44. Bundeswettbewerb Informatik - 1. Runde / 1. Aufgabe

Luna Hagemann (Team-ID 00008 (Einzelteam), Teilnahme-ID 78245)

5. Oktober 2025

Inhaltsverzeichnis

1 Lösungsidee

Es wird zuerst gespeichert, zu welchen Zeiten wie viele Bälle benötigt werden, dann werden diese Daten genommen und geguckt, zu welcher/n Zeit(-en) die meisten Bälle gleichzeitig benötigt werden.

Der Code wird durch jede Sportstunde (eine Zeile in der Eingabedatei) durchgehen und jeweils für die entsprechenden Zeiten die Anzahl der Bälle dazu addieren. Da alle Zeiten in ganzzahligen Zeiteinheiten (Stunden) angegeben werden, wird nur für jede Zeiteinheit ein Wert gespeichert, wie viele Bälle benötigt werden. Mit leichten Modifizierungen, wäre es auch möglich für jede Minute, Sekunde oder andere Zeiteinheit zu speichern, wie viele Bälle benötigt werden.

Mit diesem System können die Klassennamen komplett ignoriert werden, da jede Zeile mit überlappenden Zeiten als gleichzeitiger Sportunterricht bewertet wird, auch wenn das gleiche Klassennamen da sein sollte (Es kann zum Beispiel mehrere Klassen mit gleichem Namen geben oder eine Klasse wird geteilt und hat deswegen teilweise überlappenden Sportunterricht).

Außerdem hat mein System keine Kenntnisse über Wochentage, für jede einzigartige Zeichenkette an der Position des Wochentags wird ein neuer „Tag“ in meinem Programm erstellt, wo gleichzeitige Zeiten mit anderen Tagen nichts zu bedeuten haben.

Obwohl die Schule eigentlich nur wissen muss, wie viele Bälle sie kaufen müssen, gibt mein Programm zusätzlich den Tag und die Zeit aus, wo dies als erstes vorkommt. Mit leichten Modifikationen wäre es auch möglich, dass alle Zeiten, wo die maximale Anzahl an Bällen benötigt wird festgestellt wird.

2 Umsetzung

Mein Programm wird zuerst die Eingabe einlesen und in eine Kette von Zeilen (Sportunterrichte) aufgeteilt. Die erste Zeile wird ignoriert werden, da ich in Python einfach bis zur letzten Zeile auslesen kann.

Mit dieser Kette von Unterrichten wird dann eine Schleife durchgeführt, wobei für jeden Unterricht dann für die Zeiteinheiten (Stunden) durchgegangen und die entsprechenden Ballanzahlen in die entsprechende Zeit eingetragen werden. Direkt bei dieser Schleife wird dann immer geguckt, ob in dieser Zeiteinheit die höchste Ballanzahl benötigt wird und diese dann getrennt gespeichert, zusammen mit dem Tag und der Stunde.

Die Speicherung erfolgt dann ein einem dictionary von allen Wochentagen wobei der Wert von jedem Wochentag ein weiteres dictonary ist, welches dann für jede Zeit die Anzahl an benötigten Bällen enthält.

3 Beispiele

Ich habe diesen Code auf alle Beispiele der Beispielseite angewendet, sowie noch einige eigene, spezielle Fälle, die ausnutzen, dass mein Programm keine Kenntnisse über Wochentage hat und irgendwelche Zeiteinheiten verwenden kann.

3.1 Beispiele der Beispielseite

Beispiel-Nr.	Max. Anzahl von Bällen	Tag	Zeiteinheit/Stunde
ball00	60	Montag	14 bis 15
ball01	96	Mittwoch	11 bis 12
ball02	157	Montag	10 bis 11
ball03	152	Freitag	09 bis 10
ball04	31	Montag	10 bis 11
ball05	60	Donnerstag	08 bis 09
ball06	90	Dienstag	09 bis 10
ball07	59	Freitag	10 bis 11

3.2 Eigene Beispiele

Zuerst werden die Beispiele angegeben, diese werden dann unten in einer Tabelle zusammengefasst mit Ausgabe meines Programms.

3.2.1 Beispiel ball_eig00

An diesem Beispiel soll erkennbar gemacht werden, wie mein Programm mit gleichen Klassen umgeht. Selbst wenn eine vermeintlich gleiche Klasse (gleicher Klassenname) gleichzeitig Sport hat, werden trotzdem beide Ballanzahlen benötigt.

```
3
abc Montag 10 13 2
bcd Montag 11 14 3
abc Montag 12 13 5
```

3.2.2 Beispiel ball_eig01

An diesem Beispiel soll erkennbar gemacht werden, wie mein Programm mit anderssprachigen oder generell anderen Wochentagsnamen umgeht. Alle einzigartigen Zeichenketten am Platz des Wochentagnamens werden als eigener Wochentag aufgefasst, wobei gleiche Zeichenketten als gleicher Wochentag aufgefasst werden.

```
5
1a Tuesday 12 15 7
2b Lunes 14 17 11
3c abcdefgh 11 18 13
4d Tuesday 13 17 17
5e abcdefgh 17 19 19
```

3.2.3 Beispiel ball_eig02

An diesem Beispiel soll erkennbar gemacht werden, wie mein Programm mit Zeiteinheiten anders als Stunden umgeht. Egal welche ganze Zahl eingegeben wird, wird dies einfach als Start, bzw. Endzeit gewertet.

```
3
6f Montag -5 2 23
7g Montag -4 27 29
8h Dienstag -1000 5000000 31
```

Beispiel-Nr.	Max. Anzahl von Bällen	Tag	Zeiteinheit/Stunde
ball_eig00	10	Montag	12 bis 13
ball_eig01	32	abcdefgh	17 bis 18
ball_eig02	52	Montag	-4 bis -3

4 Quellcode

```
# INPUT PROCESSING

needed = {} # main dict for storing all amounts per day/hour
# contains one dict for each day, whoch contains the hours
highest = {'n':0, 'day':'', 'h':0} # stores the highest found occurence
# n: amount of balls, day/h: day/hour combination when the balls are needed

for line in inp: # first line was removed when reading in input
# loops through all lessons
    values = line.split(' ') # seperates each column/value
    if not values[1] in needed:  # add new day to needed list if not added yet
        needed.update({values[1]:{}}) # initialize a dict for all the hours
    for h in range(int(values[2]), int(values[3])):  # h: hour
    # loops through all hours in which the lesson is active
        if not h in needed[values[1]]:  # add new hour if not added yet
            needed[values[1]].update({h:0})
            # set it to 0 balls by default, will get increased later on
        needed[values[1]][h] += int(values[4])  # add number of balls needed
        if needed[values[1]][h] > highest['n']:
        # if it's the new highest amount, update it
            highest['n'] = needed[values[1]][h] # save number of balls
            highest['day'] = values[1] # save day
            highest['h'] = h # save hour

# OUTPUT
```