Python - Grundlagen der Programmierung

Aufgabensammlung, Tag 2

Giuseppe Accaputo

g@accaputo.ch

Aufgaben zu:
Boolesche Ausdrücke und Bedingte Anweisungen («Conditionals»)

Aufgabe 3.1: Boolesche Ausdrücke verstehen

– Zu welchem Booleschen Wert evaulieren die folgenden Code-Snippets?

Code	Evaluiert zu:
`(True and False) or (True and True)`	?
`not false and True`	?
`(True or False) and (not False and True) or not (False and False)`	?
`x = 10` `x < 100 and x % 2 == 0`	?

Aufgabe 3.2: Code verstehen

– Was wird auf dem Bildschirm ausgegeben, wenn wir diesen Code ausführen:

CODE

```
x = 100
if x > 0:
  if x < 100:
    print('A')
  elif x > 30 and x % 2 == 0:
    print('B')
  else:
    print('C')

print('D')
```

Aufgabe 3.3: Zahlen vergleichen

– Schreibe eine Funktion `vergleich`, welche zwei Zahlen entgegennimmt und dabei ausgibt, ob die erste Zahl grösser, kleiner, oder gleich der zweiten Zahl ist

– In der folgenden Tabelle findet ihr einige Funktionsaufrufe und die dazugehörigen Ausgaben um euer Programm auf dessen Korrektheit zu überprüfen:

Aufruf der Funktion im Programm	Mögliche Ausgabe auf dem Bildschirm
`vergleich(1,2)`	`Erste Zahl ist kleiner als zweite Zahl`
`vergleich(100,2)`	`Erste Zahl ist grösser als zweite Zahl`
`vergleich(123,123)`	`Beide Zahlen sind gleich gross`

– **Wichtig**: Die Funktion muss für beliebige Zahlen funktionieren

Aufgabe 3.4: Wurzel berechnen – Teil 2

- Implementiere eine Funktion `wurzel2`, die nur ein Ergebnis ausgibt, wenn die Multiplikation beider Zahlen ein positives Ergebnis ergibt
- Gib im Falle ungültiger Eingaben (wenn die Eingaben zu einem negativen Ergebnis führen) eine Fehlermeldung aus
- In der folgenden Tabelle findet ihr einige Funktionsaufrufe und die dazugehörigen Ausgaben um euer Programm auf dessen Korrektheit zu überprüfen:

Aufruf der Funktion im Programm	Mögliche Ausgabe auf dem Bildschirm
`wurzel2(-4,4)`	`Fehler: Produkt ist negativ`
`wurzel2(-3,-3)`	`3`
`wurzel2(1,2)`	`1.4142135`

- **Wichtig**: Die Funktion muss für beliebige Zahlen wie erwartet funktionieren

Aufgabe 3.5: Sandwich-Funktion

– Schreibe eine Funktion `ist_sandwich`, die drei Zahlen `x,y,` und `z` entgegennimmt und `Ist ein Sandwich` ausgibt, wenn `x <= y <= z` erfüllt ist; ansonsten soll `Ist leider kein Sandwich` ausgegeben werden

– In der folgenden Tabelle findet ihr einige Funktionsaufrufe und die dazugehörigen Ausgaben um euer Programm auf dessen Korrektheit zu überprüfen:

Aufruf der Funktion im Programm	Mögliche Ausgabe auf dem Bildschirm
`ist_sandwich(3,1,2)`	`Ist kein Sandwich`
`ist_sandwich(1,3,2)`	`Ist kein Sandwich`
`ist_sandwich(1,2,3)`	`Ist ein Sandwich`

– **Wichtig**: Die Funktion muss für beliebige Zahlen funktionieren

Aufgabe 3.6: Laufen, Auto fahren, oder fliegen

- Schreibe ein Programm, dass den Benutzer fragt wie weit er reisen möchte (in Kilometer)
- Falls er weniger als 3.2 km reisen möchte, so soll das Programm ihm vorschlagen, den Weg zu laufen
- Falls er mehr als 3.2 km und weniger als 100 km reisen möchte, so soll das Programm ihm vorschlagen, bis ans Ziel mit dem Auto zu fahren
- Falls er mehr als 100 km reisen möchte, so soll das Programm vorschlagen die Strecke mit dem Flieger zu durchreisen

