Aufgabe 1. Implementiere eine Klasse `Polynom`, welche ein Polynom mit reellen Koeffizienten modelliert. Die Koeffizienten des Polynoms sollen dabei dem Konstruktor übergeben werden und innerhalb der Klasse als Tupel gespeichert werden. Implementiere die grundlegenden Rechenfunktionen auf Polynomen, wie Addition, Substraktion und Multiplikation, sowie die Multiplikation eines Polynoms mit einem `float`. Außerdem sollen Polynome eine Gradfunktion besitzen und als String interpretiert werden können.

Mache dabei intensiv Gebrauch von den speziellen Memberfunktion, die du in der Vorlesung kennengelernt hast und die im Skript genauer besprochen werden.

Aufgabe 2. Schreibe eine Funktion, welches folgendes Verzeichnis `d` erzeugt. Der Schlüssel ist eine Zahl zwischen 2 und 12 und der zugehörige Wert ist eine Liste von Tupeln (oder Mengen wenn dir das besser gefällt) deren Einträge die möglichen Würfelaugenpaarkombinationen sind, deren Summe den Schlüssel ergibt. Zum Beispiel ist der Eintrag zum Schlüssel 3 gegeben durch

```
1 print(type(d[3]), d[3])) #Druckt: <class 'list'> [(1,2),(2,1)]
```

oder eben

```
1 print(type(d[3]), d[3])) #Druckt: <class 'list'> [{1,2}]
```

Aufgabe 3. Gegeben ein Verzeichnis dessen Schlüssel und Werte Strings sind, sodass jeder Wert höchstens einmal vorkommt. Erstelle daraus mithilfe von "Multiple Assignements" und "List Comprehension" ein Verzeichnis dessen Schlüssel-Wert-Paare umgekehrt sind. Schaffst du das in einem einzigen Ausdruck?

Aufgabe 4. In dieser Aufgabe wollen wir Klassen als Enten interpretieren, wenn sie sich wie Enten verhalten. Dazu verwenden wir die Klasse `Ente`:

```
1 class Ente:
2   """Eine einfache Entenklasse"""
3   def __init__(self, quaks=1):
4     """Erstellt eine Ente.
5        Die Anzahl der quaks kann uebergeben werden."""
6     self.quaks = quaks
7
8   def __str__(self):
9     return self.quaks * 'Quack!'
```

Erweiter die Klasse `Polynom` um die spezielle Memberfunktion `__ente__`, die eine Ente zurückgibt, wobei die Anzahl der Quaks dem Grad des Polynoms entsprechen soll.

Schreibe eine Funktion `ente(arg)`, die sich genau wie die bekannten Funktionen `str(arg)` oder `int(arg)` verhält. Das heißt, folgender Code soll interpretierbar sein:

```
1 p = Polynom(...) # Erstellt Polynom p vom Grad n
2 e = ente(p)      # Interpretiert p als Ente
3 print(e)         # Druckt: Quack!Quack!...Quack! (n mal)
```

Aufgabe 5. Implementiere eine Klasse `Telefonbuch`. Ein `Telefonbuch` soll ein Verzeichnis von Einträgen sein, in dem zu einem gegebenen Vor- und Nachnamen als Schlüssel der Eintrag die Telefonnummer ist. Die Klasse soll Funktionen zum Hinzufügen bzw. Löschen eines Namens bereitstellen. Ferner soll es eine Funktion geben, welche zu einem gegebenen Vor- und Nachnamen nachschaut, ob dieser Name im Telefonbuch steht und die zugehörige Nummer zurückgibt. Steht der Name nicht im Telefonbuch, so soll `False` zurückgegeben werden. Außerdem soll das Telefonbuch nach Vor- oder Nachnamen sortiert und ausgedruckt werden können.

Aufgabe 6. Implementiere die Klassen `Matrix` und `Vektor`, die Matrizen bzw. Vektoren modellieren. Die Einträge einer Matrix/eines Vektors sollen mit Hilfe der eckigen Klammern `[]` ausgelesen und gesetzt werden können. Implementiere die grundlegenden Rechenarten mit speziellen Funktionen und insbesondere die Matrix-Matrix-Multiplikation und die Matrix-Vektor-Multiplikation. Für zwei Vektoren `A` und `B` soll das Produkt `A*B` das Skalarprodukt der beiden Vektoren sein.