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 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,['Hauptversammlung','Skitag'])
```

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

Alternativ können Sie jedem Argument den Attributsnamen 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',
        attended_meetings = ['Hauptversammlung','Skitag'],
        entry_year=2001
        )
```

Hinweis: Die Zeilenumbrüche wurden zur besseren Übersicht eingesetzt.

Zugriff auf Attribute

Über die Referenzvariable (z.B. member) und den Attributsnamen (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'
    some_member.attended_meetings = ['Hauptversammlung','Schlitteltag']
    some_member.attended_meetings.append('Skitag')
    ...

    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,['Hauptversammlung','Skitag']))
    member_list.append(Member('Anna', 'Amstutz', 'Paradeplatz 12',
'Sonstwo', '5555', 2013, 1999, True,['Hauptversammlung','Skitag']))
    member = Member(
        place='Dort',
        address='Hauptgasse 9x',
        firstname='Emil',
        honorary_member=True,
        birth_year=1965,
        lastname='Emmerson',
        zip_code='9876',
        entry_year=2001,
        attended_meetings = ['Hauptversammlung','Skitag']
    )
    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.