Python Wiederholung 1

Ergänzungsfach Informatik, 2021/2022, pro@kswe.ch

16. November 2021

1 Werte, Typen und Variablen

Wir erklären zunächst die drei Begriffe und geben Beispiele für die Programmiersprache Python.

Erklärung 1.1 (Werte) *Programme arbeiten mit Werten (eng. values). Ein Wert ist ein Basiselement jeder Programmiersprache. Programme manipulieren Werte. Werte in einem Programm können eingetippt werden (durch den Programmierer oder Benutzer des Programms) oder erzeugt werden (z.B. Zufallszahlen).*

Beispiel: Ein Zahlenwert ist `5` (ganze Zahl) oder `2.5` ("Kommazahl" - Achtung Punkt statt Komma verwenden!). Einen Text muss man zwischen *doppelten Anführungszeichen* notieren. Ein Beispiel dafür ist `"Hello World!"`.

Erklärung 1.2 (Typen) *Werte werden in verschiedene Typen gruppiert. Jeder Wert besitzt einen Typ. Der Typ bestimmt, was man mit einem Wert "machen darf" (zum Beispiel addieren, dividieren, verbinden, sortieren etc.)*

Beispiel: Es gibt Standardtypen (built-in types), welche automatisch durch die Installation des Python-Interpreters verfügbar sind:

- `int` (Abkürzung für Integer): Alle ganzen Zahlen (z.B. `42`, `-5` oder `0`) besitzen diesen Typ.
- `float` (Abkürzung für Floating Point Number): Alle "Kommazahlen" (z.B. `3.14`) besitzen diesen Typ.
- `str` (Abkürzung für String): Alle Werte zwischen zwei doppelten Anführungszeichen gehören zu diesem Typ. Für String wird auch manchmal der Begriff Zeichenkette benutzt.
- `bool` (Abkürzung für Boolean): Die Wahrheitswerte `True` und `False` besitzen diesen Typ.

Erklärung 1.3 (Variablen) *Wir können Werte im Computerspeicher abspeichern und später wiederverwenden oder ändern. Dazu verwenden wir Variablen. Eine Variable ist ein Speicherplatz mit einem Namen. Der Inhalt kann überschrieben werden. Spätestens wenn das Programm beendet wird, ist der Inhalt jeder Variable gelöscht.*

Beispiel: Mit dem Zuweisungsoperator `=` können wir einen Wert in einer Variablen speichern. Folgendes Programm verwendet drei Variablen. In Zeile 4 wird der Inhalt der Variablen `a` überschrieben.

```
1 a = 25
2 apfelbaum = 3.14
3 c = "Python!"
4 a = 2
```

Eine Zuweisung wird **immer** von rechts nach links verarbeitet. Erst werden die Befehle rechts von `=` verarbeitet. Das Ergebnis wird dann in der Variablen links von `=` gespeichert. Der Name *Variable* soll verdeutlichen, dass sich der Inhalt der Variable (das heisst der gespeicherte Wert) ändern kann - er ist variabel.

2 Eingebaute Funktionen

Python stellt eine Anzahl von Funktionen zur Verfügung, welche man ohne Import-Anweisung direkt verwenden kann. Man verwendet eine Funktion, in dem man die Funktion aufruft. Man führt einen Funktionsaufruf aus.

Erklärung 2.1 (Funktionsaufruf) *Für einen Funktionsaufruf muss man den Namen der Funktion notieren. Nach dem Namen folgen zwei runde Klammern. Zwischen den runden Klammern kann ein Wert übergeben werden. Man nennt diesen Wert auch Übergabewert oder Argument. Bei manchen Funktionsaufrufen ist das Argument optional. Syntax:* `FUNKTIONSNAME(ARGUMENT)`.

Beispiele: Manche Funktionen haben "nur" einen Effekt (z.B. eine Ausgabe ausführen). Andere Funktionen berechnen etwas und geben ein Ergebnis zurück, welches man in einer Variablen speichern kann.

```
# print gibt das Argument in der Konsole aus.
print("Hi!")

# type bestimmt fuer das Argument den Typ und gibt diesen als
# Ergebnis zurueck. Das Ergebnis kann man dann verwenden.
b = type(-1)

# pow berechnet die Potenz zweier Zahlen. Es muessen zwei Argumente
# uebergeben werden: Basis und Exponent. Das Ergebnis ist dann
# die Potenz.
p = pow(2, 16)
print(p) # Gibt den Inhalt der Variablen p in der Konsole aus.
```

3 Kurzaufgaben

Erstellen Sie in PyCharm einen Ordner `python_wiederholung`. Erstellen Sie darin eine neue Python-Datei `wdh_1.py`. Lösen Sie darin die folgenden Teilaufgaben (einfach "untereinander" notieren).

1. Verwenden Sie die `type` Funktion um für `42`, `4.2`, `"42"` und `4,2` den Typ zu bestimmen. Geben Sie den Typ auch jeweils in der Konsole aus.

2. Speichern Sie jedes Wort aus *All work and no play makes Jack a dull boy.* in einer **eigenen** Variablen. Geben Sie den Satz dann in **einer Linie** in der Konsole aus.

3. Man in Python natürlich auch Rechnen. Bestimmen Sie mit Python das Ergebnis von $6 \cdot 1 - 2$. Geben Sie das Ergebnis in der Konsole aus. Das Ergebnis sollte 4 sein. Wie kann man die Berechnung ändern, sodass das Ergebnis -6 lautet?

