Woche 02: Programmierung – Variablen, Datentypen und Operationen in Jupyter Notebook

Skript

Erarbeitet von
Ludmila Himmelspach

Lernziele

- Eine Variable in Python anlegen, ihr einen Wert zuweisen und Variablenwerte ausgeben
- Datentypen der Variablen abfragen und ausgeben
- Datentypen der Variablenwerte umwandeln
- Operationen auf Zahlen und Zeichenkette ausführen

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

Inhalt

Variablen

Du weißt jetzt ja schon, was Variablen sind. Damit man die Bedeutung der Variablen beim Lesen des Programmcodes leichter verstehen kann, gibt man ihnen aussagekräftige Namen. Wenn man zum Beispiel ein Programm zur Zinsberechnung schreibt, geben die Variablennamen wie *kapital* und *zinssatz* mehr Auskunft über die Nutzung des Wertes als *x* und *y*.

In Python können Variablennamen aus Buchstaben, Zahlen, Sonderzeichen, mathematischen Symbolen, Umlauten und nicht-lateinischen Buchstaben bestehen. Die einzige Einschränkung ist, dass am Anfang des Variablennamens keine Zahl stehen darf. In der Praxis haben sich aber Variablennamen etabliert, die ausschließlich aus kleinen Buchstaben, Zahlen und Unterstrichen bestehen. Wenn man aber Großbuchstaben in Variablennamen verwenden möchte, muss man aufpassen, da in Python zwischen Groß- und Kleinschreibung unterschieden wird. So sind *variable* und *Variable* zwei verschiedene Variablen in Python.

Quelle [1]

Variablenzuweisung

Nachdem man sich auf einen Variablennamen festgelegt hat, muss man ihr einen Wert zuweisen. In Python werden die Werte mit einem Gleichheitszeichen („=“) den Variablen zugewiesen. So fungiert das Gleichheitszeichen unter anderem als Zuweisungsoperator in Python. Die Variablenwerte können zum Beispiel Zahlen, Buchstaben, Wörter, Sätze oder auch Bilder sein.

Wir fangen mit Zahlen und Texten an. Zuerst erzeugen wir eine Variable mit dem Namen *text*. Dieser weisen wir einen Wert „Taschenrechner“ zu. Wie wir schon gesehen haben, werden in Python Texte in Anführungszeichen geschrieben. So schreiben wir „Taschenrechner“ in Anführungszeichen. Unsere zweite Variable nennen wir *zahl*. Dieser weisen wir den Wert 100 zu.

```
# In Python werden Texte in Anführungszeichen geschrieben
text = "Taschenrechner"

zahl = 100
```

Um den Wert einer Variable in Python ausgeben zu lassen, kann man den *print()*-Befehl verwenden. Dafür setzt man einfach den Variablennamen in die Klammern innerhalb des Befehls *print().*

```
print(text)

print(zahl)
```

Hier muss man darauf achten, keine Anführungszeichen zu verwenden, weil sonst der Variablenname und nicht deren Wert ausgegeben wird.

```
print("zahl")
```

Wenn man Werte mehrerer Variablen im Programm auf einmal ausgibt, ist es praktisch, in der Ausgabe vor jedem Variablenwert den entsprechenden Variablennamen mit auszugeben. Das erreicht man, indem man in den Klammern des *print()*-Befehls in Anführungszeichen den Variablennamen eventuell mit Begleittext eintippt, dann ein Komma setzt und dann den Variablennamen eintippt. Alles, was innerhalb der Anführungszeichen steht, wird als eine Zeichenkette interpretiert und genauso in der Ausgabe angezeigt. Auf diese Weise können wir die Werte unserer Variablen *text* und *zahl* ausgeben.

```
print("Wert der Variablen text: ", text)

print("Wert der Variablen zahl: ", zahl)
```

Datentypen

Neben den Variablenwerten kann man mit dem *print()*-Befehl auch die Variablentypen ausgeben. Dafür wird der Befehl *type()* verwendet. Wie bereits erwähnt, in Python braucht man beim Anlegen einer Variablen ihren Datentyp nicht extra anzugeben. Dieser wird durch den zugewiesenen Wert bestimmt. Aber wenn wir Operationen auf den Variablen ausführen möchten, spielt der Datentyp eine wichtige Rolle, wie wir später in diesem Video noch sehen werden.

