# Array

Arrays sind spezielle Variablen, die nicht nur einen Wert, sondern eine Vielzahl von Werten speichern können. Dafür wird der Wert in ein bestimmtes Feld des Arrays abgelegt. Um auf den Inhalt eines Feldes (das Element) zugreifen zu können, werden Indizes verwendet. Es ist aber nicht nur möglich Werte (Zahlen, Text usw.) in Arrayelemente zu legen. Auch Objekte können in einem Array gespeichert werden.

Arrays sind vergleichbar mit Regalen. Wir befüllen die einzelnen Fächer, die in einer bestimmten Weise zusammengehören (bspw. Objekte der Klasse Konto).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 |
| Kontonr. | 1001287 | 1001288 | 1001265 | 1002377 | 1004389 | 1003298 |
| Inhaber | Peter  Müller | Steffen  Amsel | Jana  Licht | Nina  Franz | Horst Schulz | Sören  Jansen |
| Kontostand | 320.45 | 322.12 | 7643.57 | 554.99 | 98.99 | 101.21 |

Im oben dargestellten Array sind sechs Konten gespeichert. Auffällig ist hierbei, dass der Index bei 0 anfängt und nicht bei 1. Auf den Vergleich mit dem Regal bezogen könnte man sagen, dass im Fach 1 das Objekt des Inhabers Steffen Amsel mit der Kontonummer 1002398 und einem Kontostand von 322,12 € liegt.

## Deklaration und Initialisierung in Python

In Python werden Arrays mit Hilfe von Listen umgesetzt. Im Folgenden wird aber weiterhin von Arrays gesprochen.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | # Hauptprogramm  ko1 = Konto(1001287, "Peter Müller", 320.45)  …  ko6 = Konto(1003298, "Sören Jansen", 101.21)  kontoliste = [] #Erzeugen eines leeren Array  #Alternativ: kontoliste = [ko1, ko2, ko3, ko4, ko5, ko6]  kontoliste.append(ko1)  …  kontoliste.append(ko6) |

Die Objekte der Klasse *Konto* werden wie gewohnt erzeugt. Im Beispiel oben wird zuerst ein Array ohne Inhalt erzeugt (vgl. Zeile 6). Alternativ hätte der Array auch deklariert und initialisiert werden können (vgl. Zeile 7). Mit der Methode append()(vgl. Zeile 9) kann dem Array ein Element am Ende der Liste hinzugefügt werden. Neben der Methode append() gibt es noch [weitere Methoden](https://www.w3schools.com/python/python_ref_list.asp), die auf einem Array ausgeführt werden können.

## Zugriff auf Arrayelemente mit einem Index

Auf die einzelnen Arrayelemente kann mit einem Index oder einer Schleife zugegriffen werden. Möchte man nur auf ein bestimmtes Element zugreifen, bietet sich der Zugriff mit einem Index an. Dabei gilt es zu beachten, dass auf das erste Element in einem Array mit dem Index 0 zugegriffen wird. Folglich wird auf das letzte Element (n-te Element) in einem Array mit dem Index n-1 zugegriffen.

