5. Rechnerübung: Klassen und Objekte

Aufgabe 1: Die Klasse Hase

Betrachten Sie das folgende Programm. Es enthält die Definition einer Klasse `Hase`. Dann werden 5 Objekte `h1` bis `h5` vom Type `Hase` erzeugt und ihren Attributen Werte zugewiesen. Im Folgenden werden Sie in einzelnen Aufgaben das Programm um verschiedene Funktionalitäten erweitern.

In [ ]:
```
class Hase:
    Name = "?"
    Groesse = 0
    Gewicht = 0
    BesterFreund = None
    def wachse(self,um):
        self.Groesse = self.Groesse+um

h1 = Hase(); h1.Groesse=10; h1.Gewicht=234; h1.Name = "Hoppel";
h2 = Hase(); h2.Groesse=12; h2.Gewicht=210; h2.Name = "Smak";
h3 = Hase(); h3.Groesse=17; h3.Gewicht=333; h3.Name = "Hopp";
h4 = Hase(); h4.Groesse=15; h4.Gewicht=272; h4.Name = "Puschel";
h5 = Hase(); h5.Groesse=13; h5.Gewicht=250; h5.Name = "Baukel";

h1.BesterFreund = h5;
h2.BesterFreund = h3;
h3.BesterFreund = h2;
h4.BesterFreund = h1;
h5.BesterFreund = h4;
```

a) Attribute anzeigen

Geben Sie in der folgenden Zelle ein paar `print`-Anweisungen ein um die Objekte zu untersuchen. Probieren Sie mal `print(h1)`. Es erscheint etwas kryptisches. Python kann keine ganzen Objekte anzeigen. Nach ein bisschen Spielen, finden Sie sicher heraus, wie das Argument bzw. die Argumente für die Printanweisung aussehen müssen, damit auf dem Bildschirm der Satz: `Hoppel ist 10 cm gross und wiegt 234 Gramm.` erscheint.

In [ ]:
```

```

b) Hasenliste erzeugen

Erweitern Sie das Programm so, dass Sie eine Liste `AlleHasen` anlegen, in der (wie der Name schon sagt) alle 5 Hasen aufgeführt werden, so dass dann `AlleHasen[0]` das Objekt `h1` ist und `AlleHasen[1]` das Objekt `h2` ist und so weiter. Schreiben Sie dann eine neue Funktion mit Namen `AlleNamen` zum Anzeigen aller Hasennamen, die als Argument eine Liste erwartet, und rufen Sie

diese Funktion mit der gerade angelegten Liste als Argument auf, so dass dann auf dem Bildschirm alle Hasennamen ausgedruckt werden. Die Ausgabe soll dann

```
Hoppel
Smak
Hopp
Puschel
Baukel
```

erscheinen.

In [ ]:

c) Hasenfreunde anzeigen

Schreiben Sie nun eine Funktion namens `AlleFreunde`,, die auch als Argument eine Liste erwartet und auf dem Bildschirm für jeden Hasen in der Argumentliste ausgibt wer dessen Freund ist. Die Ausgabe nach dem Aufruf der Funktion mit dem Argument `AlleHasen` soll dann so aussehen:

```
Der beste Freund von Hoppel ist Baukel
Der beste Freund von Smak ist Hopp
Der beste Freund von Hopp ist Smak
Der beste Freund von Puschel ist Hoppel
Der beste Freund von Baukel ist Puschel
```

In [ ]:

d) Hasenpaare anzeigen

Und jetzt basteln Sie eine Funktion `AllePaare`, die auf dem Bildschirm alle Paare ausgibt, das heißt alle Hasenpaare, die sich gegenseitig der beste Freund sind. Auch diese Funktion erwartet eine Liste von Hasenobjekten auf denen Sie ihre Arbeit verrichtet. Die Ausgabe muss dann so aussehen:

```
Smak und Hopp sind ein Paar
```

Alle Paare (naja, wir haben halt nur eines) sollen aber nur einmal ausgegeben werden, also nicht auch nochmal in der anderen Richtung.

In [ ]:

e) Den kleinsten Hasen finden

Schreiben Sie nun eine Funktion `FindeKleinsten`, die ebenso wie die anderen bisherigen Funktionen eine Liste von Hasenobjekten kriegt und aus dieser Liste das Objekt, das die kleinste Größe hat sucht, und dies dann auf dem Bildschirm anzeigt. In unserem Fall muss die Ausgabe dann aussehen wie:

```
Der kleinste ist Hoppel
```

In [ ]:

f) Hasen wachsen lassen

Schreiben Sie eine Funktion mit Namen `AlleWachsen`, die auch eine Hasenliste als Argument kriegt und unter der Verwendung der schon definierten Funktion `wachse(...)` dafür sorgt, dass alle Hasen in der Argumentliste um 1cm wachsen, also ihr Attribut `Groesse` umd 1 erhöhen. Geben Sie zum Test der Funktion die Größen vor und nach dem Aufruf auf dem Bildschirm aus.

