

# LU12c - Objekte



Ein Objekt ist ein Exemplar oder Instanz einer Klasse. Während die Klasse definiert, welche Attribute vorhanden sind, enthalten Objekte die konkreten Daten.

## Objekt erzeugen

Um ein Objekt zu erzeugen, benötigen wir:

- Eine (Referenz-)Variable die das Objekt speichert.
- Die Klasse als Import.

```
from Member import Member

def main():
    some_member = Member()
```

Ein so erzeugtes Objekt wird noch keine Daten enthalten. Wir können diese anschliessend hinzufügen (siehe Abschnitt „Zugriff“)

## Objekt mit Daten erzeugen

Anstatt ein leeres Objekt zu erzeugen, können wir direkt die gewünschten Daten mitgeben:

```
from Member import Member

def main():
    another_member = Member('David', 'Demuth', 'Wegweg 0', 'Hier', '1234',
                             2010, 1998, False)
```

Entweder müssen wir die Argumente in den runden Klammern in der gleichen Reihenfolge angeben, wie die Attribute in der Klasse definiert wurden.

Alternativ können Sie jedem Argument den Attributnamen voranstellen. In diesem Fall ist die Reihenfolge egal.

```
...
old_member = Member(
    place='Dort',
    address='Hauptgasse 9x',
    firstname='Emil',
```

```
honorary_member=True,  
birth_year=1965,  
lastname='Emmerson',  
zip_code='9876',  
entry_year=2001  
)
```

*Hinweis:* Die Zeilenumbrüche wurden zur besseren Übersicht eingesetzt.

## Zugriff auf Attribute



Über die Referenzvariable (z.B. member) und den Attributnamen (z.B. firstname) können Sie die Daten in einem Attribut lesen oder ändern.

Im ersten Codebeispiel haben wir ein leeres Member-Objekt angelegt. Nun möchten wir den Attributen dieses Objekts Werte zuweisen und die Werte wieder auslesen.

```
from Member import Member  
  
def main():  
    some_member = Member()  
    some_member.firstname = 'Fabienne'  
    some_member.lastname = 'Fröhlich'  
    some_member.place = 'Freiburg'  
    ...  
  
    print(some_member.lastname)
```

Wir können Attribute fast wie sonstige Variablen verwenden. Der offensichtlichste Unterschied ist, dass wir immer die Referenzvariable auf das Objekt angeben müssen.

## Listen von Objekten

Vielleicht denken Sie sich jetzt: „Schön, ich brauche nicht mehr 8 Listen mit den einzelnen Attributen. Stattdessen brauche ich eine eigene Referenzvariable für jedes der 500 Clubmitglieder!“. Bevor Sie nun die Objekte member001, member002, ... erstellen: Listen können (auch) Objekte enthalten!

```
from Member import Member  
  
def main():  
    member_list = []
```

```
member_list.append(Member('David', 'Demuth', 'Wegweg 0', 'Hier', '1234',  
2010, 1998, False))  
member_list.append(Member('Anna', 'Amstutz', 'Paradeplatz 12',  
'Sonstwo', '5555', 2013, 1999, True))  
member = Member(  
    place='Dort',  
    address='Hauptgasse 9x',  
    firstname='Emil',  
    honorary_member=True,  
    birth_year=1965,  
    lastname='Emmerson',  
    zip_code='9876',  
    entry_year=2001  
)  
member_list.append(member)  
for item in member_list:  
    print(item.firstname)
```

Im Beispiel sehen Sie, wie Objekte in einer Liste gespeichert und über einen for-Loop ausgelesen werden.

From:  
<https://wiki.bzz.ch/> - **BZZ - Modulwiki**

Permanent link:  
<https://wiki.bzz.ch/modul/archiv/m319python/learningunits/lu12/objekt>

Last update: **2024/03/28 14:07**