- *Tipp 1*: `eingabe = input('Zahl eingeben')` gibt `Zahl eingeben` auf dem Bildschirm aus, wartet auf die Eingabe des Benutzers und speichert diese direkt in die Variable `eingabe` ab.
  - *Tipp 1.1*: Der Wert der Variable `eingabe` ist vom Datentyp *String*, wir benötigen jedoch eine Zahl → Siehe Kapitel *Typumwandlung*

Aufbaben zu:
Funktionen Teil 2 – Rückgabewerte, Wiederverwendbarkeit, und mehr

Aufgabe 4.1: Zahlen vergleichen

- Schreibe eine Funktion `vergleiche`, die folgende Werte zurückgibt:
  - `1`, falls `x > y`
  - `0`, falls `x == y`
  - `-1`, falls `x < y`

- Funktionsaufrufe und erwartete Rückgabewerte:

Aufruf der Funktion im Programm	Rückgabewert
`vergleiche(3,1)`	`1`
`vergleiche(-1,4)`	`-1`
`vergleiche(42,42)`	`0`

Aufgabe 4.2: Umrechnung von Fahrenheit zu Celsius

- Definiere eine Funktion `celsius`, welche eine Temperaturwert f (gegeben in Fahrenheit) entgegennimmt, diesen in einen Celsius Temperaturwert c umwandelt und ihn anschliessend zuückgibt.
- Dabei könnt ihr die folgende Formel für die Umwandlung verwenden:

$$c = (f - 32)\,\frac{5}{9}$$

- Funktionsaufrufe und erwartete Rückgabewerte:

Aufruf der Funktion im Programm	Rückgabewert
`celsius(33.8)`	`0.9999…`
`celsius(50)`	`10`
`celsius(95)`	`35`

Aufgabe 4.3: Kugelvolumen berechnen

– Schreibe eine Funktion `kugel_volumen`, die anhand des Radius r das Volumen $V(r)$ einer Kugel mit

– Die Formel für das Volumen V lautet

$$V(r) = \frac{4}{3} \cdot \pi \cdot r^3$$

– Funktionsaufrufe und erwartete Rückgabewerte:

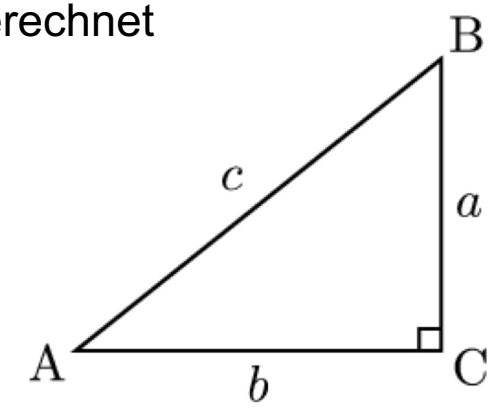
Aufruf der Funktion im Programm	Rückgabewert
`vol(1)` (Einheitskugel)	`4.18879020479` = $\frac{4}{3} \cdot \pi$
`vol(0)`	`0`
`vol(0) + vol(1)`	`4.18879020479`

Aufgabe 4.4: Hypotenuse berechnen

– Schreibe eine Funktion `hypotenuse`, welche abhängig von den Seitenlängen a und b der Katheten eines rechtwinkligen Dreiecks die Länge c der Hypotenuse berechnet

– Die Formel für die Länge $\mathbf{I}$ der Hypotenuse ist gegeben durch

$$c = \sqrt{a^2 + b^2}$$



– Funktionsaufrufe und erwartete Rückgabewerte:

Aufruf der Funktion im Programm	Rückgabewert
`hypotenuse(0,0)`	`0`
`hypotenuse(3,4)`	`5`
`hypotenuse(-1,2)`	`None (inkl. Fehlermeldung ausgeben)`

Aufgaben zu:
Iterationen – Werkzeuge für repetitive Aufgaben

Aufgabe 5.1: Login Abfrage

- Schreibe ein Programm, dass dem Benutzer erlaubt sich mittels Benutzername und Passwort für ein Service einzuloggen
- Dabei hat der Benutzer maximal 3 Versuche um das Login richtig einzugeben, danach wird sein Account gesperrt
  - Der Benutzer soll über eine erfolgte Sperrung via Meldung am Bildschirm informiert werden
- Die Login Daten lauten:
  - Benutzername: `user`
  - Passwort: `mypass`
- **Tipp**: Wir brauchen sicher mal eine Variable, um die Anzahl Versuche zu zählen. Wie könnte dann die Schleifen-Bedingung aussehen?

