Objektorientierung

Claas de Boer, Tilman Hinnerichs

7. Januar 2021

Python-Grundlagen

Rückblick

- Datentypen: `int, str, list, dict, …`
- Komperatoren: `==, !=, …`
- Kontrollstrukturen: `if, elif, else`
- Schleifen: `while, for`
- Funktionen: `return, def f(x): …`

Heute: Objektorientierung

Objektorientierung

- **Konzept:** Software als interagierende „Objekte“
- Objekte haben …
  - Eigenschaften (**Attribute**)
  - Verhalten (**Methoden**)

Wozu?

- „intuitiver“ Entwurf
- Gruppierung von Daten und Verhalten $\rightarrow$ Klasse
- Organisation großer Softwaresysteme
- $\rightarrow$ In Python grundlegend verankert

Beispiel

`random.randint(min, max)` + Variablen → Objekt „Würfel“

- Attribute: `seiten`
- Methoden: `wuerfeln() -> int`

Klassen

- Oft gleichen sich Objekte in Eigenschaften, Verhalten $\rightarrow$ **Klasse**
- Attribute, Methoden bei allen Objekten einer Klasse gleich
- Objekt ist **Instanz** einer Klasse

Objektorientierung in Python

Nutzen von Objekten

- Wir kennen bereits Klassen und Objekte:
  - `"Hello World"` ist **Instanz** von `str`
  - `[1,2]` ist **Instanz** von `list`
- Funktionen `sum([1,2,3])` $\leftrightarrow$ Methoden `[1,2,3].append(4)`!

Nutzen von Objekten

```python
# Erstellen eines neuen Objekts der Klasse list
numbers = [1,2,3]
# numbers ist tatsächlich Instanz von list
print(isinstance(numbers, list))

# Aufrufen der Methode append
numbers.append(4)
print(numbers) # Daten in der Liste wurden geaendert

# len ist (eingebaute) Funktion, nicht Methode
# -> NICHT numbers.len!
print(len(numbers))
```

Eigene Klassen

```python
import random # für heute bleibt 'import' Magie

class Wuerfel:
    # VORSICHT: keine (Instanz-)Attribute hier!

    # spezielle Methode: Konstruktor.
    # wird beim Erstellen einer Instanz aufgerufen
    def __init__(self, seiten):

        # neues Attribut "seiten" wird erzeugt.
        self.seiten = seiten

    # Methoden haben stehts "self" als ersten Parameter
    # -> können so auf eigene Attribute zugreifen
    def wuerfeln(self):
        return random.randint(1, self.seiten)
```

Eigene Klassen

```python
# Erzeugen von Objekt Instanzen (vgl. list())
d6 = Wuerfel(6)
d20 = Wuerfel(20)

# Aufruf der Methode Wuerfel.wuerfeln (vgl. list.append)
wert = d20.wuerfeln()
print(f"Ich habe eine {wert} gewürfelt!")
```

Aufgaben

Aufgaben

1. Implementiere die Würfelklasse, erzeuge einen 20-seitigen Würfel und würfle. Falls du eine 20 würfelst, rufe laut ”20”!
2. Implementiere ein Würfelspiel, bei dem du und der Computer einmal würfeln. Wer die höhere Zahl würfelt, gewinnt!
3. Schreibe eine Klasse „Spieler“, die einen Spieler bei diesem Würfelspiel darstellt. Welche Attribute und Methoden hat ein Spieler?
4. Erweitere dein Würfelspiel, sodass zwei Spielerobjekte gegeneinander spielen.
5. Zwei Spieler sind zu langweilig? Mit der Spielerklasse kannst du einfach mehr Spielerobjekte erzeugen. Erweitere dein Spiel, so dass beliebig viele Spieler spielen können (Hint: Listen)
6. Auch abstrakte Konzepte können als Klasse modelliert werden. Wie könnte man ein Würfelspiel mit mehreren Runden als Klasse darstellen?

Schöne Festtage und einen guten Rutsch ins neue Jahr!