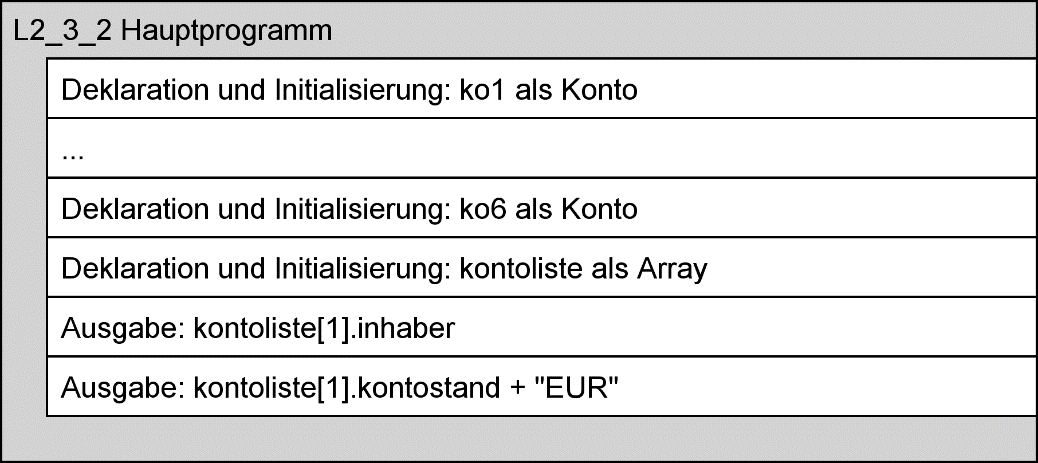
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | class Konto:  def \_\_init\_\_(self, p\_kontonummer, p\_inhaber, p\_kontostand):  self.kontonummer = int(p\_kontonummer)  self.inhaber = p\_inhaber  self.kontostand = float(p\_kontostand)  #Hauptprogramm  ko1 = Konto(1001287, "Peter Müller", 320.45)  ko2 = Konto(1001288, "Steffen Amsel", 322.12)  ko3 = Konto(1001265, "Jana Licht", 7643.57)  ko4 = Konto(1002377, "Nina Franz", 554.99)  ko5 = Konto(1004389, "Horst Schulz", 98.99)  ko6 = Konto(1003298, "Sören Jansen", 101.21)  kontoliste = []  kontoliste.append(ko1)  kontoliste.append(ko2)  …  kontoliste.append(ko6)  print(kontoliste[1].inhaber)  print(kontoliste[1].kontostand, "EUR") |

Nachdem alle Konten erzeugt und im Array *kontoliste* abgespeichert wurden, erfolgt der Zugriff auf ein Arrayelement, indem der Index in die eckigen Klammern geschrieben wird. Wie schon gewohnt kann dann mit einem Punkt und dem Attributnamen auf den Attributwert zugegriffen werden, um ihn dann bspw. auszugeben.

**Ausgabe des Programms:**

|  |
| --- |
| Steffen Amsel  322.12 EUR |

**Struktogramm:**



## Vergleich von Arrayelementen

Oft möchte man die einzelnen Arrayelemente miteinander vergleichen (siehe Zeile 15 ff.), um davon abhängig eine Aktion auszuführen.

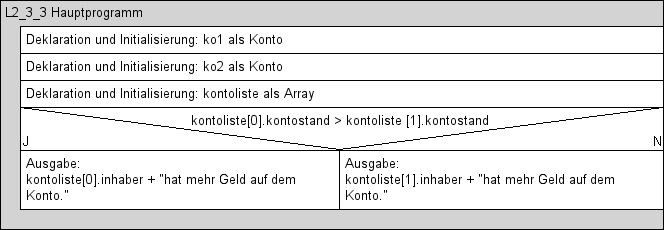
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | class Konto:  def \_\_init\_\_(self, p\_kontonummer, p\_inhaber, p\_kontostand):  self.kontonummer = int(p\_kontonummer)  self.inhaber = p\_inhaber  self.kontostand = float(p\_kontostand)  #Hauptprogramm  ko1 = Konto(1001287, "Peter Müller", 320.45)  ko2 = Konto(1001288, "Steffen Amsel", 322.12)  kontoliste = []  kontoliste.append(ko1)  kontoliste.append(ko2)  if kontoliste[0].kontostand > kontoliste[1].kontostand:  print(kontoliste[0].inhaber, "hat mehr Geld auf dem Konto.")  else:  print(kontoliste[1].inhaber, "hat mehr Geld auf dem Konto.") |

Durch den Index wird auf ein bestimmtes Arrayelement zugegriffen. Über einen Punkt kann dann auf das entsprechende Attribut oder auch Methode zugegriffen werden.

**Ausgabe des Programms:**

|  |
| --- |
| Steffen Amsel hat mehr Geld auf dem Konto. |

**Struktogramm:**



## Zugriff auf Arrayelemente mit einer Schleife

Für die Ausgabe aller Kontoinhaber bei Tausenden von Konten ist ein Zugriff mit einer Schleife komfortabler und praktikabler.

