



15

Mengen



In der Mathematik ist die Kunst, eine Frage zu stellen, höher zu bewerten als die Kunst, sie zu lösen.

Georg Cantor

Die *Menge*¹ ist im gewissen Sinne die grundlegendste Struktur der ganzen Mathematik, obwohl sie in ihrer jetzigen Form erst seit dem Ende des 19. Jahrhunderts untersucht wird. Mengen spielen in der Erforschung der Grundlagen der Mathematik eine herausragende Rolle, da sich alle anderen Objekte, mit denen sich die Mathematik beschäftigt, durch Mengen repräsentieren lassen. Insbesondere gilt das auch für Dinge wie *Zahlen* oder *Funktionen*.

Nun wollen wir uns nicht mit Grundlagenforschung beschäftigen und man könnte daher fragen, ob wir uns überhaupt um Mengen kümmern sollten. Es gibt jedoch mehrere Gründe, dies zu tun:

- Die *Mengenschreibweise* ist ein wesentlicher Teil der mathematischen Fachsprache. Wenn man sie nicht beherrscht, wird man große Teile der Fachliteratur, auch in der Informatik, nicht verstehen können.
- Gerade in der theoretischen Informatik wird die Mengenlehre in fast allen Teilbereichen regelmäßig verwendet.
- Mengen (englisch *sets*) können beim Programmieren eine nützliche Datenstruktur sein.
- Einige der Entdeckungen von Cantor sind so überraschend und gleichzeitig so faszinierend, dass sie für einen Menschen im 21. Jahrhundert eigentlich zur Allgemeinbildung gehören sollten.

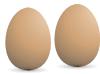
¹Georg Cantor begründete die Mengenlehre und revolutionierte mit seinen Untersuchungen zur Unendlichkeit die Mathematik. Zu seinen Lebzeiten wurde er von einigen einflussreichen Kollegen heftig angefeindet. Erst nach seinem Tod begann man zu erkennen, dass er einer der wichtigsten Mathematiker seiner Zeit war.





PYTHON kann auch mit Mengen arbeiten, allerdings gibt es einige wesentliche Unterschiede zu den Mengen der Mathematik.² Wir werden trotzdem beide Mengenbegriffe parallel entwickeln und ggf. auf die Unterschiede hinweisen.

Zudem sei der guten Ordnung halber darauf hingewiesen, dass wir hier sogenannte *naive Mengenlehre* betreiben werden. Damit bezeichnet man in der Mathematik einen Ansatz, der anschaulich in die Materie einführt und auf formale Begriffsbildungen verzichtet. Im Gegensatz dazu steht die *axiomatische Mengenlehre*, die auf strengen logischen Grundlagen beruht, aber für Nicht-Mathematiker schwer zu verstehen ist.³



NACH DER URSPRÜNGLICHEN DEFINITION von Cantor ist eine *Menge* einfach eine „Ansammlung von Objekten“. Grundsätzlich kann das alles sein, was man mit Worten beschreiben oder sich vorstellen kann; z.B. die Menge der Eier in meinem Kühlschrank oder die Menge der Mitgliedsstaaten der UNO. In der Mathematik geht es aber natürlich fast immer um *abstrakte* Dinge (siehe Seite 16) wie etwa Zahlen.

Eine Menge kann z.B. so aussehen:

$$A = \{1, 2, 3, 42\}$$

Hierbei handelt es sich um eine Ansammlung von vier Objekten. Dieser Menge haben wir den Namen *A* gegeben. Die vier Objekte sind die Zahlen 1, 2, 3 und 42. Wenn es sich um überschaubare Mengen wie diese handelt, benutzt man die obige Schreibweise: die Objekte werden durch Kommata getrennt und von geschwungenen Klammern (auch *Mengenklammern* genannt) eingeschlossen. In PYTHON sieht das genauso aus:

```
A = {1, 2, 3, 42} # oder so: set(1, 2, 3, 42)
```

Wir würden in diesem Fall sagen, dass z.B. die Zahl 3 ein *Element* der Menge *A* ist. Man schreibt⁴ dafür: $3 \in A$. (Man benutzt auch andere Formulierungen, etwa „*A* enthält das Element 3“ oder „3 gehört zu *A*“.) Die Zahl 7

²Keine Programmiersprache kann Mengen im mathematischen Sinne adäquat repräsentieren. Insbesondere gilt das natürlich für unendliche Mengen, mit denen wir uns demnächst beschäftigen werden. Das ist aber nicht so überraschend. Wir wissen ja auch schon, dass sich auch die Zahlen, mit denen man in der Mathematik hantiert, grundlegend von den Zahlen, mit denen Computer rechnen, unterscheiden.

³Die naive Mengenlehre führt unweigerlich zu logischen Paradoxien, den sogenannten *Antinomien*, die man durch den axiomatischen Ansatz vermeiden kann.

