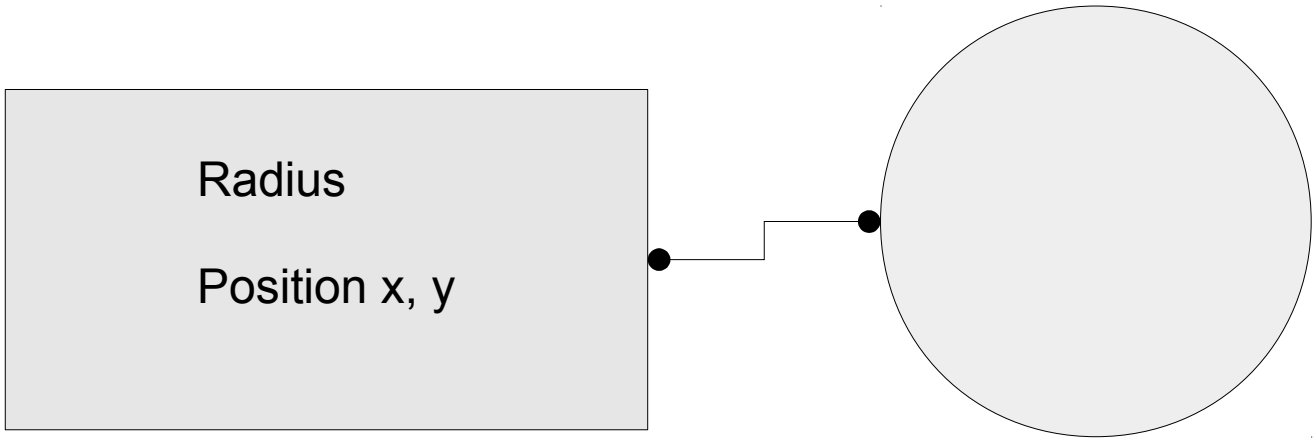
Python
„Attribute in einer Klasse“



Attribute

- Beschreibung eines Gegenstandes, einer Person, etc.
- Allgemeingültige Beschreibung für einen bestimmten Objekttyp.

… in einer Projektbeschreibung

- Der *Radius* eines Kreises wird bei der Erzeugung gespeichert. Der Radius kann im Leben des Kreises vergrößert oder verkleinert werden. Ein Kreis wird an einer bestimmten *Position* x, y gezeichnet und kann verschoben werden.
- Eine Ware in einem Geschäft hat eine exakte *Bezeichnung* und einen bestimmten *Verkaufspreis*.

Instanzvariablen (Member)

- Nutzung von Variablen in einer Klasse.
- Abbildung von Attributen eines Objekts in einer Programmiersprache.
- Speicherung von Attribut-Werten pro Instanz. Instanzen haben zwar die gleichen Attribute, aber die Ausprägungen können unterschiedlich sein.

… definieren

- Dort wo sie das erste Mal gebraucht wird. Instanzvariablen können in jeder beliebigen Methode einer Klasse definiert werden.
- Bei Erzeugung der Instanz. In der Initialisierungsmethode werden alle benötigten Attribute eines Objekts mit der gewünschten Ausprägung oder einem Standardwert belegt.
- Falls die Instanzvariable vor der Nutzung nicht definiert ist, wird der Laufzeitfehler *AttributeError: ... object has no attribute ...* angezeigt.

... in jeder beliebigen Methode

```
class clsPerson(object):

    def __init__(self, nachname, anrede,  titel, vorname):
        self.nachname = nachname

        self.anrede = anrede
        self.titel = titel
        self.vorname = vorname

    def setAdresse(self, strasse, hausnummer, postleitzahl, ort):
        self.adresse = {"Strasse" : strasse, "Hausnummer":hausnummer,
                        "Postleitzahl":postleitzahl, "Ort":ort}
```

... in der Initialisierungsmethode

```
class clsPerson(object):

    def __init__(self, nachname, anrede,  titel, vorname):
        self.nachname = nachname

        self.anrede = anrede
        self.titel = titel
        self.vorname = vorname

        self.Adresse = None

    def setAdresse(self, strasse, hausnummer, postleitzahl, ort):
        self.adresse = {"Strasse" : strasse, "Hausnummer":hausnummer,
                        "Postleitzahl":postleitzahl, "Ort":ort}
```

Wertzuweisung

self	.	nachname	=	nachname
Instanz	.	Variable	=	Ausdruck

- Mit Hilfe des Gleichheitszeichen wird der Instanzvariablen ein Wert zugewiesen.
- Die Instanzvariable wird automatisiert durch die Wertzuweisung angelegt.