4. Folgendes Programm ergibt den Fehler `NameError: name 'bruce' is not defined`.

   ```
   ergebnis = bruce + 4
   ```

   Wie kann man das Programm korrigieren, sodass 10 ermittelt wird?

5. Berechnen Sie die Potenz von 2 und 12. Speichern Sie das Ergebnis ab. Geben Sie den Typ und das Ergebnis in der Konsole aus.

6. Was bewirkt folgendes Programm?

   ```
   x1 = "Apfel"
   x2 = "baum"
   x3 = x1 + x2
   print(x3)
   ```

4 Listen

Eine Liste kann mehrere Werte speichern. Deshalb ist eine Liste ein zusammengesetzter Datentyp, man sagt auch Datenstruktur. Listen kann man, wie andere Werte auch, in einer Variablen speichern. Man kann den Inhalt einer Liste ändern (Werte ersetzen, Werte hinzufügen oder Werte löschen).

Erklärung 4.1 (Listen) *Eine neue Liste wird durch **eckige Klammern** erstellt. Zwischen den eckigen Klammern werden die Werte notiert - man spricht von **Elementen**. Zwei Elemente werden durch ein **Komma** getrennt. Syntax:* `[ELEMENT_1, ELEMENT_2, ELEMENT_3, ..., LETZTES_ELEMENT]`

Beispiele: Listen sind sehr flexibel. Man kann auch Werte mit unterschiedlichem Typ darin speichern. Der Zugriff auf ein Element (siehe Zeile 7) verändert die Liste **nicht**. Das Element bleibt in der Liste vorhanden und wird nicht entfernt.

```
1  # Speichert eine Liste mit drei Elementen in einer Variablen ab.
2  einkaufsliste = ["Apfel", "Birne", "Cola"]
3  print(einkaufsliste)
4
5  # Auf jedes Element kann mit dem Index (eine ganze Zahl)
6  # zugegriffen werden. Wir beginnen mit 0 zu zaehlen.
7  ele = einkaufsliste[0]
8  print(ele)
9  print(einkaufsliste[2])
10 einkaufsliste[1] = "Papier"
11 print(einkaufsliste)
```

Man kann auch mehrere Elemente aus einer Liste auslesen. Die Liste bleibt auch hier **unverändert**. In Python nennt man dies *Slicing*. Man "schneidet" (besser kopiert) einen Teil aus der Liste. Der Doppelpunkt `:` innerhalb der eckigen Klammern ist der Slice-Operator: `start_index:stop_index:step`. Wenn man `step` weglässt, dann wird automatisch 1 benutzt. Das Element mit dem `stop_index` wird **nicht** kopiert. Es ist lediglich die obere Grenze.

```
1  zahlen = [1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024]
2  anzahl = len(zahlen) # len bestimmt die Anzahl der Elemente
3  print(anzahl)
4  print(zahlen[0:3]) # Elemente mit Index 0, 1, 2 (ohne 3)
5  tmp = zahlen[8:11] # Elemente mit Index 8, 9, 10
6  print(tmp)
7  print(zahlen[5:]) # Alle Elemente ab und mit Index 5
8  print(zahlen[:5]) # Ab und mit Index 0 bis (aber ohne) Index 5
9  print(zahlen[0:11:2]) # Ab Index 0, bis Index 11 in 2er-Schritten
```

5 Kurzaufgaben

Erstellen Sie in `wiederholungen` eine Python-Datei `wdh_2.py`. Lösen Sie dann folgende Teilaufgaben.

1. Erstellen Sie eine Liste mit den drei Namen Tick, Trick und Track.
2. Bestimmen Sie für den Code `chaos = ["42", 42.0, 42]` den Typ der Liste und den Typ für jedes Element. Schreiben Sie dafür die passenden Code-Zeilen.
3. Speichern Sie den Vektor $\vec{a} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$ und den Vektor $\vec{b} = \begin{pmatrix} 4 \\ 5 \\ 6 \end{pmatrix}$ jeweils in einer Liste. Berechnen Sie dann mit Python das Skalarprodukt der beiden Vektoren und geben Sie das Ergebnis in der Konsole aus.

4. Man kann Listen auch verschachteln - eine Liste von Listen. Erstellen Sie mit diesem Konzept das aktuelle ATP Ranking:

    1. Djokovic, 12113 Punkte
    2. Medvedev, 10220 Punkte
    3. Nadal, 8270 Punkte
    4. Tsitsipas, 8000 Punkte
    5. Zverev, 7340 Punkte

    Pro Tennis-Spieler soll der Nachname (String) und die Punktzahl (Integer) gespeichert werden.

5. Möchte man neben den eingebauten Funktionen auch weitere, bereits existierende Funktion verwenden, dann kann man die entsprechenden Funktionen importieren. Auf Funktionen rund um den Zufall kann man mit dem Import des `random`-Moduls zugreifen. `range(0, 100)` erzeugt die Zahlen von 0 bis und mit 99. Die `sample`-Funktion wählt daraus dann 20 Zahlen aus und speichert diese in einer Liste.

    ```
    1 import random
    2
    3 zufallszahlen = random.sample(range(0, 100), 20)
    4 print(zufallszahlen)
    ```

    Erstellen Sie mit dem Slice-Operator zwei gleich grosse Listen. Die erste Liste beinhaltet die ersten 10 Elemente, die zweite Liste beinhaltet die letzten 10 Elemente. Bilden Sie dann von jeder Liste die Summe. Sie können die eingebaute `sum`-Funktion verwenden.