LU11b - Abstrakte Klassen in Python

Standardmässig stellt Python keine abstrakten Klassen bereit. Das Modul `abc` stellt die Basis für die Definition von Abstract Base Classes ABC bereitstellt. ABC funktioniert, indem Methoden der Basisklasse als "abstrakt dekoriert" und dann konkrete Klassen als Implementierungen der abstrakten Basis registriert werden. Eine Methode wird abstrakt, wenn sie mit dem Schlüsselwort `@abstractmethod` dekoriert wird. Zum Beispiel so:

Beispiel: Abstrakte Klasse für Vielecke (Polygon)

```
from abc import ABC, abstractmethod

class Polygon(ABC):
    """
    Ein unbestimmtes Polygon, das
    a) weiss, dass es eine gewisse Anzahl Seiten hat
    b) aber nicht weiss, wie viele es wirklich sind.
    """

    def __init__(self):
        """
        Konstruktor: Könnte auch weggelassen werden, solange keine Attribute
gesetzt werden.
        """
        pass


    @abstractmethod
    def my_sides(self):
        """
        Abstrakte Methode ohne Implementierung. Diese muss
        zwingend in einer abgeleiteten Klasse erfolgen.
        """
        pass

class Triangle(Polygon):
    # overriding abstract method
    def my_sides(self):
        print('I have 3 sides')

class Quadrilateral(Polygon):
    # overriding abstract method
    def my_sides(self):
        print('I have 4 sides')


if __name__ == '__main__':
    polygons = [
```

```
        Triangle(),
        Quadrilateral(),
    ]
    for p in polygons:
        p.my_sides()
```

Output:

```
I have 3 sides
I have 4 sides
```

Beachten Sie hier die Nutzung der Polymorphie, um gleichartige Objekte (Triangle, Quadrilateral) gleich zu behandeln - konkret durch den Aufruf der (überschriebenen) Methode `my_sides`.