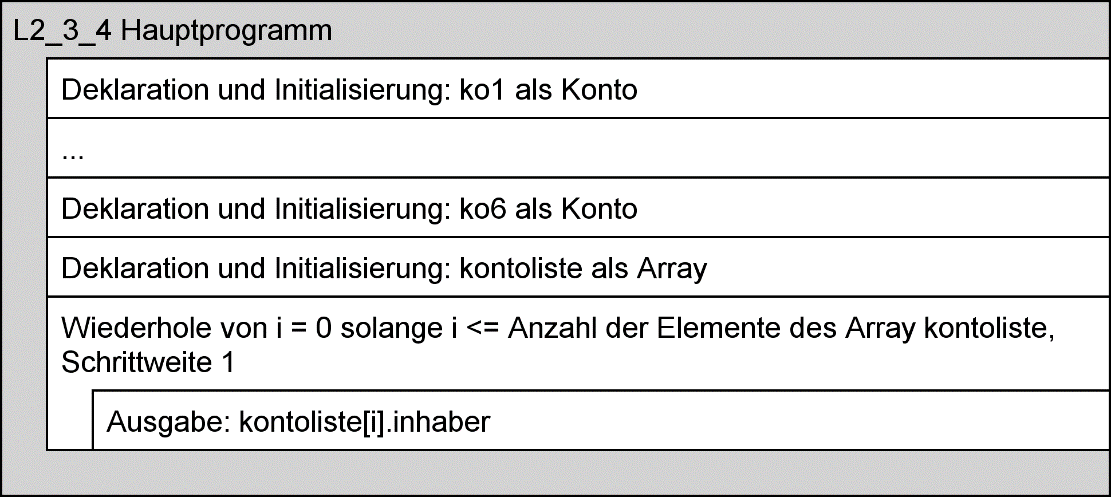
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | class Konto:  def \_\_init\_\_(self, p\_kontonummer, p\_inhaber, p\_kontostand):  self.kontonummer = int(p\_kontonummer)  self.inhaber = p\_inhaber  self.kontostand = float(p\_kontostand)  # Hauptprogramm  ko1 = Konto(1001287, "Peter Müller", 320.45)  ko2 = Konto(1001288, "Steffen Amsel", 322.12)  ko3 = Konto(1001265, "Jana Licht", 7643.57)  ko4 = Konto(1002377, "Nina Franz", 554.99)  ko5 = Konto(1004389, "Horst Schulz", 98.99)  ko6 = Konto(1003298, "Sören Jansen", 101.21)  kontoliste = [ko1, ko2, ko3, ko4, ko5, ko6]  for i in kontoliste:  print(i.inhaber) |

Die for-Schleife in Zeile 17 greift nacheinander auf jedes Element des Array kontoliste zu. In der Schleife kann auf das gerade ausgewählte Element des Arrays (hier das entsprechende Objekt der Klasse Konto) mit i zugegriffen werden (vgl. Zeile 18). Im ersten Schleifendurchlauf ist i.inhaber folglich äquivalent mit ko1.inhaber, weil das erste Element (ko1) des Array ausgewählt wurde.

**Ausgabe des Programms:**

|  |
| --- |
| Peter Müller  Steffen Amsel  Jana Licht  Nina Franz  Horst Schulz  Sören Jansen |

**Struktogramm:**



## Anzahl der Elemente eines Arrays

Bei der Programmierung ist es oft wichtig die Anzahl der Arrayelemente (Feldelemente) zu kennen. In Python kann die Anzahl der Arrayelemente mit der Funktion len() ermittelt werden.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | class Konto:  def \_\_init\_\_(self, p\_kontonummer, p\_inhaber, p\_kontostand):  self.kontonummer = int(p\_kontonummer)  self.inhaber = p\_inhaber  self.kontostand = float(p\_kontostand)  #Hauptprogramm  ko1 = Konto(1001287, "Peter Müller", 320.45)  ko2 = Konto(1001288, "Steffen Amsel", 322.12)  ko3 = Konto(1001265, "Jana Licht", 7643.57)  ko4 = Konto(1002377, "Nina Franz", 554.99)  ko5 = Konto(1004389, "Horst Schulz", 98.99)  ko6 = Konto(1003298, "Sören Jansen", 101.21)  kontoliste = [ko1, ko2, ko3, ko4, ko5, ko6]  print("Anzahl der Konten:", len(kontoliste)) |

**Ausgabe des Programms:**

|  |
| --- |
| Anzahl der Konten: 6 |

**Struktogramm:**

