Dokumentation computer.py

H. Rabus, Institut für Mathematik
Humboldt-Universität zu Berlin

18.05.2018

Inhaltsverzeichnis

Bemerkung zu dieser Dokumentation

Die Klasse `Computer` dient zur Demonstration der objektorientierten Programmierung im Rahmen der VL **Einführung in das wissenschaftliche Rechnen**. Ebenso dient diese Schnittstellendokumentation als inhaltliches Beispiel - jedoch nicht als Formatvorlage oder gar Designvoraussetzung - wie diese in LaTeX umgesetzt werden kann.

Weitere denkbare Möglichkeiten zur Darstellung des Inhaltes sind

- Tabellenumgebungen, z.B.

Input	new_room (int) tba	the room this computer will be moved to NN
Returns	None	

  oder
- itemize/enumerate anstelle der description-Umgebungen (evtl. in der hier verwendeten kompakten Form (compactitem oder compactenum) aus dem Paket *paralist*.
- ...

Probieren sie verschiedene Varianten aus und bleiben dann innerhalb einer Dokumentation konsequent bei einer Variante.

In der Sprache haben Sie freie Wahl zwischen Englisch und Deutsch. Bitte bleiben Sie innerhalb eines Dokumentes konsequent bei einer Sprache. Alternativ dürfen Sie, wie in diesem Dokument, zwischen den Sprachen wechseln, wenn Sie dabei einen konsequenten Stil verfolgen (hier: Beschreibung von Inputparameter und Rückgabewerten in Englisch und so aus den *doc-Strings* übernommen, Rest deutsch).

1 Schnittstellendokumentation computer.py

Die Klasse ermöglicht die Computer mit verschiedenen Betriebssystemen in einem Gebäude zu verwalten.

1.1 Attribute

Bemerkung Bei der Aufzählung der Attribute (ebenso wie später bei den Methoden) muss zwischen dynamischen (oder nicht-statischen) Attributen (auch Objektattribute) und den statischen Attributen (auch Klassenattributen) unterschieden werden. Eine mögliche Variante ist die Folgende; eine andere wird bei den Methoden verwendet.

nicht-statische Attribute:

`sys` *(string)* The operating system, that is currently run on the computer. Should be one of the values 'MAC', 'WINDOWS' or 'LINUX'

`room` *(int)* The room in which the computer is currently located.

`state` *(bool)* The state of the computer.
False = The computer is turned off.
True = The computer is turned on.

statische Attribute:

`all_computers` *(list of computers)* A collection of all created computers.

1.2 Konstruktor

`Computer(self, operating_system, room)`
Beim Erstellen eines neuen Objektes der Klasse `Computer` wird *state = False* gesetzt, d.h. der Computer ist zunächst ausgeschaltet. Desweiteren werden die Attribute *operating_system* und *room* entsprechend der Eingabeparameter im Konstruktor gesetzt; die Liste *all_computers* um das neue Objekt ergänzt.

Input:

`operating_system` *(string)* The system, that the computer will be launched with. Should be one of the values 'MAC', 'WINDOWS' or 'LINUX'

`room` *(int)* The room the computer will be set up in.

Beispiel Im Folgenden wird beispielhaft der Quellcode des Konstruktors eingefügt.

```
def __init__(self, operating_system, room):
    self.sys = operating_system
    self.room = room
    self.state = False

    Computer.all_computers.append(self)
```

1.3 Methoden

Konvention - Bemerkung Das Objekt `self` wird bei nicht-statischen Methoden (und dem Konstruktor) nicht mit in die Beschreibung aufgenommen.

1.3.1 `change_room(self, new_room)`

Diese Methode ermöglicht die Zuordnung eines Computers zu einem neuen Raum.

Input:

`new_room` *(int)* the room this computer will be moved to
Returns: *None*

1.3.2 `toggle_state(self)`

Diese Methode ändert den Zustand (ein- vs. ausgeschaltet).
Input: -
Returns: *None*

1.3.3 `__str__(self)`

Methode, die die Ausgabe mittels `print()` auf die Standardausgabe regelt.
Input: -
Returns: *string* a representation of the object for the use with `print()`

Beispiel Für ein Objekt der Klasse `Computer` mit den Attributen `self.room = 123`, `self.sys = LINUX` und `self.state = True` liefert die Methode folgende Zeichenkette

```
Der Computer
    - steht im Raum 123
    - läuft mit Linux
    - ist derzeit eingeschaltet.
```

1.3.4 `static get_computers_in(room)`

Diese statische Methode liefert eine Aufstellung aller Computer, die sich in einem Raum befinden.
Input:
`room` *(int)* The room number
Returns:
(list of computers) All computers in room ‘room‘

2 Nutzungshinweise und Hauptprogramm

In `computer.py` ist eine `main()` Funktion implementiert. Sie dient zur Demonstration der Klasse und wird nur ausgeführt wenn `computer.py` direkt mittels `python3 computer.py` gestartet wird.

Es wird die Verwendung des Konstruktors und der Methoden anhand dreier Objekte demonstriert.

```
def main():
    """ Main function to test the Computer class.
    """

    cmp1 = Computer("LINUX", 1115)
    print(cmp1)
```

```
    cmp1.toggle_state()
    cmp1.change_room(2407)
    print(cmp1)

    print("\n==================\n")

    Computer("MAC", 2407)
    Computer("LINUX", 1115)

    my_computers = Computer.get_computers_in(2407)
    for computer in my_computers:
        print(computer)
```

Die Standardausgabe liefert hierdurch folgende Ausgabe:

```
Der Computer
- steht im Raum 1115.
- läuft mit Linux.
- ist derzeit ausgeschaltet.
Der Computer
- steht im Raum 2407.
- läuft mit Linux.
- ist derzeit eingeschaltet.

==================

Der Computer
- steht im Raum 2407.
- läuft mit Linux.
- ist derzeit eingeschaltet.
Der Computer
- steht im Raum 2407.
- läuft mit MAC-OS.
- ist derzeit ausgeschaltet.
```