ZUSAMMENFASSUNG GRUNDLAGEN DER PYTHON PROGRAMMIERUNG (2)

EINFÜHRUNG IN DATENTYPEN

Computer machen intern einen klaren Unterschied zwischen Ganz- und Kommazahlen.
So verwendet der Computer bei der Multiplikation von Ganzzahlen komplett andere Rechenverfahren als bei Kommazahlen.
Die Multiplikation von Kommazahlen ist hierbei wesentlich schwerer und komplexer.
Aus diesem Grund gibt es Datentypen, damit der Computer direkt weiß worum es sich beim vorliegenden Wert handelt. Dadurch kann er die spezifisch benötigte Vorgehensweise für das Durchführen von Operationen wählen.

Zum Beispiel:
```
test_variable = 30
print(test_variable + 25)
```

Der Computer überprüft dabei die Datentypen der beiden Werte und erkennt dass es sich um Ganzzahlen handelt. Dann registriert er das "+" und weiß, dass die beiden Werte miteinander addiert werden sollen. Das Ergebnis 55 wird dann über die print() Funktion ausgegeben.

Tauscht man die "25" mit "Ich bin ein String", erhält man eine Fehlermeldung. Denn man kann keine Ganzzahl mit einem String addieren.

DATENTYPEN int, float und str

In Python gibt es drei Basisdatentypen, die von Python von vornhinein zur Verfügung gestellt werden.

Datentyp int

int steht für integer. Alle Ganzzahlen werden mit diesem Datentyp gespeichert.
Es gibt keine Grenze der Größe von Zahlen die darin gespeichert werden können, man kann also beliebig viele Ganzzahlen in diesem Datentypen abspeichern.

```
int_variable1 = 100
```

Datentyp float

float kommt immer bei Gleitkommazahlen zum Einsatz. Sobald eine Zahl mit einem Dezimalpunkt vorliegt, wird diese als float gespeichert.

```
float_variable1 = 13.734
```

Das gleiche gilt für die wissenschaftliche Schreibweise die wir bereits kennengelernt haben. Hier werden Zahlen ebenfalls als float abgespeichert.

Zum Beispiel:
float_variable3 = 2

Die Zahl wird automatisch als 2.0 abgespeichert und somit als float.

Genau so kann man den Vorkommateil weglassen, wenn dieser eine 0 ist. Anstatt 0.43 schreibt man einfach .43

Ober- und Untergrenzen

Der Datentyp float hat nur eine Ober-und Untergrenze an möglichen Werten die als float gespeichert werden können.
Ist dein Wert also zu groß, wird er als +inf angezeigt. Ist die Untergrenze unterschritten als -inf. Inf steht dabei für infinitiy, also Unendlichkeit

Zum Beispiel:
float_variable4 = $10e^{99}$

Dieser Wert ist noch innerhalb der Grenzen und wird korrekt erfasst.

Hängt man noch eine 9 anh, passiert folgendes:
float_variable5 = $10e^{999}$

Man erhält als Ausgabe +inf

Schreibt man ein minus vor die Zahl, unterschreitet man die Untergrenze von float und "-inf" wird ausgegeben.

Addition und Subtraktion

Man kann zwei unendlich große Zahlen auch addieren und multiplizieren. Das Ergebnis ist dann wieder unendlich.

Bei der Subtraktion von unendlich großen Zahlen erhält man das Ergebnis NaN (not a number), was zeigt, dass das Programm etwas zu berechnen versucht, was nicht berechenbar ist. Das gleiche Problem tritt auch bei der Division auf.

Datentyp str

Einen Wert von Typ string erstellt man mit doppelt- oder einfach hochgestellten Anführungsstrichen.

DATENTYPEN HÄNGEN AN WERTEN

WICHTIG In Python hängen die Datentypen an konkreten Werten, nicht an den Variablen selbst.

Bei der Erzeugung einer Variablen, wird also lediglich intern auf den Speicherort des Wertes referenziert.

Die input() Funktion

Mit dieser Funktion kann man etwas von außen in das Programm eingeben.
Wenn man die input() Funktion im Programm abruft, stoppt das Programm und wartet auf die Eingabe des Nutzers über die Konsole. Der Nutzer kann dann einen beliebigen Wert, wie beispielsweise „Hallo“ in die Konsole eingeben.

Der Wert der eingegeben wird, kann mit Hilfe einer Variable aufgefangen werden über einen Zuweisungsoperator = .
Über die print() Funktion, kann man diese Variable wieder ausgeben lassen und bekommt als Ergebnis die Eingabe aus dem Terminal.

type() Funktion und Casting Funktionen

Möchte man zwei Werte die über die input() Funktion eingegeben wurden addieren, ist es wichtig auf die Datentypen zu achten.
Zum Beispiel:
print(value1 + value2)

Hier werden die Werte lediglich verkettet, also nacheinander ausgegeben.