BILDSCHIRM AUSGABE

```
Benutzername: user
Passwort: mypas
Falsches Login (Fehlversuch 1 von 3)

Benutzername: user
Passwort: mypass
Login erfolgreich!
```

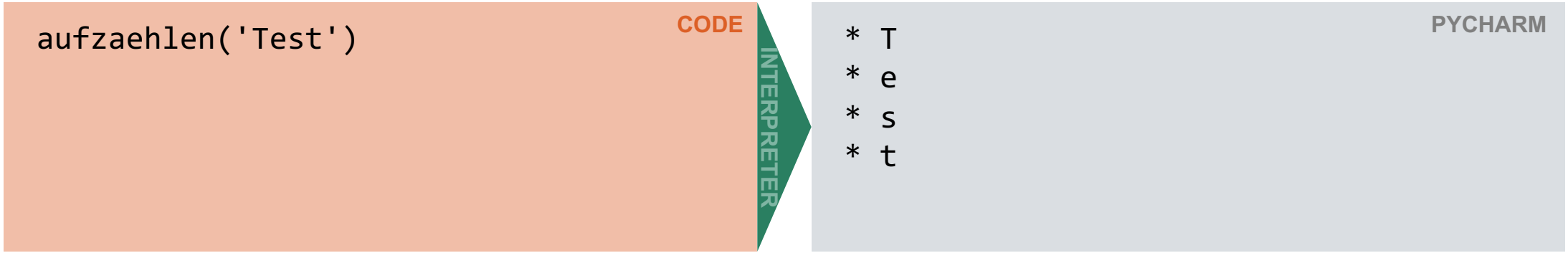
Universität Zürich[UZH]

Zentrale Informatik – IT Fort- und Weiterbildungen

Aufgaben zu: Datenstrukturen

Warm-Up: Strings besser kennenlernen

- Definiert eine Funktion, welche als Argument einen String entgegennimmt
- Die Funktion gibt jedes Zeichen des Strings mittels vorangehender Aufzählung (*) auf einer neuen Zeile aus
- Beispielaufruf der Funktion :



Warm-Up: Listen besser kennenlernen

- Definiert eine Funktion, welche als Argument eine Liste entgegennimmt
- Die Funktion überprüft zuerst die Länge der Liste, diese muss nämlich mindestens 3 Elemente enthalten
- Wenn die Anzahl Elemente korrekt ist, ersetzt die Funktion das 1. und 3. Element mit dem Wert **0**
- Die Funktion gibt danach jedes Element der Liste auf einer neuen Zeile aus

- Beispielaufruf der Funktion :

CODE

```
liste = [1,2,3,4]
liste_anpassen(liste)
```

INTERPRETER

PYCHARM

```
0
2
0
4
```

Warm-Up: Dictionaries besser kennenlernen

- Wir möchten eine Speisekarte erstellen
- Definiert ein Dictionary, dass als Schlüssel ein Gericht hat (`String`), und als Wert den Preis (`Float`)
- Fügt einige Speisen zum Dictionary hinzu
- Die Funktion gibt danach die Preisliste aus (Gericht und Preis müssen ersichtlich sein)

- Beispielaufruf der Funktion :

CODE

```
menu = {
    …
}

preisliste_ausgeben(menu)
```

INTERPRETER

PYCHARM

```
Unsere Preisliste:
 * Burger, 10.5 CHF
 * Pommes, 4.0 CHF
 * Chicken Nuggets, 8.25 CHF
```

Aufgabe 6.1: Vorkommnisse zählen

- Schreibe eine Funktion `anz_vorkommnisse(zeichen, wort)`, welche zurückgibt, wie oft `zeichen` in `wort` vorkommt (auf Grosskleinschreibung achten!)
- *Tipp*: Wir müssen einen Zähler verwenden, der sich merkt, wie oft das Zeichen bereits vorgekommen ist

- Funktionsaufrufe und erwartete Rückgabewerte:

Aufruf der Funktion im Programm	Rückgabewert
`anz_vorkommnisse("a", "Halleluja")`	`2`
`anz_vorkommnisse("e", "Mount Everest")`	`2`
`anz_vorkommnisse("k", "Kathedrale")`	`0`

Aufgabe 6.2: Mitte

– Schreibt eine Funktion `mitte`, welche eine neue Liste zurückgibt, die alle Elemente enthält, ausser dem ersten und letzten Element

– Funktionsaufrufe und erwartete Rückgabewerte:

Aufruf der Funktion im Programm	Rückgabewert
`mitte([1,2,3,4])`	`[2,3]`
`mitte([1,3,2])`	`[3]`
`mitte([3,2,5,20,5,100])`	`[2,5,20,5]`

