Universität Zürich[UZH]

Zentrale Informatik – IT Fort- und Weiterbildungen

Python - Grundlagen der Programmierung

Giuseppe Accaputo

1 Einführung

1. Schreib ein Programm, dass Deinen Namen ausgibt.

2. Schreib ein Programm, dass auf mehreren Zeilen verschiedene Namen ausgibt.

2 Datentypen, Variablen, Anweisungen, und Ausdrücke

1. Definiere drei Variablen; die drei Variablen sollten dabei alle einen anderen Datentyp haben. Gib danach alle Variablen auf dem Bildschirm aus.

2. Definiere drei Variablen `stadt`, `land`, und `fluss`, und weise allen drei Variablen einen zum Variablennamen passenden Wert (*Frage*: Welchen Datentyp verwenden wir für diese drei Variablen?). Folgendes sollte danach auf dem Bildschirm ausgegeben werden:

```
Stadt: Hier sollte der Wert der stadt Variable stehen
Land: Hier sollte der Wert der land Variable stehen
Fluss: Hier sollte der Wert der fluss Variable stehen
```

   Tipp: Der Datentyp, welcher für die obenerwähnten Variablen verwendet werden sollte bietet einen Operator an, um zwei solcher Werte zu konkatenieren / zusammenzuschliessen.

3. Definiere vier Variablen und weise allen verschieden Zahlen zu. Danach definiere eine neue Variable, welche den Durchschnitt aus den vier Variablen beinhaltet. Gib anschliessend den berechneten Durchschnitt auf dem Bildschirm aus.

   Wichtig: Der Durchschnitt sollte mittels den vier Variablen berechnet werden.

4. Schreibe ein Python Programm, dass den folgenden Verlauf wiederspiegelt:

```
1) Zahl 1 hat Wert 2
2) Zahl 2 hat Wert 8
3) Zahl 3 hat Wert 4
4) Gib (Zahl 1 * Zahl 2) + Zahl 3 auf dem Bildschirm aus
```

   Tipp: Verwende Variablen.

5. Schreibe ein Python Programm, dass den Benutzer zuerst nach seinem Namen fragt. Das Programm soll nach Eingabe des Namens «Hallo, <Name>!» auf dem Bildschirm ausgeben.

   Tipp: `eingabe = input('Dein Name: ')` gibt `Dein Name:` auf dem Bildschirm aus, wartet auf die Eingabe des Benutzers und speichert die Eingabe nach Drücken der ENTER Taste in die Variable `eingabe` ab.

6. Schreibe ein Python Programm, dass zwei ganze Zahlen von der Eingabe einliest und dabei die Summe beider Zahlen auf dem Bildschirm ausgibt. Gib anschliessend auf dem Bildschirm `Das Ergebnis lautet <SUMME>` aus, wobei statt `<SUMME>` das Ergebnis angezeigt werden sollte. *Frage*: Wie oft müssen wir `input` aufrufen um zwei Eingaben zu lesen?

   Tipp: Wir können hier `zahl1 = input('Zahl 1 eingeben: ')` verwenden um die Zahl einzulesen. Dabei sei zu beachten, dass der Wert der Variable `zahl1` den Datentyp String hat, unabhängig davon ob wir eine Zahl eingeben oder nicht. Wir möchten jedoch zwei Zahlen miteinander addieren. ***Frage***: Welcher Datentyp wäre für die beiden eingelesenen Zahlen erwünscht, damit die Addition das korrekte mathematische Resultat liefert? Und wie können wir den ursprünglichen Datentyp in einen passenden Datentyp umwandeln? Das Stichwort hier ist *Typumwandlung*.

3 Bedingte Anweisungen

1. Zu welchem Booleschen Wert werden die folgenden Code-Snippets ausgewertet? *Wichtig*: Nicht mit PyCharm ausführen, sondern kurz von Hand lösen

```
(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
```

2. Was wird beim Ausführen des folgenden Codes auf dem Bildschirm ausgegeben?

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

3. Schreibe ein Programm, dass zwei Zahlen einliest und dabei ausgibt, ob die erste Zahl grösser, kleiner, oder gleich der zweiten Zahl ist.

