Aufgaben für die 6. Übung zur LV "Grundlagen der Informatik"
Thema: Programmieren mit Python / JES, Teil 1

---

0) Vorbemerkung zu den benutzten Entwicklungsumgebungen

0.1 Python Standard Software:

Homepage: **http://www.python.org**
Dokumentationen: **http://www.python.org/doc/**
Download der z.Zt. aktuellen Version 2.5 (Freeware): **http://www.python.org/download/**
Hier kann man z.B. den **Python 2.5 Windows installer** *python-2.5.msi* laden, mit dessen Hilfe man ohne Probleme Python installieren kann.
Hinweis: Das aktuelle Installationsfile *python-2.5.msi* liegt auch auf der Festplatte unter *Y:\lehre\LV_Gr_d_Inf_WS0607\Installfiles_JES_2_und_Python_2_5* .
Nach der Installation auf einem Windows-PC findet man über *Start/All Programs/Python 2.5* u.a.:
- IDLE (Python GUI) … **I**ntegrated **D**eve**L**opment **E**nvironment for Python
(außerdem: Eric IDLE … Komödiant aus “Monty Python’s Flying Circus“)
- Python (command line) … Kommandozeilen-Interpreter für Python

→ *Die IDLE werden wir in den Übungen zusammen mit der JES-Oberfläche (siehe 0.2) nutzen.*

0.2 Jython Environment for Students (JES)

Die Entwicklungsumgebung JES (Freeware) wurde am *College of Computing* des *Georgia Institute of Technology* in den USA unter Leitung von Prof. Guzdial entwickelt.

- Download von JES: http://coweb.cc.gatech.edu/cs1315/3569
→ aktuelle Windows-Version ist die Version JES 2 (File *JES_with_java_bundle.zip* laden)
Hinweis: Sie finden das Installationsfile *JES_with_java_bundle.zip* auch unter *Y:\lehre\LV_Gr_d_Inf_WS0607\ Installfiles_JES_2_und_Python_2_5*

- Unterlagen zur Lehrveranstaltung „Introduction to Media Computation“ am *College of Computing* des *Georgia Institute of Technology* auf der Basis von JES:
http://coweb.cc.gatech.edu/cs1315/4281

→ *In der Übung nutzen wir JES zur Manipulation von Bildern, Sound usw. mit Python-Skripts, da JES dazu spezifische Funktionen, die nicht zum Standard-Python gehören, bereitstellt. Da JES allerdings keine Eingaben von Tastatur unterstützt, verwenden wir in Programmen mit Tastatur-Eingabe die oben genannte IDLE (Python GUI).*
(Das JES-Console-Modul unterstützt zwar Eingaben von Tastatur, aber nicht standardgemäß.)

0.3 Weitere Python-Umgebungen

Es gibt einige weitere Freeware-Entwicklungsumgebungen für Python, z.B. ActivePython unter http://www.activestate.com/Products/ActivePython/, die wir in der Übung *nicht* nutzen werden.

0.4 Dokumentationen, Tutorials usw. zu Python, JES …

Auf den Festplatten finden Sie unter *Y:\lehre\LV_Gr_d_Inf_WS0607\Material zu Python* einige Tutorials, Dokumentationen usw. Dazu gehören:
Python_V2.4_QuickReference_modern_a4.pdf, JES - Alle Help-Texte.doc, Getting started in JES.doc, Python - Using IDLE.doc, Python_V2_1_Tutorium_deutsch.pdf, Python_Tutorium_V2_5_Rossum_englisch.pdf.

Im Internet finden Sie zahlreiche weitere Materialien, z.B. unter:
- Introductory Material on Python: http://www.python.org/doc/Intros.html
- Python 2.5 Documentation: http://docs.python.org/download.html *usw.*

→ In der 6. Übung benutzen wir zunächst nur die IDLE (Python GUI).

---

1) Python im interaktiven Modus *(mit IDLE)*

- Öffnen Sie die IDLE (Python-GUI) - GUI bedeutet dabei *Graphical User Interface* - gemäß: *Start/Programs/Programming/Python 2.5/IDLE (Python GUI)*
Achtung: Das Laden der IDLE dauert einige Sekunden.

- Nach dem Laden öffnet sich die Python-Shell mit ihrem Prompt (Eingabeaufforderung): **>>>**
Hier kann man im interaktiven Modus des Interpreters arbeiten: Eine mit <ENTER> abgeschlossene Python-Anweisung wird vom Interpreter sofort abgearbeitet. *Wir experimentieren jetzt damit.*

(Achtung: Groß- und Kleinschreibung ist in Python signifikant!!!)