Aufgabe 6.3: Durchschnitt berechnen

– Schreibt eine Funktion `durchschnitt`, welche den Durchschnitt aus allen Elementen in einer Liste berechnet und zurückgibt (Bedingung: Liste darf nur Zahlen enthalten)

– Funktionsaufrufe und erwartete Rückgabewerte:

Aufruf der Funktion im Programm	Rückgabewert
`durchschnitt([1,2,3,4])`	`2.5`
`durchschnitt([4,18,30,-20])`	`8.0`
`durchschnitt([3,3,3,3])`	`3.0`

Aufgabe 6.4: Grösstes und kleinstes Element finden

- Schreibt eine Funktion `min_max`, welche eine Liste bestehend aus zwei Elementen zurückgibt, wobei das erste Element das kleinste Element in der Liste, und das zweite Element das grösste Element in der Liste ist
  - **Tipp**: `min(liste)` gibt das kleinste Element in `liste` zurück, `max(liste)` jeweils das grösste Element

- Funktionsaufrufe und erwartete Rückgabewerte:

Aufruf der Funktion im Programm	Rückgabewert
`min_max([102,-2,30,400])`	`[-2,400]`
`min_max([-123,430,5000,-300])`	`[-300,5000]`

Aufgabe 6.5: Die Wissensdatenbank

Aufgabe 6.5: Die Wissensdatenbank

- In den nächsten Aufgaben programmiert ihr eine Wissensdatenbank, welche Stichworte und dazugehörige Beschreibungen enthalten kann
- Folgende Funktionalität soll das Programm anbieten:
  - Hinzufügen eines Stichwortes und dessen Beschreibung zur Wissensdatenbank
  - Löschen eines Stichwortes (inkl. Beschreibung) aus der Wissensdatenbank
  - Nach der Beschreibung eines bestimmten Stichwortes in der Wissensdatenbank suchen
  - Auflistung aller Stichworte und der dazugehörigen Beschreibung die aktuell in der Wissensdatenbank vorhanden sind anzeigen

Aufgabe 6.5.1: Die Befehlseingabe

- In einem ersten Schritt sollt ihr die Befehlseingabe implementieren
- Dabei soll das Programm vom Benutzer folgende 5 Befehle solange entgegennehmen, bis `exit` eingegeben wird
  - `insert`: Der Benutzer möchte ein Stichwort inkl. Beschreibung hinzufügen
  - `delete`: Der Benutzer möchte ein Stichwort löschen
  - `list`: Auflistung aller Stichworte (inkl. Beschreibung) anzeigen
  - `exit`: Programm beenden
- In dieser Aufgabe könnt ihr nach der Eingabe des Befehls mal nur den Befehl ausgeben um zu testen, ob die Befehlseingabe auch wirklich funktioniert
- Wenn `exit` eingegeben wird, wird das Programm jedoch beendet

Aufgabe 6.5.1: Die Befehlseingabe

– Das Programm könnte wie folgt aussehen (Benutzereingaben sind **blau** markiert):

BILDSCHIRM AUSGABE

```
Befehl: insert
insert wurde eingegeben
Befehl: delete
delete wurd eingegeben
…
Befehl: exit
Auf wiedersehen!
```

Aufgabe 6.5.2: Einträge auflisten

- Implementiert die Funktionalität für den `list` Befehl
- Wenn der Benutzer diesen Befehl eingibt, so sollen als nächstes gleich alle Einträge in der Wissensdatenbank ausgegeben werden (pro Zeile ein Stichwort inkl. Beschreibung)
- Verwendet eine mit Beispieldaten vorausgefüllte Wissensdatenbank um die Funktion zu testen
- Diesen Befehl werden wir für das Testen der restlichen Befehlen brauchen

Aufgabe 6.5.2: Einträge auflisten

– Das Programm könnte wie folgt aussehen (Benutzereingaben sind **blau** markiert):

BILDSCHIRM AUSGABE

```
Befehl: list
Folgende Eintraege befinden sich in der Wissensdatenbank:
> Stichwort1: Beschreibung zu Stichwort 1
> Stichwort2: Beschreibung zu Stichwort 2
> Stichwort3: Beschreibung zu Stichwort 3

Befehl: exit
Auf wiedersehen!
```