Was ist der Grund dafür?

Man kann nun den Datentyp der Variablen über die type() Funktion testen.

Zum Beispiel:
print(type(value1) = <class ‘str’>

Es zeigt sich, dass der Datentyp von value1 ein string ist und nicht wie erwartet eine Zahl.

WICHTIG Die input() Funktion, speichert jeden Wert immer als String.
Also auch wenn man Zahlen eingibt, wird ein String gespeichert

Lösung

Mittels Type-Casting kann man den Datentyp eines Wertes in einen anderen umwandeln.
Im obigen Beispiel funktioniert das folgendermaßen:

int(value1) + int(value2)

Man ruft also für jede Variable die Funktion int() auf, um den Wert der Variable in einen integer umzuwandeln. Achtung: Das funktioniert natürlich nur bei Ganzzahlen.

Genau so, funktioniert es auch für float() oder str()

LISTEN IN PYTHON

Listen können mehrere Werte speichern. Man verwendet hier einen Zuweisungsoperator, gefolgt von einem eckigen Klammerpaar.
Zwischen diesem Klammerpaar kann man dann beliebige Werte speichern, die durch ein Komma getrennt sind.

Tip

Wähle den Bezeichner von Listen immer im Plural. Das signalisiert anderen, dass hier nicht nur ein, sondern mehrere Werte enthalten sind.

list ist ein Basisdatentyp in Python. Wenn man type() abfragt, erhält man ‚list' als Ausgabe
Man kann in Listen jeden beliebigen Typ von Wert speichern, beispielsweise strings oder Ganzzahlen
Außerdem kann man in einer Liste auch andere Listen speichern

Zugriff auf Listen

Man kann gezielt auf einzelne Elemente in einer Liste zugreifen, indem man den Index des gewünschten Elements in die eckige Klammer schreibt.
Zum Beispiel:

print(list[0]) -> Hier erhält man den ersten Wert der Liste
print(list[1]) -> Hier erhält man den zweiten Wert der Liste

Wählt man einen höheren Index als den Index des letzten Elements, erhält man die Fehlermeldung „list index out of range"

Um einfach auf die letzten Elemente einer Liste zuzugreifen, kann man mit negativen Indexen arbeiten.
Zum Beispiel:
print(list[-1]) -> Hier erhält man den letzten Wert der Liste
print(list[-2]) -> Hier erhält man den vorletzten Wert der Liste

Zuweisung in Listen

Man kann über den Index nicht nur Werte ausgeben, sondern auch Werte ersetzten.

Zum Beispiel:
names[0] = „Fritz“

Damit weist man dem ersten Element der Liste den Wert „Fritz“ zu.

Range in Listen

Mit den eckigen Klammern kann man auf eine spezielle Range in der Liste zugreifen und sich so ausgewählte Elemente ausgeben lassen.

Zum Beispiel
print(names[1:4])

Der Index vor dem Punkt bezieht sich auf den Startpunkt, der Index nach dem Punkt auf den Endpunkt. Es werden also der zweite, dritte und vierte Wert der names Liste ausgegeben.

WICHTIG Das Element am Startindex wird immer inkludiert, das Element am Endindex wird immer exkludiert.

Wenn man am 0ten Element starten möchte, kann man auch eine verkürzte Schreibweise verwenden. Hier wird die 0 nicht in die Klammer geschrieben sondern:
names[:4]

Umgekehrt gilt das gleiche, wenn man alle Werte ab einem bestimmten Index bis zum Ende der Liste haben möchte. Zum Beispiel:
names[3:]

Wichtige Basisfunktionalitäten von Listen

Es gibt noch weitere Funktionen die hilfreich im Umgang mit Listen sind.

len()

Mit dieser Funktion wird die Anzahl der Elemente einer Liste in Form einer Ganzzahl ausgegeben.

append()

Mit dieser Funktion kann man ein neues Element an eine bestehende Liste anhängen. Hier schreibt man zuerst den Listennamen, auf den die Funktion angewendet werden soll, gefolgt von einem Punkt und append().

Zum Beispiel:
names.append(„Carsten“)

insert()

Hiermit kann man einen neuen Wert in eine bestehende Liste an einer bestimmten Stelle einfügen.

Zum Beispiel:
names.insert(1, „Karl“)

Hier fügen man den Wert „Karl“ an die zweite Stelle der Liste ein. Alle nachfolgenden Elemente werden entsprechend um eine Stelle nach hinten verschoben.

pop()
Hiermit kann der letzte Wert einer Liste entfernt werden.

remove()
Hiermit kann man einen bestimmten Wert an einer bestimmten Stelle entfernen.

Zum Beispiel:
names.remove(„Carsten“)