

Einführung in die computerlinguistische Programmierung mit Python

08-02: Mutability

Veränderbarkeit von Variablenwerten

Wir haben nun schon viele Datentypen kennengelernt und mit ihnen gearbeitet. Dabei ist uns immer wieder ein Begriff begegnet: **Mutability**. Der Begriff bezeichnet die Eigenschaft bestimmter Datentypen, dass Variablenwerte dieser Typen nach der Initialisierung "in-place" verändert werden können. In Python sind die Datentypen `list` (Listen), `dict` (Dictionaries), und `set` (Mengen).

```
In [1]: listel = [1,2,3]
        liste2 = listel

        listel.append(4)

        dictionary1 = {1: "hello", 2: "world"}
        dictionary2 = dictionary1

        dictionary1[1] = "hi"

        mengel = {1,2,3}
        menge2 = mengel

        mengel.add(4)

        print(liste2)
        print(dictionary2)
        print(menge2)
        print("----")
        print(listel == liste2)
        print(dictionary1 == dictionary2)
        print(mengel == menge2)
```

Obwohl wir im Code-Schnipsel oben jeweils die erste Variable der veränderbaren Datentypen nach der Initialisierung verändern, spiegelt die zweite Variable diese Veränderung wieder. Wir dürfen hierbei nicht vergessen, dass der Vergleichsoperator `==` nur die Werte von zwei Variablen vergleicht.

Diese Beobachtung basiert darauf, dass wir bei Variablenzuweisungen der Form `variable1 = variable2` für die Variable `variable2` (nur) einen **Verweis auf den Speicherort** von `variable1` vornehmen. Die beiden Variablennamen verweisen damit auf den gleichen Speicherort, d.h. ist mit Hilfe beider Variablennamen möglich auf ein und den selben Wert zuzugreifen oder diesen zu verändern.

```
In [1]: word1 = "hello"
        word2 = word1

        word1 = "world"

        print(word1)
        print(word2)

        print("----")

        number1 = 1
        number2 = number1

        number1 = 2

        print(number1)
        print(number2)
```

Bei nicht-veränderbaren Datentypen ist der Verweis auf einen Speicherort mittels zwei oder mehr Variablennamen ebenfalls möglich. Jedoch wird bei der Änderung des Werts einer nicht-veränderbaren Variable für diesen neuen Wert auch ein neuer Speicherort benutzt. Deshalb verweisen `word1` und `number1` nach der Zuweisung eines neuen Werts auf einen anderen Speicherort als `word2` und `number2` und es werden zwei verschiedene Werte ausgegeben.

Die Funktionen `id()` und der Operator `is`

Mit Hilfe der Funktion `id()` lässt sich für jede Variable deren Identität ermitteln. Die Identität ist eine Ganzzahl die jeden Variablenwert während des Programmablaufs eindeutig bezeichnet. Über die Identität einer Variable lässt sich auf den Speicherort des zugehörigen Variablenwerts schließen. (In der Python-Implementierung die mit Hilfe der Programmiersprache C vorgenommen wurde, entspricht die Identität direkt dem Speicherort der Variable.)

```
In [1]: listel = [1,2,3]
        liste2 = listel
        print("Die Identität der ersten Liste: " + str(id(listel)))
        print("Der Typ der Identität der ersten Liste: " + str(type(id(listel))))

        listel.append(4)

        id_listel = id(listel)
        id_liste2 = id(liste2)

        print(id_listel)

        print(id_listel == id_liste2)
        print(listel == liste2)
```

Da `listel` und `liste2` auf den gleichen Speicherort verweisen, ist deren Identität gleich. Dies ist **vor und nach der Wertänderung** der Fall.

Um die Identität von zwei Variablen noch einfacher vergleichen zu können können wir den Operator `is` benutzen. Er ermittelt die Identitäten der Variablen rechts und links und falls diese gleich sind, wird `True` zurückgegeben, andernfalls `False`.

```
In [1]: word1 = "hello"
        word2 = "world"

        print(word1 == word2)
        print("----")
        print(word1 is word2)
```

Unter bestimmten Bedingungen können zwei unveränderbare Variablen die gleiche Identität haben:

```
In [1]: word1 = "hello"
        word2 = "hello"

        print(word1 is word2)
```

Immutable des gleichen Typs und des gleichen Werts können aus Gründen der Optimierung auf den selben Speicherort verweisen, falls sie eine bestimmte Länge nicht überschreiten.

```
In [1]: for i in range(4):
        word1 = "oh"*2*i
        word2 = "ohoh"*i
        print("The following two words are identical: " + word1 + "\t" + word2)
        print(word1 is word2)
```

```
In [1]: import random

        for i in range(4):
            zufallszahl_stringed = str(random.randint(1,9))
            n = 1
            number1 = int(zufallszahl_stringed*n)
            number2 = int(zufallszahl_stringed*n)
            print("Der Wert von number1 und number2 ist: " + str(number1))
            print(number1 is number2 )
            print("----")
```

Unabhängige Mutables werden immer separat angelegt, auch wenn sie identische Werte haben. Zwei unabhängige Mutables haben deshalb immer auch verschiedene Identitäten.

```
In [1]: listel = [1,2,3]
        liste2 = [1,2,3]

        print(listel is liste2)
        print(listel == liste2)
```

Zusammenfassung

- Zwei Variablennamen können auf den gleichen Wert/Ort im Speicher verweisen.
- Veränderbare Datentypen können verändert werden ohne einen neuen Speicherort zugewiesen zu bekommen.
- Die Funktion `id()` gibt die Identität einer Variable zurück.
- Der Operator `is` vergleicht die Identitäten von zwei Variablen und gibt `True` zurück, falls sie identisch sind.
- Zwei unabhängig voneinander definierte veränderliche Variablen haben immer unterschiedliche Identitäten. Dies gilt bei unveränderlichen Variablen erst ab einer bestimmten Länge.

Weitere Themen dieser Woche

- Mengen 