Um den Datentyp unserer Variable *zahl* auszugeben, müssen wir die Befehle *print()* und *type()* folgendermaßen verschachteln. Da wir den Typ der Variable *zahl* bestimmen wollen, tippen wir zuerst *type* und in Klammern den Variablennamen, also *zahl,* ein. Da wir den Variablentyp ausgeben möchten, umschließen wir den berechneten Variablentyp mit dem Befehl *print()*. Wir können noch den Ausgabetext 'Datentyp der Variable zahl' der *print*-Anweisung hinzufügen.

```
print("Datentyp der Variable zahl: ", type(zahl))
```

In der Ausgabe steht *„class: int“*. *int* ist eine Abkürzung für *integer* und bedeutet „ganze Zahl“. Also ist der Wert der Variable *zahl* eine ganze Zahl.

Neben der ganzen Zahl gibt es noch Dezimal- oder Fließkommazahl als Datentyp in Python. Dieser wird als *float* abgekürzt und bedeutet *„floating-point number“.* Wir erzeugen eine Variable *zahl_2* und weisen ihr den Wert 2.5 zu. Hier ist zu beachten, dass angelehnt an die

englische Schreibweise ein Dezimalpunkt statt eines Kommas verwendet wird. Wir geben den Datentyp der Variablen *zahl_2* aus. Dafür tippen wir die Anweisung wie oben ein und führen das Programm aus.

```
zahl_2 = 2.5

print("Datentyp der Variable zahl_2: ", type(zahl_2))
```

Wie wir in der Ausgabe sehen können, ist der Datentyp der Variable *zahl_2* ein *float*.

Ein weiterer gängiger Datentyp in Python ist *String* bzw. Zeichenkette. Zeichenketten werden mit *„str“* in Python abgekürzt und bestehen aus einer Abfolge von Buchstaben, Ziffern, Sonderzeichen und können sich zu Wörtern, Sätzen und ganzen Texten zusammensetzen. Zum Beispiel ist der Datentyp unserer Variable *text* ein String. Das können wir schnell mit dem *print()*-Befehl prüfen. Dazu geben wir folgende Anweisung ein.

```
print("Datentyp der Variable text: ", type(text))
```

Wie ich bereits gesagt habe, können Zeichenketten aus Ziffern bestehen, auch eine einzelne Ziffer kann als eine Zeichenkette bzw. als ein String gespeichert werden. Wir erzeugen eine weitere Variable *text_2*. Darin speichern wir die Zahl „100“ als Zeichenkette. Damit die Zahl „100“ als ein String und nicht als ein Integer gespeichert wird, müssen wir die Zahl „100“ wie alle Zeichenketten in Anführungszeichen schreiben. Jetzt geben wir den Datentyp der Variable *text_2* aus.

```
text_2 = "100"

print("Wert der Variablen text_2: ", text_2)

print("Datentyp der Variable text_2: ", type(text_2))
```

Wie wir in der Ausgabe sehen können, ist der Datentyp der Variable *text_2* anders als der Datentyp der Variable *zahl* ein String und kein Integer. Welchen Unterschied das beim Anwenden von Operationen auf diesen beiden Variablen ausmacht, sehen wir später.