****

→ ***Vorbemerkungen und Übungen im interaktiven Modus sind STETS als Vorbereitung VOR der Übung durchzuarbeiten.***

****

a) Ausdruckanweisungen: Arithmetische, logische u. Zeichenkettenausdrücke

Geben Sie folgenden Ausdruckanweisungen ein und schließen diese mit ENTER ab. **Interpretieren Sie die Antwort des Interpreters:**

Arithmetische Ausdrücke:

```
>>>34                    <ENTER>        #34 vom Typ int
>>>34L                                  #34 vom Typ long
>>>5+83                                 #Wert vom Typ int
>>>5.0+83                               #Wert vom Typ float
>>>type(5+83.0)                         # Achtung: Groß- u. Kleinschreibung ist signifikant!!!
>>>(2+5)*(7-2)
>>>22/5
>>>22.0/5
>>>22/5.0
>>>22/0
>>>6*(7-4)/3-3*(6+1)                    #Berechnen Sie vorher das Resultat auf dem Papier!
                                        #Abarbeitung von links nach rechts; Klammern zuerst;
                                        #Punkt- vor Strichrechnung (* / vor +-)
>>>6*7-4/3-3*6+1                       #Berechnen Sie vorher das Resultat auf dem Papier!
>>>22%5                                #Divisionsrest
>>>2**6                                #2^6
>>>3.4**2.9                            #3.4^2.9
>>>011                                 #Ausgabe der Oktalzahl 011 als Dezimalzahl 9
>>>0x11                                #Ausgabe der Hexadezimalzahl 0x11 als Dezimalzahl 17
```

Logische Ausdrücke:

```
>>>4>=2
>>>7<5
>>>"Tag"=="Nacht"              #Zeichenketten in "..." oder '...'
>>>'Tag'=='Nacht'
>>>(5>3) and (8>12)
>>>5>3 and 8>12                #Klammern nicht notwendig, da Vergleichsoperatoren
                               #vor logischen Operatoren abgearbeitet werden
>>>not(1<0) or (1<0)
```

Zeichenketten-Ausdrücke:

```
>>>'Freiberg'                  #String ist Sequenz von Zeichen
>>>"Frei"+"berg"               #Konkatenation von Strings
>>>"143"+"17"                  #Konkatenation von Strings

>>>"""Lange Zeichenketten ueber
mehr als eine Zeile werden in drei hintereinander
gestellte Anfuehrungszeichen oder Hochkommata
eingeschlossen"""
```

b) Aufruf ausgewählter Python-Standardfunktionen:

```
>>>abs(-34.6)          #Liefert absoluten Wert einer Zahl (ganz, reell oder komplex)
>>>chr(67)             #Gibt Zeichen mit ASCII-Nr. i zurück (chr … character)

>>>ord("B")                 #Gibt die ASCII-Nr. des ASCII-Zeichens "B" zurück
>>>float(4)                 #konvertiert String oder Zahl in Gleitkommazahl
>>>float("23.5")
>>>int(19.8)                #konvertiert String oder Zahl in ganze Zahl (ganzer Anteil)
>>>len("Freiberg")          #Gibt Länge einer Sequenz (hier: String) zurück
```

c) Aufruf ausgewählter mathematischer Funktionen aus dem Modul *math*

```
>>>import math
>>>math.e                              #Eulersche Zahl e
>>>math.exp(3)                         #e^3

#Oder anders:
>>>from math import e
>>>e
>>>exp(3)                              #Fehler!

#Oder anders:
>>>from math import *                  #In Programmen i. Allg. nicht
                    #zu empfehlen wegen moeglicher Namenskonflikte
>>>exp(3)                              #e^3
>>>pow(5.1,4)                          #5.1^4
>>>log(e)                              #ln(e)
>>>log10(1000)                         #^10log(1000)
>>>pi                                  #Die Zahl π
>>>sqrt(100.0)                         #Quadratwurzel aus 100.0
```

```
>>>sin(pi/2)
>>>fabs(-23.5)                    #Absolutwert einer Zahl (als Gleitkommazahl)
```

d) Zuweisungen, Standardausgabe und formatierte Ausgabe mit print

```
>>>x=3.66666     #Zuweisung in Python intern anders als in Sprachen wie C und Pascal:
      #Wert 3.66666 wird in einen Speicherbereich der Objekte (Daten) gespeichert,
      #der Name x zusammen mit einem Verweis („Adresse“) auf den Wert in einen anderen
      #Speicherbereich.
>>>x
>>>id(x)       #Ausgabe des zum Namen x gehörenden Verweises auf den zugehörigen Wert
>>>y=24
>>>y
>>>id(y)
```

