# Klassen I

Ein sehr nützliches Konzept sind Klassen. Diese stellen eine Art Bauplan dar, nachdem man beliebig viele Dinge erzeugen kann, die jeweils einen definierte Eigenschaften und Funktionen haben. Jedes Ding, auch Objekt oder Instanz einer Klasse genannt, hat die gleichen Eigenschaften (Größe, Farbe), diese können allerdings unterschiedliche Werte haben.

> Gummibärchen haben alle die gleiche Form, es gibt sie jedoch in verschiedenen Farben, Größen und Geschmacksrichtungen.

Eine Klasse beschreibt das 'Aussehen' und Verhalten eines Dinges, aus der Sicht dieses Dinges. Das hört sich jetzt vielleicht etwas komisch an...

> Stell dir vor, du bist das Gummibärchen, du weißt welche Farbe du hast und wonach du schmeckst.

Eine Klasse für ein Gummibärchen könnte also so aussehen:

```
1  class Gummibaerchen:
2
3    def __init(self, farbe, geschack):
4      self.farbe = farbe
5      self.geschack = geschmack
```

## 1 Die `__init__()`-Methode - Konstruktor

Die wichtigste Methode jeder Klasse ist die `__init__()`-Methode. Diese wird genau einmal aufgefufen, wenn ein neues Ding erstellt wird. Hier werden alle Eigenschaften, die ein Ding hat angelegt.
Um zu kennzeichnen, dass eine Eigenschaft/Variable zu einer Klasse gehört und nicht nur eine normale Variable ist müssen diese mit `self.mein_variable` erstellt werden. Diese bedeuet, dass die Variable zu diesem Ding gehört und jederzeit abgefragt werden kann!

## 2 Dinge/Instanzen erstellen

Eine Klasse an sich tut noch nichts, sie gibt nur vor, wie sich Instanzen dieser Klasse verhalten. Hat man eine Klasse kann man verschiedene Instanzen erstellen. Dazu verwendet man den Namen der Klasse (hier `Gummibaerchen`) und übergibt in Klammern alle von `__init__()` geforderten Parameter. Wie bereits beschrieben, wird beim Erstellen die `__init__()`-Methode aufgerufen und so werden die Parameter an diese übergeben.

```
1  # Erstellt zwei Instanzen
2  baer1 = Gummibaerchen("rot", "Erdbeere")
3  baer2 = Gummibaerchen("gruen", "Apfel")
4
5  # Die Eigenschaften abfragen:
6
```

```
print(baer1.farbe) # rot
print(baer1.geschmack) # Apfel
```

In diesem Beispiel sieht man wie man Instanzen erstellt und Werte abfragt. Jede Instanz hat die in der Klasse definierten Variablen `farbe` und `geschmack`. Diese haben verschiedene Werte und können über den Namen der Instanz abgefragt `baer1.farbe` und sogar geändert werden `baer1.farbe = 'blau'`.

# 3 Sichtweise: Von Innen vs. von Außen

In der Klassendefinition werden Eigenschaften mit `self.farbe` abgefragt, bzw. gesetzt. Wir befinden uns innerhalb der Klasse, sind als quasi das Gummibaerchen selber (engl. *self*). Was wir hier programmieren, wird sich auf alle `Gummibaerchen`-Instanzen auswirken, wenn dieser Code für die jeweilige Instanz aufgefufen wird.
Haben wir eine Instanz erstellt, müssen wir angeben auf welcher Instanz wir arbeiten wollen. Hier ist `self` nicht eindeutig, weil wir z.B. `baer1` und `baer2` haben. Aus diesem Grund verwenden wir den Namen der Instanz und geben dann an, welche Eigenschaft wir abfragen/verändern wollen.

## 3.1 Klassenmethoden

Zusätzlich zu Eigenschaften können Klassen bzw. Instanzen davon natürlich auch Funktionen haben. Bisher haben wir nur die Spezialmethode `init()` gesehen. Eine Funktion wird innerhalb der Klasse (eingerückt) ganz normal definiert. Einzige Ausnahme ist, dass der erste Parameter **IMMER** `self` sein muss!

```
class Gummibaerchen:

  def __init(self, farbe, geschack):
    # ...

  def info(self):
    print("Gummibaerchen in ", self.farbe, " mit Geschmack",
     ↪ self.geschmack)
```

Eine einfache Methode, die Information über unser Gummibärchen ausgibt. Da wir uns innerhalb der Klasse (des Gummibaerchen) befinden, können wir die Farbe über `self.farbe` abfragen. Zu diesem Zeitpunkt gibt es ja noch keine konkreten Bärchen! Wir fügen lediglich jedem Bär die Funktion `info()` hinzu. Diese steht ab sofort jedem `Gummibaerchen` zur Verfügung:

```
baer1 = Gummibaerchen("rot", "Erdbeere")
baer2 = Gummibaerchen("gruen", "Apfel")

baer1.info() # Gummibaerchen in rot mit Geschmack Erdbeere
baer2.info() # Gummibaerchen in gruen mit Geschmack Apfel
```

Wichtig, beim Aufrufen muss man `self` nicht mit übergeben, obwohl es in der Methodendefinition eigentlich gefordert wird! Dies ist eine Art Trick, damit in der Methode `self` immer zur Verfügung steht. Bei einem Aufruf einer Klassenmethode über eine Instanz wird `self` immer auf die aktuell aufrufende Instanz (`baer1` in `baer1.info()`).