Hochschule Bochum
Bochum University
of Applied Sciences

Fachbereich Wirtschaft
Prof. Blümel, Prof. Bockermann, Prof. Klingspor

Wirtschaftsinformatik 2

Beispielaufgabe Schleifen – erste Statistiken

Aufgabe 1

Programmieren Sie eine Funktion `berechne_statistik(tupelliste)`, der Sie eine Liste von Tupeln übergeben. Die Tupel haben folgende Form:

```
(name, anzahl, einkaufspreis, kategorie, versandart, gewinn)
```

Die Funktion berechnet folgenden statistischen Wert: Summe von Gewinn

Für die unten dargestellte Beispielliste

```
tupelliste = [
 ("PDF Reader", 1, 26.00, "Utilities", "Download", 2.08),
 ("Kalkulation", 5, 82.95, "Office", "CD-Versand", 23.04),
 ("Windows 4711", 2, 112.00, "Betriebssysteme", "Download", 6.74),
 ("Windows 0815", 10, 87.00, "Betriebssysteme", "Download", 26.1),
 ("Writer", 1, 58.45, "Office", "CD-Versand", 5.22)
]
```

gibt die Funktion das Ergebnis 63.18 zurück:

```
ergebnis = berechne_statistik(tupelliste)
print(ergebnis)   # gibt 63.18 aus
```

Lösung:

```
def berechne_statistik(tupelliste):
    summe=0
    for tupel in tupelliste:
        gewinn = tupel[5]
        summe += gewinn
    return summe
```

Der Aufruf erfolgt durch:

```
tupelliste = [
 ("PDF Reader", 1, 26.00, "Utilities", "Download", 2.08),
 ("Kalkulation", 5, 82.95, "Office", "CD-Versand", 23.04),
 ("Windows 4711", 2, 112.00, "Betriebssysteme", "Download", 6.74),
 ("Windows 0815", 10, 87.00, "Betriebssysteme", "Download", 26.1),
 ("Writer", 1, 58.45, "Office", "CD-Versand", 5.22)
]

berechne_statistik(tupelliste)
```

Anmerkungen zur Lösung: Listen können nicht nur, wie in den Beispielen des Buchs, Zahlen oder Strings beinhalten, sondern selbst wieder Listen oder auch Tupel. Dies ist wichtig, wenn man wie im Beispiel oben Daten von Verkaufsvorgängen auswerten möchte.

In unserem Beispiel bestehen die Daten aus 5 Datensätzen (die Zeilen im Beispiel). Jeder Datensatz besteht aus 5 Werten (z.B. "PDF Reader", 1, 26.00, "Utilities", "Download", 2.08), mit den Bedeutungen Name des Produkts, Anzahl, Einkaufspreis, Kategorie, Versandart, Gewinn). Diese Werte gehören zusammen (deswegen sind sie ein Datensatz) und sollen auch nicht geändert werden. Daher ist ein Tupel dafür der richtige Datentyp.

Die Tupel definieren wir durch:

```
tupel1 = ("PDF Reader", 1, 26.00, "Utilities", "Download", 2.08)
tupel2 = ("Kalkulation", 5, 82.95, "Office", "CD-Versand", 23.04)
tupel3 = ("Windows 4711", 2, 112.00, "Betriebssysteme", "Download"
                                    , 6.74)
tupel4 = ("Windows 0815", 10, 87.00, "Betriebssysteme", "Download"
                                    , 26.1)
tupel5 = ("Writer", 1, 58.45, "Office", "CD-Versand", 5.22)
```

Für die gesamten Daten, die ja nun ebenfalls zusammengehören, bietet sich nun eine Liste von Tupeln als Datentyp an:

```
tupelliste = [tupel1, tupel2, tupel3, tupel4, tupel5]
```

Tupel sind also genauso wie Zahlen oder Strings als Elemente einer Liste möglich und werden ebenfalls genauso wie Zahlen oder Strings durch eine kommaseparierte Aufzählung in eckigen Klammern zu einer Liste zusammengefügt.

Den Zwischenschritt der Definition der Tupel können wir uns sparen und die Liste von Tupeln, wie auch weiter oben bereits gezeigt, direkt erzeugen:

```
tupelliste = [
 ("PDF Reader", 1, 26.00, "Utilities", "Download", 2.08),
 ("Kalkulation", 5, 82.95, "Office", "CD-Versand", 23.04),
 ("Windows 4711", 2, 112.00, "Betriebssysteme", "Download", 6.74),
 ("Windows 0815", 10, 87.00, "Betriebssysteme", "Download", 26.1),
 ("Writer", 1, 58.45, "Office", "CD-Versand", 5.22)
]
```

Eine tupelliste ist genauso eine Variable wie eine Variable, die eine Zahl oder einen String enhält, sie hat nur einen anderen Datentyp. Darum kann sie wie jede andere Variable auch an Funktionen übergeben werden. Darum ist der Aufruf

```
berechne_statistik(tupelliste)
```

möglich. In der Funktion `berechne_statistik` erfolgt nun die Berechnung in einer `for`-Schleife:

```
    for tupel in tupelliste:
        gewinn = tupel[5]
        summe += gewinn
```

Weil unsere Tupelliste 5 Werte (unsere 5 Datensätze) umfasst, läuft die Schleife 5 mal.

