Universität Potsdam
Institut für Informatik

GdP-Rechnerübung

Aufgabenblatt 12
(zuletzt aktualisiert: 6. September 2023)

Lernziele (zum Abhaken): Die Student:innen können...
☐ eine Klasse mit zugehörigen Attributen, Methoden und Operatormethoden implementieren.

☐ unter der Verwendung von Objekten und ihren Eigenschaften, Konstruktoren sowie Methoden vorgegebene Algorithmen implementieren.

23 Objektorientierte Programmierung (OOP)

1. Erstellen Sie eine Klasse namens `Haus` in Python. Es sollen zunächst keine Methoden oder Datenelemente angelegt werden, d.h. der Klassenkörper soll nur eine Zeile mit der Anweisung `pass` beinhalten.

2. Mit welcher Anweisung können Sie nun ein Objekt der Klasse erzeugen?

   ---

3. Objekte der Klasse sollen folgende Datenelemente haben: die Größe in m^2 (`qm`), die Anzahl an Räumen (`anzahl_raeume`) und den Preis pro m^2 (`qmpreis`). Wie können Sie den Datenelementen Ihres erstellten Objekts Werte zuweisen?

   Wie können Sie die Datenelemente des Objekts wieder auslesen?

   ---

4. Konstruktoren sind spezielle Methoden, die zur Erzeugung eines Objekts einer Klasse dienen. Innerhalb des Konstruktors werden meist die Datenelemente des Objekts initialisiert.

   In Python heißen die Konstruktormethoden `__init__`. Wenn Sie innerhalb einer Klasse keine Konstruktoren selbst definieren, hat sie nur einen parameterlosen Standardkonstruktor.

   a) Fügen Sie der Klasse einen dreistelligen Konstruktor hinzu, der alle oben genannten Datenelemente mit übergebenen Werten initialisiert.

b) Testen Sie, ob Sie weiterhin Objekte ohne Eingabeparameter erstellen können:

```
1 h1 = Haus()
```

Was stellen Sie dabei fest?

c) Dieses Problem können Sie umgehen, indem Sie Standardwerte für die Eingabeparameter des Konstruktors definieren. Betrachten Sie zunächst das folgende Beispiel:

```
1 class rectangle:
2     def __init__(self,a=1,b=1):
3         self.a=a
4         self.b=b
5
6 rect1=rectangle()
```

Hier kann ein Objekt der Klasse `rectangle` auch ohne Angabe von `a` und `b` konstruiert werden. Welche Werte haben dann `a` und `b` von `rect1`?

`rect1.a`: _____ `rect1.b`: _____

Sie können Parameter auch per Schlüssel übergeben. So kann man den oben definierten Konstruktor `rectangle` wie folgt aufrufen:

```
1 rect2 = rectangle(a=7)
```

Welche Werte haben dann `a` und `b` von `rect2`?

`rect2.a`: _____ `rect2.b`: _____

d) Definieren Sie Standardwerte für die Eingabeparameter des Konstruktors der Klasse `Haus`.

e) Können Sie nun auch ein Haus nur mit einer Größe und ein anderes nur mit einer Anzahl von Zimmern erstellen? Geben Sie die passenden Konstruktoraufrufe an.

5. Fügen Sie anschließend der Klasse die folgenden Methoden hinzu und testen Sie ihr Verhalten:

a) `calcPrice()`, die den Wert des Hauses aus dem Quadratmeterpreis und der Größe berechnet und diesen zurückgibt.

b) `setPrice(price)`, die den Gesamtpreis des Hauses in Euro übergeben bekommt und von der Größe abhängig den Preis pro m^2 anpasst.

c) `getDescription()`, die eine Beschreibung des Hauses als String zurückgibt, z.B. `"Das Haus hat 250 qm, 12 Zimmer und kostet 1000000€."`

6. Als Nächstes verbessern wir das Format, in dem die Objekte der Klasse ausgegeben werden.

   a) Wenn `h1` ein Objekt der Klasse `Haus` ist: Welche Ausgabe liefert `print(h1)`?

   ---

   b) Ändern Sie das Verhalten Ihrer Klasse so, dass bei der Ausgabe eines Objekts mit `print()` jetzt die Beschreibung des Objekts ausgegeben wird, die Ihre Methode `getDescription()` liefert. Implementieren Sie hierzu eine eigene `__str__`-Methode.

7. Wir wollen nun mehrere Häuser miteinander vergleichen.

   a) Implementieren Sie die Operatormethoden (*Magical Operators*) für `<` (kleiner als) und `==` (gleich)!

   Dabei sollen die Gesamtpreise der Häuser verglichen werden.

   b) Konstruieren Sie zwei Objekte `h_teuer` und `h_guenstig` der Klasse `Haus` so, dass `h_teuer` einen höheren Gesamtpreis aufweist als `h_guenstig`. Überprüfen Sie mit diesen Beispielobjekten, ob die Operationen `!=` und `>` ebenfalls funktionieren, obwohl Sie diese Methoden nicht implementiert haben. Was stellen Sie fest?

   ---

   c) *Für Schnelle: Python kann die anderen Vergleichsoperatoren selbstständig hinzufügen, indem Sie folgende Zeile vor Ihre Klassendefinition einfügen:*

```
1 import functools
2
3 @functools.total_ordering
4 class Haus:
```

   Testen Sie das neue Verhalten!

Weiterführende nützliche Links

- Einführung in *Magical Comparison Operators* in Python
  `https://www.pythonpool.com/python-__lt__/`