Woche 04: Programmierung - Module und Funktionen am Beispiel von NumPy (Teil 2)

Skript

Erarbeitet von
Ludmila Himmelspach

Lernziele

- Auf die Elemente der *ndarrays* zugreifen können
- Suchen nach Elementen mit bestimmten Eigenschaften in *ndarrays*

Inhalt

Zugriff auf die Elemente in einem *ndarray*

Wir schauen uns an, wie man auf die Elemente eines *ndarrays* zugreifen kann. Da der Aufbau eines *ndarrays* einer Liste sehr ähnlich ist, erfolgt auch der Zugriff auf seine Elemente in ähnlicher Weise. Du kannst auf die Elemente in einem *ndarray* über deren Indexe zugreifen. Zum Beispiel, wenn du wissen möchtest, welches Element in der zweiten Zeile und der dritten Spalte steht, kannst du die beiden Indexe durch ein Komma getrennt in

Projekt „KI für alle"
HeiCAD – Heine Center for Artificial Intelligence and Data Science

der eckigen Klammer hinter dem Array Namen angeben. Beachte dabei, dass die Indexnummerierung in *ndarrays* genau wie in den Listen mit null anfängt.

```
print("Element in der zweiten Zeile und dritten Spalte des "
      "Arrays patienten_daten:", patienten_daten[1,2])
```

In der Ausgabe sehen wir den Wert *56.3*, was dem Gewicht des zweiten Patienten entspricht.

Du kannst ganze Zeilen oder Spalten eines *ndarrays* ausgeben oder in einem weiteren *ndarray* speichern. Zum Beispiel kannst du alle Daten des vierten Patienten in einem neuen Array speichern. Dafür gibst du den Namen des neuen Arrays, also *patient_4*, an. Nach dem Zuweisungsoperator wird der Name des Ursprungsarrays, also *patienten_daten,* angegeben. In der eckigen Klammer musst du den Index der vierten Zeile angeben, also 3. Um die Werte aller Spalten im neuen Array zu speichern, muss man an zweiter Stelle – man sagt auch „in der zweiten Dimension“ einen Doppelpunkt angeben. Optional kannst du auch 0:4 schreiben.

```
patient_4 = patienten_daten[3,:]

print("Daten des vierten Patienten aus dem Array patienten_daten:",
      patient_4)
```

In der Ausgabe der *print()*-Anweisung sehen wir, dass im *ndarray patient_4* alle Daten des vierten Patienten gespeichert wurden.

Auf ähnliche Weise kannst du die Elemente einer Spalte in einem neuen *ndarray* speichern. Hier musst du in der ersten Dimension einen Doppelpunkt angeben. In der zweiten Dimension gibst du den Index der entsprechenden Spalte an.

```
patienten_alter = patienten_daten[:, 0]

print("Alter aller Patienten des Array patienten_daten:",
      patienten_alter)
```

Wenn du Werte eines Elements in einem *ndarrays* ändern möchtest, kannst du das Element über seinen Index ansprechen und ihm einen neuen Wert zuweisen. Wir ändern jetzt die Größe des zweiten Patienten. Dafür geben wir in der eckigen Klammer in der ersten Dimension den Index des zweiten Patienten, also eine Eins, an. Da die Körpergröße in der zweiten Spalte gespeichert wird, geben wir in der zweiten Dimension eine Eins an. Nach dem Zuweisungsoperator geben wir einen neuen Wert zum Beispiel *1.92* an.

```
print("Array patienten_daten vor der Änderung:")
print(patienten_daten)
print()

patienten_daten[1, 1] = 1.92

print("Array patienten_daten nach der Änderung:")
print(patienten_daten)
```

Wenn wir das gesamte *ndarray patienten_daten* ausgeben, sehen wir, dass in der zweiten Zeile und in der zweiten Spalte ein neuer Wert steht.

Suche in *ndarrays*

In *ndarrays* kann man mit der *where()*-Funktion die Position eines Elements mit bestimmten Eigenschaften ausgeben lassen. Dabei wird das Ergebnis in einem Array gespeichert.

Quelle [1]

Wenn du zum Beispiel wissen möchtest, welche Patienten älter als 35 Jahre sind, kannst du dies mit der folgenden Anweisung herausfinden: Zuerst schreiben wir den Namen der Funktion, also *where()*, des NumPy Moduls. In der runden Klammer geben wir an, dass uns nur die Elemente der ersten Spalte interessieren, wo das Alter der Patienten erfasst ist. Danach legen wir die Bedingung, also größer als 35, fest.

```
print("Indizes der Patienten, die älter als 35 Jahre sind:",
      np.where(patienten_daten[:,0] > 35))
```

In der Ausgabe sehen wir, dass der erste und der dritte Patient älter als 35 Jahre sind.

Wenn wir wissen wollen, welche Patienten älter als oder genau 35 Jahre alt sind, müssen wir in der Bedingung „>=“ angeben.

```
print("Indizes der Patienten, die älter oder genau 35 Jahre sind:",
      np.where(patienten_daten[:,0] >= 35))
```

In der Ausgabe sehen wir, dass es auf die ersten drei Patienten zutrifft.

Wenn wir die Position des Patienten herausfinden wollen, der genau 35 Jahre alt ist, müssen wir das doppelte Gleichheitszeichen als Vergleichsoperator angeben, weil das einfache Gleichheitszeichen als Zuweisungsoperator in Python fungiert und an dieser Stelle zu einer Fehlermeldung führen würde. Also schreiben wir unsere Anweisung wie folgt auf:

```
print("Indizes der Patienten, die genau 35 Jahre sind:",
      np.where(patienten_daten[:,0] == 35))
```

In der Ausgabe sehen wir, dass nur der zweite Patient genau 35 Jahre alt ist.

Take-Home Message

In diesem Video hast du gelernt, dass man in Python nicht alles selbst programmieren muss, sondern Module nutzen kann. Anhand des NumPy-Moduls hast du gelernt, wie du Module in deine Programme importierst. Weiterhin hast du die Datenstruktur *ndarray*

kennengelernt, in der oft Werte aus Datentabellen gespeichert werden. Außerdem weißt du jetzt, wie man auf die einzelnen Elemente zugreifen, diese ändern und wie man die Position der Elemente mit bestimmten Eigenschaften in einem *ndarray* bestimmen kann.

Quellen

Quelle [1] NumPy (2022). Sorting, searching, and counting. In *NumPy Reference, Release 1.23*. https://numpy.org/doc/1.23/reference/generated/numpy.where.html

Weiterführendes Material

Schmitt, S. (2021). *Python Kompendium: Professionell Python Programmieren lernen.* BMU Media Verlag

Barry, P. (2017). *Python von Kopf bis Fuß.* O'Reilly

Disclaimer

Transkript zu dem Video „Woche 04: Programmierung - Module und Funktionen am Beispiel von NumPy (Teil 2)", Ludmila Himmelspach.