Datentypumwandlung

Manchmal möchte man im Programmverlauf den Datentyp eines Variablenwertes (explizit) ändern. Dazu bietet Python den Befehl *int()* für die Integerumwandlung, den Befehl *float()* für die Fließkommazahlumwandlung und den Befehl *str()* für die Umwandlung in eine Zeichenkette an. So können wir zum Beispiel den Wert unserer Variable *text_2*, was bis jetzt ein String war, in eine Fließkommazahl umwandeln. Dazu geben wir Folgendes ein: *text_2 = float(text_2)*. Jetzt können wir den Datentyp von *text_2* überprüfen.

```
# Die String-Variable text_2 wird in eine Fließkommazahl umgewandelt.
text_2 = float(text_2)
```

```
# Gebe den Datentyp und den Variablenwert aus.
print("Datentyp der Variable text_2: ", type(text_2))
print("Wert der Variablen text_2: ", text_2)
```

In der Ausgabe sieht man, dass der Wert der Variable *text_2* nach der Umwandlung eine Fließkommazahl ist.

Operationen auf Zahlen

Die Werte der Variablen können sich im Laufe des Programms durch verschiedene Operationen ändern. Dies kann zum Beispiel eine Neuzuweisung sein. So können wir zum Beispiel unserer Variablen *zahl* einen neuen Wert von 10 zuweisen.

```
zahl = 10

print("Wert der Variablen zahl: ", zahl)
```

Die Variablenwerte können sich auch durch Rechenoperationen ändern. Python verfügt über eine Vielzahl von arithmetischen Operatoren, von denen wir die wichtigsten gleich ausprobieren werden.

Der erste Operator ist der *Additionsoperator*, mit dem man die Summe von Zahlen berechnet. Dieser Operator wird in Python mit einem Plus-Zeichen dargestellt. Mit dem Additionsoperator können wir unserer Variablen *zahl* die Summe zweier Zahlen, zum Beispiel 10 plus 10, zuweisen.

```
# Weise der Variable zahl die Summe zweier Zahlen zu.
zahl = 10 + 10

print("Wert der Variablen zahl: ", zahl)
```

Wir können aber zu dem Wert einer Variablen eine Zahl hinzuaddieren. Das geht, indem wir einen Operanden durch den Namen der entsprechenden Variablen ersetzen. Also tippen wir *zahl = zahl + 10* ein. Aus mathematischer Sicht ist dieser Ausdruck nicht korrekt. Aber beim Gleichheitszeichen handelt es sich um einen Zuweisungsoperator in Python, deswegen ist dieser Ausdruck wie folgt zu verstehen: Der Variable *zahl* weisen wir einen neuen Wert zu, der sich aus der Summe des alten Variablenwerts und der Zahl 10 zusammensetzt. Der alte Variablenwert war 20, deswegen sollte der neue Variablenwert 30 sein. Das können wir mit dem *print()*-Befehl einfach überprüfen.

```
# Addiere 10 zu dem Wert der Variablen zahl und
# speichere das Ergebnis in zahl ab.
zahl = zahl + 10

print("Wert der Variablen zahl: ", zahl)
```

Natürlich kann man in Python auch subtrahieren, nämlich mit dem *Subtraktionsoperator* -, und multiplizieren, nämlich mit dem *Multiplikationsoperator* *.

Der *Divisionsoperator* wird in Python mit einem Schrägstrich dargestellt. Wir teilen den Wert der Variable *zahl* durch 2 und speichern das Ergebnis in einer neuen Variable *zahl_2* ab.

```
# Teile den Wert der Variable zahl durch 2 und
# speichere das Ergebnis in einer neuen Variable zahl_2 ab.
zahl_2 = zahl / 2

print("Wert der Variablen zahl: ", zahl)
print("Wert der Variablen zahl_2: ", zahl_2)
```

Der Wert der Variable *zahl* hat sich nicht geändert, weil wir die Variable *zahl* für die Berechnung zwar benutzt, dieser aber keinen neuen Wert zugewiesen haben. Der Wert der Variable *zahl_2* ist 50.0. Das bedeutet, dass es sich um eine Fließkommazahl handelt. In Python ist das Ergebnis einer Division immer eine Fließkommazahl, selbst wenn im Ergebnis eigentlich eine ganze Zahl rauskommt. Wenn man möchte, dass im Ergebnis eine ganze Zahl steht, so muss man entweder zusätzlich eine Typumwandlung durchführen oder eine Ganzzahldivision statt einfacher Division durchführen.