⁴Das ist eine stilisierte Version des griechischen Epsilon. Dieses Symbol (und diverse andere, die heute noch verwendet werden) wurde von dem italienischen Mathematiker Giuseppe Peano eingeführt.



hingegen gehört nicht zu den Objekten, deren Gesamtheit A bildet. Sie ist also *kein* Element von A und man schreibt dann: $7 \notin A$. In PYTHON benutzt man dafür einen Operator, den wir von Listen schon kennen:

```
3 in A, 7 in A, 7 not in A
```

in

Jedes Objekt, das man sich vorstellen kann, ist entweder ein Element von A oder es ist kein Element von A . Und das ist die definierende, und aus Sicht der Mengenlehre *einzig relevante*, Eigenschaft einer Menge: Die Identität einer Menge ergibt sich aus ihren Elementen.

Außerdem sind die *Zahl* 42 und die *Menge* $\{42\}$ zwei ganz unterschiedliche Dinge. Es besteht eine Beziehung zwischen ihnen (nämlich $42 \in \{42\}$), aber sie sind *nicht* identisch.⁵

Schauen wir uns nun diese Menge an:

$$B = \{3, 42, 2, 1\}$$

Jedes Objekt, das Element von A ist, ist auch Element von B . Jedes Objekt, das *nicht* Element von A ist, ist auch nicht Element von B . (Und das gilt ebenso, wenn wir in den beiden vorherigen Sätzen A mit B vertauschen.) Wenn also die Identität einer Menge ausschließlich von ihren Elementen abhängt, sind A und B *identisch*:

$$A = B$$

A und B sind also nur zwei Namen für *dasselbe* Objekt. Man kann es auch so formulieren: Bei der Angabe der Elemente einer Menge spielt die Reihenfolge keine Rolle. Und in PYTHON funktioniert das auch so:⁶

```
B = {3, 42, 2, 1}
A == B
```

Identität ist ohnehin ein ganz wesentliches Konzept in der Mathematik. Die obige Menge gibt es nur einmal, wir hatten ihr lediglich zwei verschiedene Namen gegeben. Ebenso gibt es z.B. die Zahl 3 nur einmal. Das hat aber dann auch Konsequenzen für diese Menge:

$$C = \{1, 2, 3, 3, 42\}$$

Da es die 3 nur einmal gibt, hat es für die Identität der Menge keine Auswirkungen, wie oft man sie hinschreibt; entweder ist 3 ein Element von C oder

⁵Solche Mengen mit nur einem Element spricht man typischerweise als „*Singleton*“ aus.

⁶Wenn Sie sich B anzeigen lassen, dann wird PYTHON für die Ausgabe dieselbe Reihenfolge wie bei A verwenden. Das ist nur „zufällig“ (aus technischen Gründen) so. Es ist ein Beleg dafür, dass die Reihenfolge keine Rolle spielt.

nicht. Damit ist aber klar, dass C nur ein weiterer Name für die Menge ist, die wir schon kennen:

$$A = B = C$$

Auch das hat in PYTHON seine Entsprechung:

```
C = {1, 2, 3, 3, 42}
A == C
```

Aufgabe 227: Wie unterscheiden sich angesichts dessen, was Sie bisher gelernt haben, in PYTHON Mengen von Listen?

len

Aufgabe 228: Führen Sie den folgenden Code aus und erklären Sie das Ergebnis:

```
A = [1, 1, 2]
B = {1, 1, 2}
len(A), len(B)
```

WENN WIR UNS DIE BEIDEN MENGEN

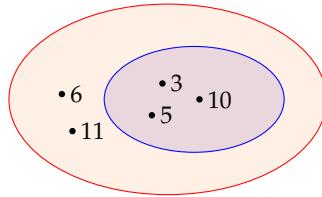
$$X = \{3, 5, 10\} \quad \text{und} \quad Y = \{3, 5, 6, 10, 11\}$$

anschauen, so sehen wir, dass Y Elemente hat, die nicht Elemente von X sind (z.B. die Zahl 6). X und Y sind also *nicht* identisch. Allerdings ist *jedes* Element von X auch ein Element von Y . In so einem Fall sagt man, dass X eine **Teilmenge** von Y ist, und schreibt: $X \subseteq Y$. (Man sagt manchmal auch, dass Y **Obermenge** von X ist.) Gilt so eine Beziehung *nicht*, so schreibt man $X \not\subseteq Y$. (Das bedeutet also lediglich, dass X mindestens ein Element enthält, das *nicht* zu Y gehört.)

