Aufgabe 1

(a) Ein Bauplan für Objekte (ein abstrakter Datentyp)

(b) Ein Objekt, das zur Laufzeit aus einer Klasse erzeugt wird

(c) Eine Variable, die zu einem Objekt gehört

(d) Eine Funktion, die zu einem Objekt gehört

(e) Eine Variable, die zu einer Klasse gehört

(f) Eine Funktion, die zu einer Klasse gehört

(g) Eine spezielle Methode, mit der ein Objekt initialisiert wird.

Aufgabe 2

```
12
```

Aufgabe 3

```
(4, 2)
```

Aufgabe 4

```
1.0
```

Aufgabe 5

```
25
```

Aufgabe 6

```
class Lampe:

    def __init__(self):
        self.zustand = False

    def __str__(self):
        if self.zustand == True:
            return 'Lampe an'
        else:
            return 'Lampe aus'

    def schalten(self):
        if self.zustand == False:
            self.zustand = True
        else:
            self.zustand = False

# Test-Client: (nur zur Illustration)
L = Lampe()  # erzeugt einen neue (ausgeschaltete) Lampe
L.schalten() # Schaltet die Lampe ein
L.schalten() # Schaltet die Lampe aus
L.schalten() # Schaltet die Lampe ein
print(L)     # => 'Lampe ein'
```

Aufgabe 7

Hier sind vier Vorteile:

(a) Die Objektorientierung ist an die Sichtweise der Menschen angelehnt. Sie schafft eine Abstraktion, durch die wir Programme besser verstehen und einfacher programmieren können.

(b) Durch Vererbung kann Code in anderen Klassen wiederverwendet werden, was effizient ist.

(c) Polymorphie („Vielgestaltigkeit“): Variablen- und Funktionen können in unterschiedlichen Klassen den gleichen Namen haben, ohne das Konflikte enstehen.

(d) OOP unterstützt *Information Hiding*. Das bedeutet, dass Änderungen an den Objekten nur über dafür vorgesehene Methoden (sog. Schnittstellen) möglich sind. Objekte