Operationen auf Zeichenketten

Neben den arithmetischen Operatoren verfügt Python über eine Vielzahl von Operatoren auf Zeichenketten. Dabei haben einige Operatoren die gleiche Darstellung, produzieren aber unterschiedliche Ergebnisse auf Zahlen und Zeichenketten. Wir werden die wichtigsten von ihnen kennenlernen.

Wenn wir den „+“-Operator auf Zeichenketten anwenden, heißt dieser *Verkettungsoperator* und bewirkt die Hintereinanderschaltung bzw. Verkettung von Zeichenketten.

Wir erzeugen zwei Zeichenketten *text_1 = „hallo“* und *text_2 = „zusammen“.* Wenn wir auf beiden Zeichenketten den Verkettungsoperator benutzen, erhalten wir in der Ausgabe „hallozusammen“.

```
text_1 = "Hallo"
text_2 = "zusammen"
text_3 = text_1 + text_2
print(text_3)
```

Wenn man zwischen den Worten ein Leerzeichen und am Ende ein Ausrufezeichen schreiben möchte, kann man diese bei der Verknüpfung zwischen den beiden Variablen bzw. am Ende schalten.

```
# Hier überschreiben wir den Inhalt der Variable text_3.
text_3 = text_1 + " " + text_2 + "!"
print(text_3)
```

Die Zahlen, die als Zeichenketten gespeichert werden, werden wie Zeichenketten miteinander verknüpft. Also werden diese nicht addiert, sondern verkettet. Wir erzeugen

wieder zwei Zeichenketten *text_4 = „20“* und *text_5 = „23“*. Wenn wir die beiden Zeichenketten mit dem Verkettungsoperator miteinander verknüpfen, also

```
text_4 = "20"
text_5 = "23"
text_6 = text_4 + text_5
print(text_6)
```

erhalten wir in der Ausgabe 2023. Hätten wir die beiden Zahlen als Integer gespeichert, hätte der +-Operator die Summe der beiden Zahlen, also 43, berechnet.

Und was passiert, wenn man eine Zeichenkette, die eine Zahl enthält, mit einer Ganzzahl durch den Plusoperator verknüpft? Fungiert der Plusoperator dann als ein Additions- oder als ein Verkettungsoperator? Das erfährst du, indem du es ausprobierst. Man lernt Programmieren am besten, wenn man viel ausprobiert. Selbst wenn du versuchst, Befehle auszuführen, die zu Fehlermeldungen führen, kannst du den fehlerhaften Code entweder wieder löschen oder auskommentieren, d. h. du markierst den Codeabschnitt als Kommentar.

Unsere Variable *text_6* enthält die Zahl 2023, die als Zeichenkette gespeichert ist. Diese verknüpfen wir mit 1 durch den Plusoperator und geben das Ergebnis aus.

```
# text_x = text_6 + 1
# print(text_x)
```

Es kommt zu einer Fehlermeldung, die besagt, dass man eine Zeichenkette nur mit einer Zeichenkette verknüpfen kann.

Du kannst selbst ausprobieren, was passiert, wenn du die Zeichenkette mit der Zahl tauschst. Vergiss danach nicht, den fehlerhaften Code zu löschen oder auszukommentieren, damit du ohne Probleme weiterarbeiten kannst.

Take-Home Message

In diesem Video hast du gelernt, wie man in Python Variablen anlegen, ihre Werte und ihre Datentypen ausgeben kann. Außerdem kannst du jetzt die Datentypen der Variablenwerte selbst ändern. Des Weiteren hast du gelernt, wie man unterschiedliche Operationen auf Zahlen und Zeichenketten ausführt. Eigentlich brauchst du keinen Taschenrechner mehr, weil du Rechenoperationen ab jetzt auch in Python ausführen kannst.

Quellen

Quelle [1] Van Rossum, G., & Drake, F. L. (2009). *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace.

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 02: Programmierung – Variablen, Datentypen und Operationen in Jupyter Notebook", Ludmila Himmelspach.