Durch `for tupel in tupelliste` wird bei jedem Schleifendurchlauf eine Variable mit dem Namen `tupel` erzeugt.

Beim ersten Schleifendurchlauf enthält tupel den Wert:

```
("PDF Reader", 1, 26.00, "Utilities", "Download", 2.08)
```

beim zweiten Schleifendurchlauf

```
("Kalkulation", 5, 82.95, "Office", "CD-Versand", 23.04)
```

beim dritten Schleifendurchlauf

```
("Windows 4711", 2, 112.00, "Betriebssysteme", "Download", 6.74)
                                        usw.
```

Der Gewinn ist das sechste Element eines jeden Tupel. Da die Zählung der Elemente bei Tupeln und Listen wie im Buch beschrieben mit 0 beginnt, haben wir mit `tupel[5]` Zugriff auf den in jedem Tupel gespeicherten Gewinn.

```
gewinn = tupel[5]
```

belegt also die Variable `gewinn` mit dem jeweiligen Gewinn. `gewinn` hat also beim ersten Schleifendurchlauf den Wert 2.08, beim zweiten Schleifendurchlauf den Wert 23.04, beim dritten Schleifendurchlauf den Wert 6.74 usw.

```
summe += gewinn
```

addiert die Gewinne auf, die wir dann mit

```
return summe
```

zurückgeben.

Überlegen sie selber, warum die Zeile

```
summe = 0
```

notwendig ist!

Löschen Sie dazu im Jupyter-Notebook diese Zeile und schauen sich an, was passiert!

Tipp: Übersetzen Sie die Fehlermeldung ins deutsche :-).

Aufgabe 2 (Erweiterung der Aufgabenstellung)

In dieser Aufgabe soll das Programm der ersten Aufgabe etwas erweitert werden. Dazu sollen Sie eine Funktion `berechne_statistik2(tupelliste)` schreiben, der Sie eine Liste von Tupeln übergeben. Die Tupel haben folgende Form:

```
(name, anzahl, einkaufspreis, kategorie, versandart, gewinn)
```

Die Funktion berechnet folgenden statistischen Wert:

- Anzahl der Datensätze
- Summe von Gewinn
- Durchschnitt von Gewinn

Für die unten dargestellte Beispielliste

```
tupelliste = [
 ("PDF Reader", 1, 26.00, "Utilities", "Download", 2.08),
 ("Kalkulation", 5, 82.95, "Office", "CD-Versand", 23.04),
 ("Windows 4711", 2, 112.00, "Betriebssysteme", "Download", 6.74),
 ("Windows 0815", 10, 87.00, "Betriebssysteme", "Download", 26.1),
 ("Writer", 1, 58.45, "Office", "CD-Versand", 5.22)
]
```

gibt die Funktion folgendes Ergebnis zurück:

```
(5, 63.18, 12.636)
```

Lösung:

```
def berechne_statistik2(tupelliste):
    summe = 0
    anzahl = 0
    for tupel in tupelliste:
        gewinn = tupel[5]
        summe += gewinn
        anzahl += 1
    durchschnitt = summe/anzahl
    ergebnistupel = (anzahl, summe, durchschnitt)
    return ergebnistupel
```

Der Aufruf erfolgt wieder durch:

```
tupelliste = [
 ("PDF Reader", 1, 26.00, "Utilities", "Download", 2.08),
 ("Kalkulation", 5, 82.95, "Office", "CD-Versand", 23.04),
 ("Windows 4711", 2, 112.00, "Betriebssysteme", "Download", 6.74),
 ("Windows 0815", 10, 87.00, "Betriebssysteme", "Download", 26.1),
 ("Writer", 1, 58.45, "Office", "CD-Versand", 5.22)
]

berechne_statistik2(tupelliste)
```

Anmerkungen zur Lösung: Die Schwierigkeit hier ist, dass Funktionen grundsätzlich nur einen Wert zurückgeben können. Wert ist allerdings in der Programmierung relativ. Jede Variable hat einen Wert. Eine Variable vom Typ Tupel hat also auch einen Wert, das Tupel, mit dem diese Variable belegt ist.

Die Zuweisung

```
tupel = (6, 27, 54.8, "Bochum")
```

bewirkt, dass die Variable `tupel` den Wert `(6, 27, 54.8, "Bochum")` hat.

Wir lösen unser Problems also dadurch, dass wir die 3 Ergebnisse (Anzahl, Summe, Durchschnitt), die wir berechnen, zu einem Tupel zusammenfassen.

Dies erfolgt durch die Zeile

```
    ergebnistupel = (anzahl, summe, durchschnitt)
```

Und dieses Ergebnistupel geben wir zurück

```
    return ergebnistupel
```

Funktionen können also Variablen beliebiger Typen zurückgeben.

Unsere Funktion `berechne_statistik2` ist sehr ausführlich programmiert.

Wir können dies auch kompakter schreiben:

```
def berechne_statistik3(tupelliste):
    summe = 0
    anzahl = 0
    for tupel in tupelliste:
        gewinn = tupel[5]
        summe += gewinn
        anzahl += 1
    return (anzahl, summe, summe/anzahl)
```