ALP II
SS 2012
Prof. Dr. Margarita Esponda

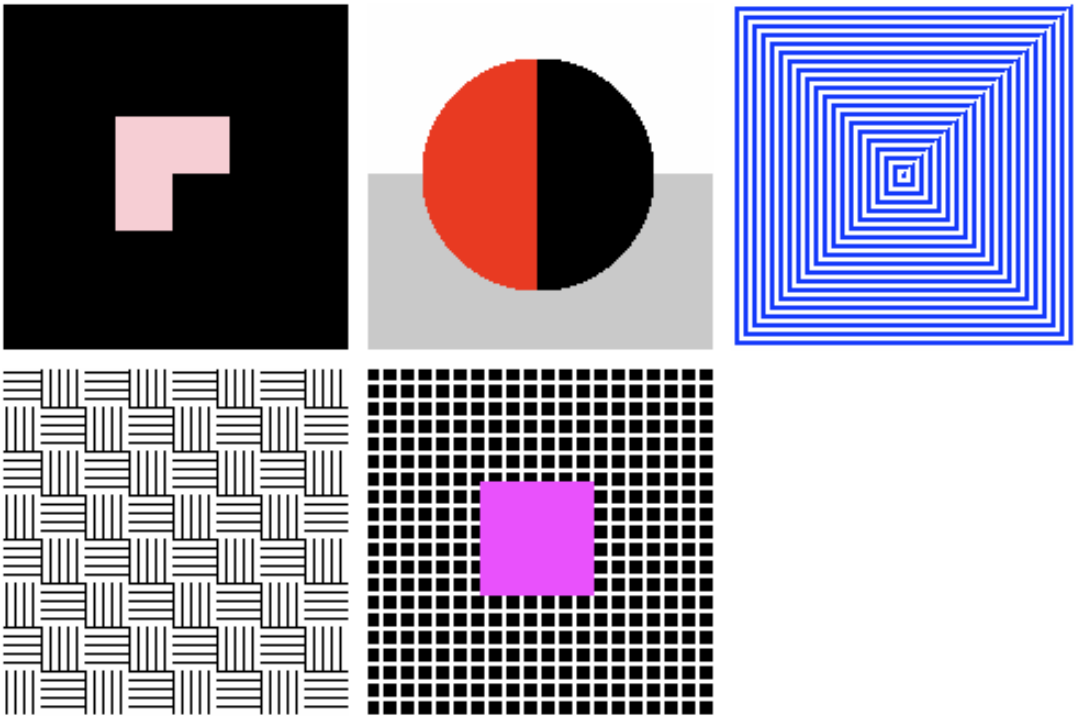
2. Übungsblatt

Ziel: Auseinandersetzung mit logischen Ausdrücken, höheren Datenstrukturen, Schleifen und Funktionen in Python.

1. Aufgabe (6 Punkte)

a) In dieser Aufgabe sollen drei der folgenden fünf Funktionen definiert werden, die die unten stehenden Mosaiken erzeugen.

"decide_color_rects"
"decide_color_circle"
"decide_color_squares"
"decide_color_illusion"
"decide_color_illusion_square"



Die "**decide_color**"-Funktionen bekommen als Argumente immer eine **x**, **y** - Koordinate und einen **size**-Wert, der der Seitenlänge des Bildes entspricht und entscheiden, welche Farbe zurückgegeben wird. Ein Programmierrahmen, um die Graphik zu erstellen, wird zur Verfügung gestellt.

Siehe: http://www.inf.fu-berlin.de/lehre/SS12/ALP2/homework.htm

b) Zwei Bonuspunkte können erworben werden, wenn alle fünf Funktionen definiert werden.

c) Ein oder zwei Zusatzpunkte können erzielt werden, wenn interessante

Bilder innerhalb der **decide_color_own**1 und **decide_color_own**2 Funktionen produziert werden, die geplant gewesen und nicht aus Versehen entstanden sind.

2. Aufgabe (3 Punkte)

Schreiben Sie eine Python-Funktion, die eine per Zufall generierte Zeichensequenz mit nur Nullen und Einsen als Eingabe bekommt und Anfang und Länge der längsten Folge von nebeneinander stehenden Einsen innerhalb der Sequenz als Tupel zurückgibt.

Beispiel:

```
>>> largest_ones_seq( [0,0,1,1,0,0,1,0,0,1,1,1,1,0,0,1,1,0,0,1,0] )
>>>  (9, 4)
```

3. Aufgabe (4 Punkte)

a) Schreiben Sie in Python eine rekursive Funktion **change_money**, die bei Angabe einer Geldmenge das optimale Wechselgeld (minimale Anzahl von Münzen) für diese Geldmenge berechnet und in einer Liste zurückgibt. Sie können eine konstante Hilfsliste [200, 100, 50, 20, 10, 5, 2, 1] dafür verwenden, die die Liste aller verschiedenen Münzsorten darstellt.

b) Schreiben Sie eine iterative Version der **change**-Funktion.

4. Aufgabe (7 Punkte)

a) Schreiben Sie ein Python-Programm, dass das Drei-Türen-Spiel (Monty-Hall-Spiel) simuliert.

Spielregeln:

1. Zwei Ziegen und ein Auto werden zufällig hinter drei Tore platziert, die geschlossen sind.
2. Der Spieler wählt ein Tor aus, das aber vorerst verschlossen bleibt.
3. Der Moderator öffnet von den zwei nicht gewählten Toren das Tor, hinter dem sich kein Auto befindet und fragt den Kandidaten, ob er bei seiner Wahl bleibt.
4. Der Spieler entscheidet, ob er seine Entscheidung ändert oder nicht.

b) Simulieren Sie mit Hilfe von Zufallszahlen das Spiel und berechnen Sie, wie oft der Spieler bei 10000 Versuchen gewinnt,

1. wenn er seine Entscheidung nie ändert.
2. wenn er jedes Mal zufällig seine Entscheidung ändert.
3. wenn er seine Entscheidung immer ändert.

5. Aufgabe (3 Punkte)

Nehmen Sie an, Sie haben ein Bild, in dem jedes Pixel mit seinem RGB-Wert dargestellt wird. Wir wissen, dass in dem Bild maximal nur 500 verschiedene Farben vorkommen können. Programmieren Sie einen möglichst effizienten Algorithmus, der mit minimalem Speicherverbrauch die Tabelle der Farbfrequenzen aus den vorhandenen Farben berechnet.

Wichtige Hinweise:

1) Verwenden Sie geeignete Namen für Ihre Variablen und Funktionsnamen, die den semantischen Inhalt der Variablen oder die Funktionalität der Funktionen darstellen.

2) Verwenden Sie vorgegebene Funktionsnamen, falls diese angegeben werden.

3) Kommentieren Sie Ihre Programme.

4) Verwenden Sie geeignete Hilfsvariablen und Hilfsfunktionen in Ihren Programmen.

5) Löschen Sie alle Programmzeilen und Variablen, die nicht verwendet werden.

6) Schreiben Sie getrennte Test-Funktionen für alle 5 Aufgaben für Ihren Tutor.