Aufgabe 6.5.3: Einträge hinzufügen

- Implementiert die Funktionalität für den `insert` Befehl
- Nachdem `insert` eingegeben wurde, soll der Benutzer aufgefordert werden, zuerst das Stichwort, und dann die Beschreibung einzugeben
- Falls es bereits ein Eintrag unter diesem Stichwort gibt, so wird der vorhandene Eintrag lediglich überschrieben und der Benutzer über die *Aktualisierung* informiert
- Falls es noch kein Eintrag unter diesem Stichwort gibt, so wird der Eintrag erstellt und der Benutzer über die *Erstellung* informiert
- Teste das Programm mit der Hilfe des `list` Befehls, indem man z.B. in einem ersten Schritt einige Einträge mittels `insert` hinzufügt, und dann in einem zweiten Schritt mittels `list` diese ausgibt und auch überprüft

Aufgabe 6.5.3: Einträge hinzufügen

– Das Programm könnte wie folgt aussehen (Benutzereingaben sind **blau** markiert):

BILDSCHIRM AUSGABE

```
Befehl: list
Keine Einträge vorhanden
Befehl: insert
Stichwort: Stichwort1
Beschreibung: Beschreibung1
Befehl: list
> Stichwort1: Beschreibung1

Befehl: exit
Auf wiedersehen!
```

Aufgabe 6.5.4: Einträge entfernen

- Implementiert die Funktionalität für den `delete` Befehl
- Nachdem `delete` eingegeben wurde, soll der Benutzer aufgefordert werden, das Stichwort einzugeben, dass gelöscht werden soll
- Falls es ein Eintrag unter diesem Stichwort gibt, so wird der vorhandene Eintrag gelöscht und der Benutzer über die *Entfernung* informiert
- Falls es kein Eintrag unter diesem Stichwort gibt, so wird der Benutzer über das *Fehlen dieses Eintrags* informiert
- Teste das Programm mit der Hilfe des `list` Befehls, indem man z.B. in einem ersten Schritt einige Einträge mittels `insert` hinzufügt, dann einige Einträge mittels `delete` löscht, und zum Schluss mittels `list` diese ausgibt und auch überprüft

Aufgabe 6.5.4: Einträge entfernen

– Das Programm könnte wie folgt aussehen (Benutzereingaben sind **blau** markiert):

BILDSCHIRM AUSGABE

```
Befehl: list
Folgende Eintraege befinden sich in der Wissensdatenbank:
> Stichwort1: Beschreibung zu Stichwort 1
> Stichwort2: Beschreibung zu Stichwort 2

Befehl: delete
Stichwort: Stichwort1
Stichwort1 wurde erfolgreich gelöscht
Befehl: list
Folgende Eintraege befinden sich in der Wissensdatenbank:
> Stichwort2: Beschreibung zu Stichwort 2

Befehl: exit
Auf wiedersehen!
```

Aufgabe 6.5.5: Eintrag anzeigen (Optional)

- Implementiert die Funktionalität für den `show` Befehl
- Nachdem `show` eingegeben wurde, soll der Benutzer aufgefordert werden, das Stichwort einzugeben, dass angezeigt werden soll
- Falls es ein Eintrag unter diesem Stichwort gibt, so wird der vorhandene Eintrag angezeigt (Stichwort inkl. Beschreibung)
- Falls es kein Eintrag unter diesem Stichwort gibt, so wird der Benutzer über das *Fehlen dieses Eintrags* informiert
- Teste das Programm mit der Hilfe des `list` Befehls, indem man z.B. in einem ersten Schritt einige Einträge mittels `insert` hinzufügt, und dann in einem zweiten Schritt mittels `show` jeden einzelnen Eintrag versucht auszugeben
  - Weiter kann man mittels `delete` Einträge wieder löschen und dann die gelöschten Einträge mittels `show` versuchen zu anzeigen; dies sollte natürlich nicht gehen

Aufgabe 6.5.5: Eintrag anzeigen (Optional)

– Das Programm könnte wie folgt aussehen (Benutzereingaben sind **blau** markiert):

BILDSCHIRM AUSGABE

```
Befehl: list
Folgende Eintraege befinden sich in der Wissensdatenbank:
> Stichwort1: Beschreibung zu Stichwort 1
> Stichwort2: Beschreibung zu Stichwort 2

Befehl: show
Stichwort: Stichwort2
> Stichwort2: Beschreibung zu Stichwort 2
Befehl: show
Stichwort: Stichwort3
Leider gibt es keinen Eintrag zu Stichwort3

Befehl: exit
Auf wiedersehen!
```