4. Schreibe ein Programm, dass zwei Zahlen entgegennimmt und die Wurzel aus der Multiplikation beider Zahlen nur dann ausgibt, wenn die Multiplikation ein positives Ergebnis ergibt. Im Falle ungültiger Eingaben (wenn die Eingaben zu einem negativen Ergebnis führen) soll die Funktion eine Fehlermeldung auf dem Bildschirm ausgeben.

   Tipp: Die `math.sqrt(x)` Funktion berechnet die Wurzel von x. Diese findest Du in der `math` Library, welche Du mit `import math` importieren und dann mittels `math.sqrt(x)` verwenden kannst.

5. Schreibe ein Programm, das den Benutzer fragt wie weit er reisen möchte. Dabei sollte abhängig von der angegebenen Distanz folgende Ausgaben auf dem Bildschirm anzeigen:
   a. Falls der Benutzer weniger als 3.2 km reisen möchte, so soll das Programm ihm vorschlagen, den Weg zu laufen
   b. Falls der Benutzer 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 der Benutzer mehr als 100 km reisen möchte, so soll das Programm vorschlagen die Strecke mit dem Flieger zu durchreisen

4 Funktionen Teil 1 – Ein Einstieg

1. Definiere eine Funktion `summe` welche drei Zahlen entgegennimmt und dabei die Summe aus den drei Zahlen auf dem Bildschirm ausgibt.

   Tipp: Werte, welche Funktionen entgegennehmen können nennt man *Parameter* oder auch *Argumente* (siehe Slides)

2. Definiere eine Funktion `durchschnitt` welche vier Zahlen entgegennimmt und dabei den Durchschnitt aus den vier Zahlen auf dem Bildschirm ausgibt.

3. Definiere eine Funktion `jahre_bis_100`, die den Namen und das Alter einer Person entgegennimmt. Die Funktion soll danach auf dem Bildschirm ausgeben in welchem Jahr die angegebene Person 100 Jahre alt wird.

5 Funktionen Teil 2 – Rückgabewerte, Wiederverwendbarkeit, und mehr

1. Schreibe eine Funktion vergleiche die zwei Parameter `x` und `y` entgegennimmt und:
   - *1* zurückgibt falls `x` grösser als `y` ist
   - *0* zurückgibt falls `x` gleich `y` ist
   - *-1* zurückgibt falls `x` kleiner als `y` ist

   Wichtig: Die Funktion soll den Wert *zurückgeben*, und nicht auf dem Bildschirm ausgeben. Das Stichwort hier ist *Rückgabewerte*.

2. Schreibe eine Funktion `fahrenheit_to_celsius`, welche einen Temperaturwert f (gegeben in Fahrenheit) entgegennimmt und dabei den umgerechneten Wert in Celsius zurückgibt. Für die Konvertierung von Fahrenheit f zu Celsius c kannst Du die folgende Formel verwenden:

   $c = (f - 32)\frac{5}{9}$ *Tipp*: Der Aufruf `fahrenheit_to_celsius(50)` sollte *10* und `fahrenheit_to_celsius(95)` sollte *35* zurückgeben.

3. Schreibe eine Funktion `kugel_volumen` die anhand des Radius r das Volumen $V(r)$ einer Kugel berechnet. Die Formel für das Volumen $V(r)$ lautet dabei

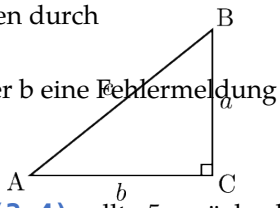
   $$V(r) = \frac{4}{3} \cdot \pi \cdot r^3$$

   Die Konstante π findest Du in der `math` Library, welche Du mit `import math` importieren und dann mittels `math.pi` verwenden kannst.