In [ ]:

g) Neue Objekt-Operation

Definieren Sie die Klasse Hase erneut und erweitern Sie sie um eine weitere Objekt-Operation namens `friss`, die als Argument eine Zahl erwartet und dafür sorgt, dass das Objekt, für das sie aufgerufen wurde sein Gewicht vergrößert, namlich um den Wert des Arguments.

In [ ]:
```
class Hase:
    Name = "?"
    Groesse = 0
    Gewicht = 0
    BesterFreund = None
    def wachse(self,um):
        self.Groesse = self.Groesse+um

    # Hier kommt Ihr Code für die Funktion friss(...) hin


h1 = Hase(); h1.Groesse=10; h1.Gewicht=234; h1.Name = "Hoppel";
h2 = Hase(); h2.Groesse=12; h2.Gewicht=210; h2.Name = "Smak";
h3 = Hase(); h3.Groesse=17; h3.Gewicht=333; h3.Name = "Hopp";
h4 = Hase(); h4.Groesse=15; h4.Gewicht=272; h4.Name = "Puschel";
h5 = Hase(); h5.Groesse=13; h5.Gewicht=250; h5.Name = "Baukel";

h1.BesterFreund = h5;
h2.BesterFreund = h3;
h3.BesterFreund = h2;
h4.BesterFreund = h1;
h5.BesterFreund = h4;

print(h4.Gewicht)   # Ausgabe müsste 272 sein
h4.friss(13)
print(h4.Gewicht)   # Ausgabe müsste 285 sein
```

Aufgabe 2: Hochschulverwaltung

Das folgende Programm stellt eine Verwaltungssoftware der Hasen-Uni in Karlsruhe dar. Ein paar Funktionen sind unvollständig. Ersetzen Sie die ausgeixten Stellen durch

richtigen Python code, so dass die Ausgabe des Programms wie folgt aussieht:

```
Name:      Hoppla
Matr.-Nr.: 84
Noten:     Hüpfen 2.3 Trommeln 3.7

Noch nichts geprüft haben:
Hoppel
Puschel
```

Dazu müssen Sie das Programm studieren, erkennen und verstehen, welche Teile wofür zuständig sind, und sich an der obigen Ausgabe orientieren, Sie kriegen das hin - früher oder später.

In [ ]:
```
class Student:
    def toString(self):
        # VERVOLLSTÄNDIGEN SIE HIER DIE FUNKTION toString DAMIT SIE EINE
        # SCHÖNE ANZEIGE EINES STUDENTEN ALS STRING ZURÜCK LIEFERT, SO WI
        # IN DEN ERSTEN DREI ZEILEN DER AUSGABE IN DER AUFGABENSTELLUNG.

class Note:
    def toString(self):
        return self.Fach+" "+str(self.Bewertung)

class Hochschule:
    def getStudent(self,MatrNr):
        # VERVOLLSTÄNDIGEN SIE DIESE FUNKTION DAMIT SIE EIN Student-OBJEK
        # ZURÜCK LIEFERT, DESSEN MATRIKEL NUMMER GLEICH DEM GEGEBENEN PAR

    def getSchieber(self):
        s = []
        # FÜLLEN SIE HIER DIE LISTE S, SO DASS SIE ALLE Student-OBJEKTE E
        # DEREN NOTENLISTE LEER IST.
        return s

HKA = Hochschule()

n1 = Note(); n1.Fach = "Nagen";    n1.Bewertung = 1.0
n2 = Note(); n2.Fach = "Hüpfen";   n2.Bewertung = 2.3
n3 = Note(); n3.Fach = "Trommeln"; n3.Bewertung = 3.7

s1 = Student(); s1.Name = "Baukel";  s1.MatrNr = 42; s1.Noten=[n1]
s2 = Student(); s2.Name = "Hoppel";  s2.MatrNr = 21; s2.Noten=[]
s3 = Student(); s3.Name = "Hoppla";  s3.MatrNr = 84; s3.Noten=[n2,n3]
s4 = Student(); s4.Name = "Puschel"; s4.MatrNr = 63; s4.Noten=[]

HKA.Studenten = [s1,s2,s3,s4]

print(HKA.getStudent(84).toString())
print()

Schieber = HKA.getSchieber()
print("Noch nichts geprüft haben:")
for sch in Schieber:
    print(sch.Name)
```