Modul `collections` wird eine Klasse `defaultdict` bereitgestellt. Die Nutzung von `defaultdict` kann die Erzeugung der Zahlenstatistik (6a) erleichtern. Informieren Sie sich in der Dokumentation zu Python über die Klasse und erarbeiten Sie eine Alternativlösung zu 6a.