Wert einer Instanzvariablen

self	.	nachname	=	nachname
Instanz	.	Variable	=	Ausdruck

- In der Initialisierungsmethode werden häufig die Werte der Parameter in einer Instanzvariablen gespeichert. Der Name des Parameters und der Instanzvariablen können gleich sein, müssen aber nicht.
- In einer Methode kann die Ausprägung eines Attributs mit Hilfe eines Ausdrucks berechnet werden.
- Die Instanzvariable kann in einer beliebigen Methode einen Wert als Literal übergeben bekommen.

Attribut der Instanz

self	.	nachname	=	nachname
Instanz	.	Variable	=	Ausdruck

- Das Objekt, welches durch die angegebenen Attribute beschrieben wird, werden mit dem Punktoperator verbunden.
- Die aktuell zu ändernde Instanz wird links vom Punktoperator definiert.
- Das Attribut des gebundenen Objekts wird rechts vom Punktoperator angegeben.

Name einer Instanzvariablen

self	.	breite
Instanz	.	Variable

- Der Bezeichner kommt exakt einmal in der Klasse vor.
- Der Bezeichner der Instanzvariablen ist frei wählbar.
- Der Name sollte aber das Attribut des Objektes in der realen Welt widerspiegeln.

Regeln für Bezeichner

- Der Bezeichner beginnt mit einem Buchstaben oder Unterstrich.
- Ein Bezeichner besteht aus den lateinischen Groß- und Kleinbuchstaben und den Ziffern.
- Ein Bezeichner enthält als Sonderzeichen nur den Unterstrich.
- Die Groß- und Kleinschreibung wird beachtet.
- Schlüsselwörter der Programmiersprache und Bezeichnungen für Elemente aus der Standardbibliothek sind nicht erlaubt.

Anbindung an ein Objekt

self	.	breite
Instanz	.	Variable

- Die Variable kann eine beliebige Instanz gebunden werden. Die Instanzvariable ist in der Klasse, auf die die Instanz basiert, definiert.
- Häufig wird die Instanz `self` genutzt.

Parameter self

```
def __init__(self, nachname, anrede,  titel, vorname):
    self.nachname = nachname
```

- Der erste Parameter einer Methode muss immer self sein.
- Wer hat die Methode aufgerufen? Welche Instanz wird erstellt?
- Der Parameter verweist auf die aktuelle Instanz. An diese Instanz wird ein Attribut gebunden.

Attribute mit einem undefinierten Wert

```
class clsPerson(object):

    def __init__(self, nachname, anrede,  titel, vorname):
        self.Adresse = None
```

- Jede Person wohnt an einer Adresse. In einigen Fällen ist die Adresse bei Erzeugung der Person nicht bekannt.
- Die Wert None definiert eine Instanzvariable. Der Wert der Instanzvariablen ist aber nicht bekannt.
- Mit Hilfe einer Methode kann der Attribut-Wert angepasst werden.

Attribute mit einem definierten Wert

```
class clsPerson(object):


   def __init__(self, nachname, anrede,  titel, vorname):
      self.nachname = nachname
```

- Jede Person hat einen Nachnamen. Der Nachname ist bei der Erzeugung der Person bekannt.
- Der Wert wird mit Hilfe eines Parameters an eine Methode übergeben.
- Der Attribut-Wert wird mit Hilfe eines Ausdrucks berechnet. Strings können zu einer neuen Zeichenkette verknüpft werden.

Variablen und Instanzvariablen

- Parameter in einer Parameterliste in einer Methode und lokale Variablen können nur in einer Methode genutzt werden. Dort wo sie definiert sind, können sie genutzt werden.

variable

- Die Instanzvariable ist an ein Objekt gebunden. Durch die Definition in einer beliebigen Methode kann die Variable überall in der Klasse genutzt werden.

instanz . variable

Beispiel

```
class clsPerson(object):
    def getAdresse(self):
        strAdresse = ""

        for key, value in self.adresse.items():
            if len(strAdresse) > 0:
                strAdresse = strAdresse + '\n'

            strAdresse = strAdresse + key + ": " + value

        return strAdresse
```

Sichtbarkeit von lokalen Variablen

- Lokale Variablen werden in einem Codeblock definiert. In diesem Codeblock sind sie gekapselt.
- In diesem Codeblock sind die lokalen Variablen sichtbar.
- In diesem Codeblock können die lokalen Variablen genutzt werden.