Die Beziehungen zwischen Mengen lassen sich in einfachen Fällen grafisch durch sogenannte **Mengendiagramme** darstellen.⁷ Dabei stellt man die relevanten Objekte (Elemente) durch beschriftete Punkte dar und die Mengen durch Flächen, so dass ein Objekt Element einer Menge ist, wenn der zugehörige Punkt innerhalb der entsprechenden Fläche liegt. In unserem Fall sieht das folgendermaßen aus:



⁷Da die Mengenlehre so grundlegend für die Mathematik ist, wurde sie eine Zeit lang in den Grundschulen unterrichtet, ohne dass die Kinder vorher etwas über Zahlen und Rechnen lernten. Das stieß auf erbitterten Widerstand, insbesondere bei den Eltern. Mengendiagramme wurden zum Symbol der verhassten sogenannten „Neuen Mathematik“. Dem SPIEGEL war das 1974 sogar ein provokantes Titelbild wert.



Die Menge X wird dabei durch die kleine, blau berandete Ellipse dargestellt, während die Menge Y durch die große, rot berandete repräsentiert wird. Die oben definierte Teilmengenbeziehung liegt dann vor, wenn eine Fläche komplett innerhalb einer anderen liegt. (Oder noch etwas genauer: wenn kein Stück der Fläche der Teilmenge außerhalb der Fläche der Obermenge liegt.)

In PYTHON wird für \subseteq das zusammengesetzte Symbol $<=$ „missbraucht“:⁸

```
{1, 4, 3} <= {1, 2, 3, 4}
```

Aufgabe 229: Man kann in PYTHON mit `for` durch die Elemente einer Menge iterieren. Das funktioniert genauso wie bei Listen, allerdings kann man sich nicht darauf verlassen, dass die Menge in einer bestimmten Reihenfolge durchlaufen wird.

for

Schreiben Sie eine Funktion `subset`, die \leq für Mengen ersetzen kann. Wenn A und B Mengen sind, dann soll `subset(A, B)` also genau dann `True` zurückgeben, wenn A eine Teilmenge von B ist.

Aufgabe 230: Wenn $A \subseteq B$ gilt, hat B dann mehr Elemente als A ?

Aufgabe 231: Wenn $A \not\subseteq B$ gilt, gilt dann $B \subseteq A$?

Aufgabe 232: Wenn $A \not\subseteq B$ gilt, hat B dann weniger Elemente als A oder genauso viele? Oder mehr?

Aufgabe 233: Wenn Sie zwei Mengen A und B haben, von denen Sie wissen, dass sowohl $A \subseteq B$ als auch $B \subseteq A$ gilt, was können Sie dann über A und B aussagen?

Es spricht nichts dagegen, eine Menge hinzuschreiben, die so

$$\{\}$$

aussieht. Diese Menge hat *keine* Elemente. Für jedes beliebige Objekt a , das Sie sich vorstellen können, gilt also $a \notin \{\}$. Da eine Menge durch die Angabe ihrer Elemente eindeutig bestimmt ist, kann es nur *eine* Menge geben, die gar keine Elemente hat. Man nennt sie **die leere Menge** (und nicht etwa „eine“ leere Menge) und benutzt für sie auch das Symbol \emptyset . In PYTHON wird die

⁸Wie bei `len` (siehe Aufgabe 228) ist die Wahl des Namens etwas unglücklich, weil sie zu falschen Assoziationen führen kann. Siehe auch Aufgabe 231.

Zeichenfolge {} für etwas anderes als Mengen benutzt. Man muss die leere Menge daher mittels `set()` erzeugen.

Aufgabe 234: Auf `set() <= {1, 2, 3}` wird PYTHON mit True antworten. Und ein Mathematiker würde Ihnen ebenfalls sagen, dass $\emptyset \subseteq \{1, 2, 3\}$ gilt. Er würde Ihnen sogar sagen, dass $\emptyset \subseteq A$ für jede Menge A gilt. Können Sie sich das erklären?

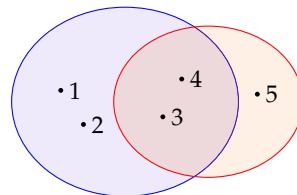


MIT VERKNÜPFUNGEN WIE + ODER · KANN MAN zwei Zahlen kombinieren, um eine neue zu erhalten. Wir wollen uns nun Verknüpfungen für *Mengen* anschauen; also solche, die zwei Mengen zu einer neuen verknüpfen.

