# Aufgaben für die 13. Übung zur LV "Grundlagen der Informatik"
## *Thema: Python, Teil 8 (Bildbearbeitung mit JES / Zwei Aufgaben mit IDLE)*

---

*In der 13. Übung wiederholen wir Inhalte der bisherigen Übungen, nämlich die Bildbearbeitung mit JES sowie rekursive Algorithmen und Schleifenstrukturen mit der Python-IDLE.*

**********

**→ *Im 1.Teil der 13. Übung benutzen wir JES***

**********

## 26) Bildbearbeitung mit JES: Umwandeln eines Farbpositivs in ein Farbnegativ bzw. (mit dem gleichen Programm!) eines Farbnegativs in ein Farbpositiv *(mit JES)*

*(Wiederholung zur Bildmanipulation mit JES)*

## a) Kurze Vorbemerkung

In der 5. Vorlesung wurde eine sehr ähnliche Aufgabe mit JES gelöst (siehe 5.Vorlesung in Vorbereitung der Übung).

**Hinweis zur Lösung:** Die RGB-Werte eines digitalen Bildes liegen bekanntlich im Bereich [0, 255]. Um ein Farbpositiv in ein Farbnegativ bzw. (identisches Vorgehen, da Umkehrung) ein Farbnegativ in ein Farbpositiv umzuwandeln, muss man **den Farbwert jedes Pixel** des Negativs bzw. Positivs überschreiben durch den „inversen“ Farbwert:

(255-Rotwert, 255-Grünwert, 255-Blauwert)

## b) JES-Programm

Schreiben Sie ein vollständiges JES-Programm zur Umwandlung eines Farbpositivs in ein Farbnegativ bzw. (mit dem gleichen Programm) eines Farbnegativs in ein Farbpositiv.

Wählen Sie beim Test des Programms aus dem bekannten Bildverzeichnis die JPG-Bilder *daisies_Negativ.jpg* bzw. *daisies_Positiv.jpg* sowie *twoSwans_Negativ.jpg* bzw. *twoSwans_Positiv.jpg*.

Gehen Sie im Programm wie folgt vor:
- Beginnen Sie den Quelltext mit einer gesonderten Funktion zur Umwandlung eines Farbpositivs in ein Farbnegativ bzw. eines Farbnegativs in ein Farbpositiv
- Im Hauptprogramm sind folgende Anweisungen ins Programm einzugeben:
  - -- Name des Farbnegativs bzw. Farbpositivs (per Menüauswahl) einem Namen zuweisen
  - -- Erzeugen einer internen Datenstruktur (Matrix aus Pixeln), d.h. eines “Bildobjekts“
  - -- Das Ausgangsbild vom Programm aus anzeigen
  - -- Aufruf der o.g. Funktion zur Umwandlung eines Farbpositivs in ein Farbnegativ bzw. eines Farbnegativs in ein Farbpositiv (Bildobjekt übergeben)
  - -- Nach Rückkehr aus der Funktion das umgewandelte Bild anzeigen
  - -- schließlich das umgewandelte Bild unter dem Bezeichner *InversesBild.jpg* nach *W:\Bilder* schreiben

*(ohne Startfile).*

---

**********

***→ Im 2.Teil der 13. Übung benutzen wir wieder die Python-IDLE***

**********

## 27) Rekursive Berechnung der Summe von n reellen Messwerten *(mit IDLE)*

***(Wiederholung zur Rekursion)***

## a) Vorbemerkung:

In der 6.Vorlesung, Seite 7-8, und auch schon in der 2.Vorlesung, Seite 9, wurden rekursive Algorithmen (jeweils die rekursive Berechnung der Fakultät) behandelt.
In der 7. Übung, Aufgabe 7, berechneten wir ebenfalls die Fakultät mit einem rekursiven Algorithmus.
In diesem Zusammenhang ist auch auf die Datei *folie_rekursive_fakultaet.doc* im Verzeichnis *Y:\lehre\LV_Gr_d_Inf_WS0607\Lehrfolien_zur_Uebung* hinzuweisen.
**Wiederholen Sie diese Quellen in Vorbereitung der 13. Übung.**

Wesen eines rekursiven Algorithmus ist es, dass die Aufgabe solange auf immer einfachere Versionen der Aufgabe zurückgeführt wird, bis die Aufgabe schließlich erstmals lösbar ist. Von diesem „Rekursionsanfang“ aus wird nun die Lösung der ursprünglichen Aufgabe aufgebaut. Beim jetzt beginnenden „Rekursionsaufstieg“ werden schrittweise rekursiv die erhaltenen Sublösungen eingesetzt und jedes Unterprogramm (Funktion) bis zum Ende abgearbeitet (zur graphischen Erläuterung dieser Ausführung siehe o.g. Datei *folie_rekursive_fakultaet.doc*).
Beispiel: Bei der rekursiven Berechnung der Fakultät n! führte dieser Weg auf:

n_fakultaet = n*(n-1)! (n>=0, ganz),

wobei (n-1)! die oben genannte einfachere Version der Aufgabe ist. Schließlich ist die Aufgabe für n=0 erstmals lösbar, da *per definitionem* gilt: 0! = 1

---

## b) Programm zur rekursiven Berechnung der Summe von n reellen Messwerten *(mit IDLE)*

***(Einfachster rekursiver Algorithmus)***

Schreiben Sie ein vollständiges Programm zur **rekursiven** Berechnung der Summe von n reellen Messwerten.

Im Hauptprogramm sollen zunächst die n reellen Messwerte; n>=2; eingelesen werden (n sei bekannt). Danach ist die rekursive Funktion zur Berechnung der Summe aufzurufen. Das Resultat (Summe) ist im Hauptprogramm auszugeben.