   Tipp: Der Aufruf `kugel_volumen(0)` sollte *0* und `kugel_volumen(1)` sollte *4.18879020479* = $\frac{4}{3} \cdot \pi$ zurückgeben.

4. Schreibe eine Funktion `hypotenuse`, welche abhängig von den Seitenlängen a und b der Katheten eines rechtwinkligen Dreiecks die Länge c der Hypotenuse berechnet und zurückgibt. Die Formel für die Länge **c** der Hypotenuse ist gegeben durch

   $c = \sqrt{a^2 + b^2}$ Des Weiteren sollte die Funktion bei negativem a oder b eine Fehlermeldung ausgeben und `None` zurückgeben.

   Tipp: Der Aufruf `hypotenuse(0,0)` sollte *0* und `hypotenuse(3,4)` sollte *5* zurückgeben; der Aufruf `hypotenuse(-1,2)` stattdessen sollte eine Fehlermeldung auf dem Bildschirm ausgeben und `None` zurückgeben.

6 Datenstrukturen und mehr: Listen, Strings, und Dictionaries

1. Schreibe eine Funktion `string_ausgeben` welche als Parameter einen String entgegennimmt und jedes Zeichen des Strings auf einer neuen Zeile ausgibt. Dabei soll vor jeder Zeile ein `*` dargestellt werden.

   Tipp: Der Aufruf `string_ausgeben("Test")` sollte folgendes auf dem Bildschirm ausgeben:

   ```
   * T
   * e
   * s
   * t
   ```

2. Schreibe eine Funktion `liste_anpassen`, welche als Parameter eine einzelne Liste entgegennimmt. Die Funktion überprüft zuerst die Länge der Liste. Enthält die Liste weniger als 3 Elemente, so gebt eine Fehlermeldung aus auf dem Bildschirm. Sind stattdessen mindestens 3 Elemente vorhanden, so ersetzt die Funktion das 1. Und 3. Element der Liste mit dem Wert 0. Die Funktion soll danach jedes Element der Liste auf einer neuen Zeile ausgeben.

3. 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.

   Tipp: Der Aufruf `anz_vorkommnisse('c', 'cabcccab')` sollte *4* zurückgeben.

4. Schreibe eine Funktion `durchschnitt`, welche eine Liste als Parameter entgegennimmt und dabei den Durchschnitt aus allen enthaltenen Elementen zurückgibt.

   Tipp: Der Aufruf `durchschnitt([1,2,3,4,5])` sollte *3.0* zurückgeben.

5. Schreibe eine Funktion `listen_summe`, welche zwei Listen als Parameter entgegennimmt und dabei jedes Element der einen Liste mit dem Element an der gleichen Position der anderen Liste summiert. Die Funktion soll dabei eine Liste mit den berechneten Summen zurückgeben.

   Wichtig: Die beiden Liste können nur summiert werden, wenn sie gleich viele Elemente enthalten. Sollten die Listen nicht gleich lang sein, so gebt eine Fehlermeldung aus auf dem Bildschirm.

   Tipp: Der Aufruf `listen_summe([1,2,3], [4,5,6])` sollte die Liste `[5,7,9]` zurückgeben

6. Schreibe eine Funktion `listen_konkatenieren`, welche zwei Listen als Parameter entgegennimmt und dabei die Konkatenation beider Listen zurückgibt.

   Tipp: `listen_konkatenieren([1,2,3],[10,9,8])` sollte `[1,2,3,10,9,8]` zurückgeben.

7. Schreibe eine Funktion `minimum`, welche eine Liste als Parameter entgegennimmt und dabei das kleinste Element zurückgibt.

   Tipp: `minimum([2,1,3])` sollte *1* zurückgeben.

8. Schreibe eine Funktion `finde_index(element, liste)`, welches den Index von `element` in `liste` zurückgibt. Falls `element` nicht in `liste` enthalten ist, so soll die Funktion *-1* zurückgeben.

   Tipp: Der Aufruf `finde_index(100, [1,2,100,4,5])` sollte *2* zurückgeben, der Aufruf `finde_index(76, [1,2,100,4,5])` stattdessen *-1*.

