Objektorientierte Programmierung

Die objektorientierte Programmierung ist ein wichtiges Konzept in der modernen Programmierung. Doch in wie weit unterscheidet sie sich von der Programmierung in Scratch? Die Antwort ist, dass sie sich nicht wirklich davon unterscheidet. Trotzdem ist es wichtig zu wissen, was Objekte und Klassen sind und was diese mit der objektorientierten Programmierung zu tun haben.

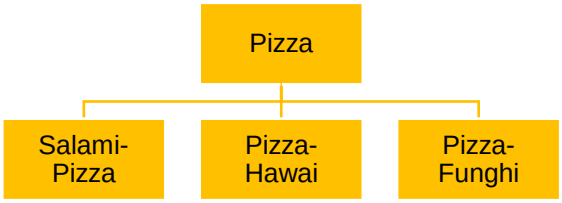
Ein Beispiel aus dem Alltag

Schauen wir uns doch mal die folgenden drei Pizzen an:



Sie haben alle einen unterschiedlichen Belag, aber bei allen handelt es sich um eine Pizza. Für eine Pizza lässt sich festlegen, dass jede Pizza einen Grundteig (z. B. normaler Hefeteig oder Vollkornteig) und verschiedene Zutaten für den Belag besitzt. Zusätzlich gibt es noch bestimmte Tätigkeiten, die mit einer Pizza möglich sind, wie zum Beispiel bestellen, backen, schneiden oder essen. Doch was hat das mit Objekten oder Klassen zu tun?

Diese Frage ist ganz einfach zu beantworten. Die verschiedenen Varianten von Pizzen sind unsere Objekte und diese lassen sich alle als Pizza klassifizieren. Oder anders gesagt: Pizza bildet eine Klasse und die tatsächliche Umsetzung einer Pizza, zum Beispiel die Salami-Pizza, ist ein Objekt dieser Klasse.



Bitte einmal Pizza mit Python …

Doch wie sieht das ganze in Python aus und was kann man damit machen? Eine Umsetzung des Pizza-Beispieles kann wie folgt aussehen:

```
01  class Pizza:
02      def __init__(self):
03          self.teig = 'Hefeteig'
04          self.zutaten = ['Tomatensosse', 'Kaese']
05          self.rand = 'ohne'
06
07      def bestellen(self):
08          pass
```

Klassen werden in Python mit dem Befehl *class* eingeleitet. Innerhalb von Klassen wird der Quellcode so eingerückt, wie es bei Schleifen oder Bedingungen bekannt sein sollte. Die Methode *__init__* wird ausgeführt, sobald ein Objekt dieser Klasse initialisiert (erzeugt) wird. Im obigen Beispiel wird der Teig standardmäßig auf Hefeteig, der Rand auf ohne und die Zutaten

auf Tomatensoße und Käse gesetzt. Dies soll die Vorlage für jede Pizza sein (dies kann sich aber bei einzelnen Objekten ändern). Wichtig ist die Verwendung von *self*. Bei der Definition der Methode *__init__* (z.B. Zeile 2) wird es innerhalb der Klammern eingebunden. Dies ist wichtig, da durch *self* gekennzeichnet wird, dass die daran anschließende Variable (siehe z. B. Zeile 3) zu den einzelnen Objekten gehört. Tut man dies nicht, so könnte es sich um eine beliebige globale Variable handeln. Der Befehl *pass* wird benutzt, wenn eine Methode (innerhalb von Klassen nennt man Funktionen Methoden) nicht detaiiert beschreiben möchte.

Wie bereits erwähnt ist die objektorientierte Programmierung nicht viel anders als das bekannte Programmieren in Scratch. In Scratch stellten die einzelnen Sprites/Figuren die Objekte und Klassen dar, wobei Objekt und Klasse hier sehr eng verbunden waren. Das eigentliche Scratch-Programm hatte mehrere Sprites bzw. Klassen, welche miteinander interagiert haben.

Wo liegt der nutzen?

An dieser Stelle ist die Frage nach dem Nutzen sehr berechtigt. Ein sehr großer Vorteil, den du auch in den kommenden Unterrichtsstunden noch sehen wirst, ist, dass Variablen und Methoden sinnvoll zusammengepackt werden. Schau dir dieses Beispiel einmal an:

```
01  class Pizza:
02      def __init__(self):
03          self.teig = 'Hefeteig'
04          self.zutaten = ['Tomatensosse', 'Kaese']
05          self.rand = 'ohne'
06
07      def bestellen(self):
08          print('Pizza mit ' + self.teig + ' und ' + self.rand + \
09                '-Rand wird bestellt.')
10          print('Zutaten-Liste:')
11          print(self.zutaten)
12
13      def vergleiche_mit(self, andere_pizza):
14          pass
15
16  pizza1 = Pizza()
17  pizza1.zutaten.append('Salami')
18  pizza1.zutaten.append('Schinken')
19  pizza1.rand = 'Käse'
20
21  pizza2 = Pizza()
22  pizza2.teig = 'Vollkornteig'
23  pizza2.zutaten.remove('Tomatensoße')
24  pizza2.zutaten.append('Currysoße')
25  pizza2.zutaten.append('Ananas')
26  pizza2.zutaten.append('Hähnchenfleisch')
27  pizza2.rand = 'mit'
28
29  pizza1.bestellen()
30  pizza2.bestellen()
31  pizza1.vergleiche_mit(pizza2)
```

Wichtig ist, auf das self zu achten.

Die Methode definieren wir nich vollständig und nutzen daher das *pass*.

Objekte werden initialisiert.

Die Methode ist zwar nicht denau definiert, aber man erkennt, dass auch Objekte übergeben werden können.
Dies hat den Vorteil, dass nicht alle Variablen des Objekt an dessen stelle übergeben werden müssen.

Im obigen Beispiel sieht man, wie zwei Objekte der Klasse Pizza initialisiert (erzeugt) und angepasst werden. Dabei sind die Eigenschaften der beiden Pizzen voneinander getrennt und beeinflussen sich gegenseitig nicht. Gerade in größeren Programmen ist es wesentlich einfacher mit Objekten zu arbeiten, als ohne. Neben diesem Vorteil gibt es aber noch weitere Vorteile, auf die wir jedoch erst später im Unterricht eingehen werden. Übrigens kann man mit \ einen Zeilenumbruch im Quellcode anzeigen (siehe Zeile 8).