In der Funktion ist nicht mit globalen, sondern nur mit lokalen Variablen zu arbeiten.

---

## 28) Berechnung der Quersumme einer natürlichen Zahl sowie der Summe von Quersummen natürlicher Zahlen *(mit IDLE)*

***(wenn die Zeit nicht reicht, ist der Rest Hausaufgabe)***

### a) Berechnung der Quersumme *einer* natürlichen Zahl

Schreiben Sie ein vollständiges Python-Programm für die Berechnung der Quersumme einer natürlichen Zahl (natZahl = 0, 1, 2, 3, …).
Lesen Sie dazu die natürliche Zahl im Hauptprogramm ein. Rufen Sie dann eine Funktion zur Berechnung der Quersumme auf. Das Resultat (Quersumme) soll im aufrufenden Hauptprogramm ausgegeben werden.
Fordern Sie den Nutzer nach jeder erfolgten Berechnung einer Quersumme zur Eingabe einer nächsten natürlichen Zahl auf. Ein Abbruch der Berechnung soll dabei durch Eingabe einer (beliebigen) negativen Zahl (als „natürliche Zahl“) erreicht werden.
- **Beispiel** zur Erklärung der Quersumme: quersumme(4233) = 4+2+3+3 = 12

**- Hinweis zum Lösungsalgorithmus:**
Schreiben Sie **zwei Versionen** zur Berechnung der Quersumme:
- in einer *ersten Version* mit Konvertierung der eingelesenen natürlichen Zahl in einen String, zeichenweisen Zugriff auf die „Zifferbuchstaben“ und Konvertierung der „Zifferbuchstaben“ in eine ganze Zahl
- und in einer *zweiten Version* mit sukzessiver Verwendung des Divisionsrests der (ganzzahligen) Division durch 10

(Das Hauptprogramm ist in beiden Versionen gleich.)
---

### b) Erweiterung: Summe von Quersummen natürlicher Zahlen *(mit IDLE)*

**Erweitern** Sie die **erste** Version der Lösung von 28a) wie folgt:
Jetzt soll nicht die Quersumme ***einer*** natürlichen Zahl berechnet werden, sondern die **Summe der Quersummen *mehrerer*** eingegebener natürlicher Zahlen. Eine Wiederholung der Berechnung (während eines Programmlaufs) wie in 28.a) soll hier aus Einfachheitsgründen nicht vorgesehen werden.
**Beispiel:** Eingegeben wurde: [34, 567, 700] → Summe der 3 Quersummen = 32

**Hinweise zur Lösung:**
→ Verwenden Sie als **Startfile** das unvollständige Python-Programm *summeVonQuersummen_Aufg28b_Startfile.py* aus dem bekannten Übungsverzeichnis *Ueb13_Python8*.
In diesem Startfile erfolgt die Berechnung der Summe der Quersummen in folgender Weise:
- Im Hauptprogramm werden die natürlichen Zahlen eingegeben (Abbruch: beliebige negative Zahl) und in einer *Liste* natürlicher Zahlen gespeichert.
- Dann wird eine Funktion zur Berechnung der Summe der Quersummen aufgerufen. In *dieser* Funktion (!!!) wird innerhalb der Anweisung zur Summenberechnung jeweils eine Funktion zur Berechnung der Quersumme einer natürlichen Zahl aufgerufen (und zwar genau die Funktion zur Berechnung der Quersumme aus der **ersten** Lösungsversion von 28.a).
*(Hier wird also erstmals in unserer Übung eine Funktion von einer anderen Funktion aus aufgerufen. Natürlich muss die aufgerufene Funktion im Quelltext vor der aufrufenden Funktion stehen, da ihr Funktionskopf dem Interpreter zum Aufrufzeitpunkt bekannt sein muss.)*
- Die Lösung wird schließlich wieder im Hauptprogramm ausgegeben.

**→ Ergänzen Sie das Startfile an den gekennzeichneten Stellen, und testen Sie das Programm.**

**Ausdruck des Startfiles** *summeVonQuersummen_Aufg28b_Startfile.py* aus *Ueb13_Python8* :

```
# -*- coding: cp1252 -*-
#Summe von Quersummen natürlicher Zahlen (unvollständiges Startfile)
#(Erweiterung von: Quersumme einer natürlichen Zahl)
#####################################################################
def querSum(natZahl): #Stimmt überein mit Funktion aus 28.a), Version 1
    zahlString=str(natZahl)
    querSumme=0
    for zifferBuchstabe in zahlString:
        querSumme=querSumme+int(zifferBuchstabe)
    return querSumme
#####################################################################
def summeVonQuersummen(SELBST EINFUEGEN):
    sum=SELBST EINFUEGEN
    for SELBST EINFUEGEN:
        sum=sum+SELBST EINFUEGEN
    return SELBST EINFUEGEN
#####################################################################
#Hauptprogramm:
print "Berechnung der Summe von Quersummen mehrerer natürlicher Zahlen:"
print "-------------------------------------------------------------\n"
natZahlList=[]
natZahl=input("1-te natürliche Zahl (Abbruch: negative Zahl) eingeben: ")
i=1
while natZahl>=0:
    natZahlList.SELBST EINFUEGEN
    i=SELBST EINFUEGEN
    print "%d-te natürliche Zahl (Abbruch: negative Zahl) eingeben:" %i,
    natZahl=input()
print "\nSumme der Quersummen aller Zahlen ---> %i" \
      %SELBST AUFRUF EINFÜGEN
#####################################################################
```