Aufgabe 2 - Identität und Gleichheit

Ziel

Sie kennen den Unterschied zwischen Gleichheit (gleiche Klasse) und Identität (gleiche Referenz). Zudem können Sie den Inhalt zweier Objekte vergleichen.

Auftrag

1. Akzeptieren Sie das Assignment im GitHub Classroom.
2. Klonen Sie ihr Repository in die Entwicklungsumgebung.
3. Ergänzen Sie den Code der Klasse `ObjectIdentity` gemäss der Erklärung im Quellcode der Methode `__eq__`.
4. Führen Sie den Unit-Test aus. Er muss fehlerfrei ablaufen. Sollten Fehler auftreten, begrenzen sich diese auf die von Ihnen implementierte Methode `__eq__`.
5. Ergänzen Sie den Code in `main`. Beachten Sie insbesondere den letzten Teilauftrag, bei dem es um die Frage des „Inhalts“ der Objekte geht. Instanzieren Sie daher hier die beiden Objekte mit den gleichen Zeichenketten.
6. Führen Sie einen Commit und einen Push durch.

Dauer

30 Minuten + Hausaufgabenzeit

Abgabe

Die Abgabe erfolgt auf github.

—

Wichtig zu Wissen

In dieser Aufgabe werden 3 verschiedene Arten von „Gleichheit“ angewendet.

1. die Gleichheit der bzw. Zugehörigkeit zur Klasse
2. die Gleichheit der Objekt-Referenzen (eben die Identität)
3. die Gleicheit der Inhalte von Objekten

Die ersten beiden Fälle sind durch Keywords eindeutig erkennbar und in allen Fällen durch den Interpreter zu verstehen. Die Klassenzugehörigkeit wird durch `isinstance` geregelt, während die Identität durch `is` überprüft wird.
Bei der inhaltlichen Gleichheit wird das ganze etwas komplexer. Bei einfachen Datentypen (int, float, char, boolean) kann der Interpreter die Anweisung

```
value = 7
```

```
 ...
if value == 77:
 ...
```

umsetzen, da klar ist, um welchen Datentyp es sich handelt. Das gilt auch für einzelne Zeichen...

```
if sign == 'a':
```

... und den Vergleich von Attributen (bzw. Attributswerte) die einen einfachen DAtentyp repräsentieren.

```
val1 = 99
val2 = 33
...
if val1 > val2:
...
```

Bei einer komplexen Klasse mit verschiedenen Attributen kann der Interpreter den ==-Befehl aber nicht mehr ausführen. Hier braucht es eine (implizite) Methode, die den Vergleich korrekt bewerkstelligt.
Jede Klasse verfügt über eine (verborgene) Methode __eq__ (für equal). Diese muss im Sinne des Vergleichs der Attribute entsprechend programmiert werden. Die Methode muss dabei zwei Aspekte abdecken:
Es muss

1. die (inhaltliche) Gleichheit zweier Objekte sowie
2. ein Vergleich mit einem konstanten Wert

sichergestellt sein.

Ein Beispiel dazu.
Eine Klasse Person weist die Attribute Name und Vorname auf. Es wird festgelegt, dass bei einem Vergleich mit == auch eine Konstante der Art „Vorname Nachname“ als gleich erkannt werden soll. Der Einfachheit halber deklarieren wir hier eine Datenklasse.

```
from dataclasses import dataclass
from dataclasses import dataclass
@dataclass
class Person:
    # die Attribute der Datenklasse
    first_name: str = None
    last_name: str = None

    # die Methode die den ==-Befehl korrekt umsetzt.
    def __eq__(self, other_object):
        # zuerst prüfen, ob das Objekt überhaupt "passt".
        if isinstance(other_object, Person):
```

```
            return self.first_name == other_object.first_name and
self.last_name == other_object.first_name
        # danach prüfen, ob das evtl. ein konstanter Wert vorliegt, der zu
prüfen ist.
        elif isinstance(other_object, str):
            return self.first_name + ' ' + self.last_name == other_object
        else:
            return False



if __name__ == '__main__':
    carla = Person('Carla', 'Meier')
    theo = Person('Theo', 'Meier')

    if carla == theo:
      print(f'Objekt carla und theo haben den gleichen Inhalt')
    else:
      print(f'Objekt carla und theo haben unterschiedlichen Inhalt')
    if carla == "Carla Meier":
      print(f'Objekt carla weist den Inhalt "Carla Meier" auf')
```