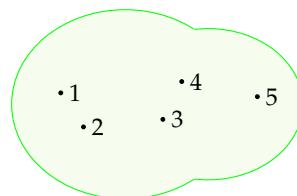
Die **Vereinigung** zweier Mengen ist die Menge, die aus allen Elementen besteht, die in mindestens einer der beiden Mengen enthalten sind. Man benutzt dafür das Symbol \cup . Mit

$$A = \{1, 2, 3, 4\} \quad \text{und} \quad B = \{3, 4, 5\}$$

erhält man z.B. $A \cup B = \{1, 2, 3, 4, 5\}$. Im Mengendiagramm sieht das folgendermaßen aus:



A entspricht der blau berandeten Fläche, B der rot berandeten. Und $A \cup B$ ist die „Gesamtfläche“:



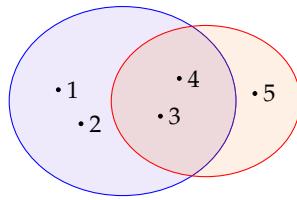
In PYTHON erreicht man dasselbe Ergebnis so:

```
{1, 2, 3, 4} | {3, 4, 5}
```

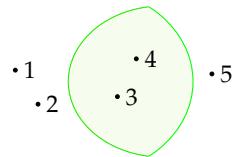
Der **Durchschnitt** zweier Mengen ist die Menge, die aus allen Elementen besteht, die in *beiden* Mengen enthalten sind. Man benutzt dafür das Symbol \cap . Setzt man wie oben

$$A = \{1, 2, 3, 4\} \quad \text{und} \quad B = \{3, 4, 5\},$$

so erhält man $A \cap B = \{3, 4\}$. Im Mengendiagramm ist die Ausgangssituation identisch:



Der Durchschnitt sieht dann so aus:



Und in PYTHON macht man es folgendermaßen:⁹

```
{1, 2, 3, 4} & {3, 4, 5}
```

&

Aufgabe 235: Wenn Sie noch nie vorher etwas von diesen Mengenoperationen gehört haben, dann denken Sie sich Paare von Mengen aus und berechnen Sie jeweils Vereinigung und Durchschnitt. Überprüfen Sie dann Ihre Ergebnisse mit PYTHON.

Aufgabe 236: Was kommt heraus, wenn man $A \cap A$ und $A \cup A$ für eine Menge A ermittelt?

Aufgabe 237: Man nennt zwei Mengen *disjunkt*, wenn ihr Durchschnitt die leere Menge ist. (Sie haben dann also nichts gemeinsam.) Geben Sie zwei Mengen an, die disjunkt sind, die aber beide selbst nicht leer sind.

Eine weitere Mengenverknüpfung ist die *mengentheoretische Differenz*, für die man das Symbol \setminus benutzt:¹⁰ $A \setminus B$ ist die Menge, die aus allen Elementen von A besteht, die *nicht* Elemente von B sind.

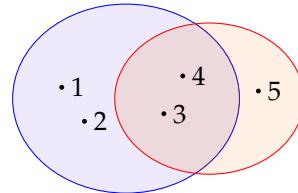
Mit unserem Beispiel von eben

$$A = \{1, 2, 3, 4\} \quad \text{und} \quad B = \{3, 4, 5\}$$

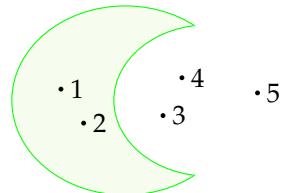
hat man $A \setminus B = \{1, 2\}$. Wir gehen wieder von demselben Mengendiagramm aus:

⁹Falls Ihnen aufgefallen ist, dass man in PYTHON dieselben Symbole für Vereinigung und Durchschnitt benutzt, die man in Sprachen wie C und JAVA als Junktoren verwendet: das ist Absicht. Sie werden den Grund dafür bald sehen.

¹⁰In manchen Büchern wird auch einfach das Minuszeichen verwendet.



Das Ergebnis ist in diesem Fall die folgende Menge:



Und in **PYTHON** geht es so:

```
{1, 2, 3, 4} - {3, 4, 5}
```

Aufgabe 238: Geben Sie für das obige Beispiel $B \setminus A$ an. Was ist der Unterschied zwischen der mengentheoretischen Differenz und den beiden Verknüpfungen, die wir uns vorher angesehen haben?

add

Aufgabe 239: Mit `A.add(x)` können Sie in **PYTHON** zu einer Menge `A` ein Element `x` hinzufügen. Schreiben Sie Funktionen, die Vereinigung, Durchschnitt und mengentheoretische Differenz implementieren.

Aufgabe 240: Als Lösung der vorherigen Aufgabe gibt ein Kommilitone von Ihnen die folgende Funktion für die Vereinigung ab:

```
def myUnion (A, B):
    for x in B:
        A.add(x)
    return A
```

Die Funktion gibt zwar das korrekte Ergebnis zurück, ist aber aus Sicht der Informatik keine gute Lösung. Warum?

Aufgabe 241: Wenn man die Methode `add` (siehe Aufgabe 239) nicht hätte, wie könnte man stattdessen zu einer Menge ein Element hinzufügen?

Aufgabe 242: Im Folgenden sind eine Reihe von Aussagen aufgeführt, die für *alle* Mengen A , B und C gelten. Einige davon wurden im Text schon behandelt, die meisten aber nicht.

- $A \subseteq A$ (\subseteq ist reflexiv.)
- $\emptyset \subseteq A$