Sichtbarkeit von Instanzvariablen

- Instanzvariablen sind in einer Klasse gekapselt.
- Instanzvariablen werden in einer Klasse definiert. In dieser Klasse sind sie sichtbar und nutzbar.
- In Abhängigkeit der gewählten Instanz wird der Wert der Instanzvariablen angezeigt.

Zugriff auf Attribute

- Privater (privat) oder öffentlicher (public) Zugriff auf Instanzvariablen.
- Geschützter (protected) Zugriff bei einer Eltern-Kind-Beziehung von Objekten ist in Python nicht implementiert.
- Kennzeichnung von Instanzvariablen oder Methoden entsprechend des gewünschten Zugriffs.

Kein Präfix

```
class clsRechteck(object):

    def __init__(self, breite = 1):
        self.breite = breite
        self.hoehe = None
```

```
>>> rechteck = clsRechteck()
>>> rechteck.breite = 5
>>> rechteck.hoehe = 4
```

Erläuterung

- Standardmäßig sind alle, an ein Objekt gebundene Variablen öffentlich.
- Attribut-Werte können von außen unkontrolliert manipuliert werden.
- Öffentliche Variablen sind für alle zugänglich.

Präfix: Zwei Unterstriche

```
class clsRechteck(object):

    def __init__(self, breite = 1):
        self.__breite = breite
        self.__hoehe = None
```

```
>>> rechteck = clsRechteck()
>>> rechteck.breite = 5
>>> rechteck.hoehe = 4
```

Erläuterung

- Kennzeichnung mit zwei Unterstrichen vor dem Bezeichner.
- Kapselung von Attributen in einer Klasse. Der Attribut-Wert wird kontrolliert mit Hilfe einer Methode verändert.
- Laut Konvention sind die Attribute nach außen geschützt. Falls das Attribut an ein Kind vererbt wird, kann das Kind dieses Attribut nur mit den entsprechenden Methoden ändern.

Präfix: Ein Unterstrich

```
class clsRechteck(object):

    def __init__(self, breite = 1):
        self._breite = breite
        self._hoehe = None
```

```
>>> rechteck = clsRechteck()
>>> rechteck.breite = 5
>>> rechteck.hoehe = 4
```

Erläuterung

- Kennzeichnung mit einem Unterstrich vor dem Bezeichner.
- Die Attribute können von außen oder von einer erbenden Klasse verändert werden.
- Marker für die Aussage „Bitte diese Variable nicht ändern“.

Property

- Ein Zugriff auf die Instanzvariable erfolgt über Methoden. Die Methoden sind hinter einem Label versteckt.
- Das Label wird wie eine lokale Variable nach außen hin genutzt.

… in Python

```
class clsRechteck(object):

    def __init__(self, breite = 1):
        self.__breite = breite
        self.__hoehe = None

    def __setBreite(self, breite):
        self.__breite = breite

    def __getBreite(self):
        return self.__breite;

    rectBreite = property(__getBreite, __setBreite)
```

1. Schritt

- Die Attribut-Namen beginnen mit zwei Unterstrichen.
- Die Methode „set“ wird definiert. Die Methode verändert exakt ein Attribut des daran angebundenen Objekts.
- Die Methode „get“ wird definiert. Die Methode liest exakt einen Attribut-Wert. Die Methode gibt den Wert einer Instanzvariablen zurück.

2. Schritt: Definition der Property

breite	=	property	(	__getBreite	,	__setBreite	)
variable	=	property	(	get-Methode	,	set-Methode	)

- Die Funktion property ist eine eingebaute Funktion, die ein Property-Objekt zurückgibt.
- Der Funktion wird als erster Parameter die get-Methode übergeben. Der zweite Parameter definiert die set-Methode.
- Das erzeugte Property-Objekt wird in einer Variablen gespeichert. Der Variablen-Namen entspricht dem Attribut in der realen Welt.

Nutzung

```
rechteck = clsRechteck()
rechteck.rectBreite = 5
print(rechteck.rectBreite)
```

- Die Variable verweist auf ein Property-Objekt.
- Wenn der Variablen ein Wert zugewiesen wird, wird die Methode „set“ aufgerufen.
- Wenn die Variable als Parameter oder in einem Ausdruck genutzt wird, wird die Methode „get“ aufgerufen.