Weitere Aufgaben zu Listen findest Du unter
https://www.w3resource.com/python-exercises/list/

9. Schreibe eine Funktion `dictionaries_mergen`, welche zwei Dictionaries als Parameter entgegennimmt und beide Dictionaries in ein neues Dictionary zusammenführt. Dabei sollte das neue Dictionary zurückgegeben werden.

   Tipp: `dictionaries_mergen({'a':100,'b':42},{'c':41,'d':32})` sollte `{'a':100,'b':42,'c':41,'d':32}` zurückgeben.

10. Schreibe eine Funktion `dictionary_erstellen(keys, values)`, welche zwei Listen `keys` und `values` als Parameter entgegennimmt und daraus ein Dictionary mit den angegebenen Schlüsseln und Werten erstellt.

    Tipp: `dictionary_erstellen(['a','b'], [100,200])` sollte `{'a':100,'b':200}` zurückgeben.

11. Schreibe eine Funktion `dictionaries_addieren`, welche zwei Dictionaries als Parameter entgegennimmt, diese in ein neues Dictionary zusammenführt und bei Einträgen welche in beiden Parameter-Dictionaries vorhanden sind wird die Summe der beiden Werte abgespeichert.

    Tipp: `dictionaries_addieren({'a':100,'b':42},{'b':200,'d':32})` sollte `{'a':100,'b':242,'d':32}` zurückgeben (der Eintrag mit dem Schlüssel `b` kommt in beiden Parameter-Dictionaries vor).

Weitere Aufgaben zu Dictionaries findest Du unter https://www.w3resource.com/python-exercises/dictionary/

7 Die Wissensdatenbank: Dictionaries im Einsatz

1. *Beschreibung*: In den nächsten Aufgaben programmierst Du 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

2. In einem ersten Schritt sollst Du die Befehlseingabe implementieren. Dabei soll das Programm vom Benutzer folgenden 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
   - `show`: Zeige ein bestimmtes Stichwort und dessen Beschreibung an
   - `exit`: Programm beenden

   In dieser Aufgabe kannst Du 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.

   Tipp: Eine mögliche Ausführung des Programms könnte wie folgt aussehen (Benutzereingaben sind im Folgenden **`schwarz`** markiert; Ausgaben, die euer Programm machen sollte sind stattdessen `hellblau` markiert):

```
Befehl: insert
insert wurde eingegeben

Befehl: delete
delete wurde eingegeben

Befehl: exit
exit wurde eingegeben, Programm wird verlassen!
```

3. 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.

Tipp: Eine mögliche Ausführung des Programms könnte wie folgt aussehen (Benutzereingaben sind im Folgenden **schwarz** markiert; Ausgaben, die euer Programm machen sollte sind stattdessen **hellblau** markiert):

```
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
exit wurde eingegeben, Programm wird verlassen!
```

4. 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

   Tipp: Eine mögliche Ausführung des Programms könnte wie folgt aussehen (Benutzereingaben sind im Folgenden **schwarz** markiert; Ausgaben, die euer Programm machen sollte sind stattdessen **hellblau** markiert):

```
Befehl: list
Keine Einträge vorhanden

Befehl: insert
Stichwort: Stichwort1
Beschreibung: Beschreibung1

Befehl: list
> Stichwort1: Beschreibung1

Befehl: exit
Auf wiedersehen!
```

5. 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

Tipp: Eine mögliche Ausführung des Programms könnte wie folgt aussehen (Benutzereingaben sind im Folgenden **`schwarz`** markiert; Ausgaben, die euer Programm machen sollte sind stattdessen **`hellblau`** markiert):

```
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!
```

6. 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 nichts anzeigen.

   Tipp: Eine mögliche Ausführung des Programms könnte wie folgt aussehen (Benutzereingaben sind im Folgenden **`schwarz`** markiert; Ausgaben, die euer Programm machen sollte sind stattdessen **`hellblau`** markiert):

```
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!
```