Zu print( ): siehe auch *Formatierte Ausgabe mit print in Python.doc* im Übungsverzeichnis

```
>>>print x                #Standardausgabe des Wertes der Variablen x auf Bildschirm
                          #(mit 5 Stellen nach dem Dezimalpunkt; gerundet auf 5.Stelle)
>>>print "%8.4f" %x       #Formatierte Ausgabe im Datenfeld der Breite 8 mit 4 Stellen
                          #nach dem Dezimalpunkt (gerundet auf 4. Stelle nach dem Punkt)
                          #--> Rundung gemäß der internen Zahlendarstellung (siehe oben)
>>>print "%.5f" %x       #Formatierte Ausgabe mit 5 Stellen nach dem Dezimalpunkt
                          #(gerundet; vor dem Punkt so viele Stellen wie notwendig)
>>>print x,y             #Standardausgabe zweier Werte (getrennt durch ein Leerzeichen!)
>>>print "%.2f %.4f" %(x,y)    #Formatierte Ausgabe zweier Werte;
                                #Leerzeichen zwischen Formatangaben beachten!
>>>print "%.2f\n%.4f" %(x,y)   #\n … Steuerzeichen Newline, d.h. Zeilenwechsel
>>>x                           #bisheriger Wert:        3.66666
>>>x=x+1                       # zugewiesen wird:       4.66666
>>>x                           #aktueller Wert:         4.66666
>>>zkette="Freiberg"+"er"      #Konkatenation von Strings
>>>print zkette                #Standardausgabe des Wertes der Variablen zkette
>>>print zkette[0],zkette[9]   #Ausgabe des 1. und letzten Zeichens von zkette;
                               #dazwischen fügt print ein Leerzeichen ein!
>>>zkette=zkette+" Mulde"     #Beachte Leerzeichen vor Mulde
>>>print zkette

>>> komp1 = (2+4j)            #komplexe Zahl zuweisen
>>> komp2 = (7+6j)
>>> print komp1+komp2         #Standardausgabe der Summe zweier komplexer Zahlen
```

e) Eingabe von Tastatur mit input()

```
>>>input("Reelle Zahl eingeben: ")      #Eingabe einer reellen Zahl von Tastatur
                                        #Als Prompt wird ausgegeben:  Reelle Zahl eingeben:
>>>input()                             #Eingabe einer Zahl von Tastatur ohne Prompt (!)
>>>input("String eingeben: ")                  #String müssen mit input()
```

#in Hochkommata (!!!) eingeben werden, sonst Fehler *(ohne Hochkommata probieren!)*!

```
>>>a=input("Ganze Zahl eingeben: ")     #Zuweisung der Eingabe an a
>>>print a                    #Standardausgabe des Wertes von a
```

```
>>>zk=raw_input("Zeichenkette eingeben: ")
          #Eingabe eines raw-String (ohne Hochkommata!) von Tastatur und Zuweisung an zk
>>>print zk                    #Standardausgabe des Wertes von zk
```

Hausaufgabe 1: Die gesamte Aufgabe 1) kann man auch lösen mit
- dem **Standard-Python-Kommandozeileninterpreter**:
Start/Programs/Programming/Python 2.5/Python (command line)
- sowie natürlich auch mit der **JES-Entwicklungsumgebung**
(aber bei JES input() von Tastatur nicht möglich!):
Start/All Programs/°Programmierung/~Jython Environment for Students/JES

→ ***Führen Sie dies als Hausaufgabe durch.***
Achtung: Das Laden der Oberfläche JES dauert in den Pools einige Sekunden.

---

2) Das berühmte „*Hello world“ (vgl. Vorlesung)*
im interaktiven Modus und als Programm (jeweils auch mit Funktion) ***(mit IDLE)***

a) Im interaktiven Modus (vgl. Vorlesung):
Versuchen Sie es zunächst selbst!

```
>>>print "Hello World"
```

b) Im interaktiven Modus als Funktion plus Funktionsaufruf (vgl. Vorlesung):
Versuchen Sie es zunächst selbst!

```
>>> def hello():
        print "Hello world"

>>> hello()              #Aufruf der Funktion hello()
Hello world              #Ausgabe des Python-Interpteters
>>>
```

c) Als Python-Programm (zunächst als (Haupt-)Programm ohne Funktion):
(1) In Python-Shell (IDLE) ausführen: `File/New Window`
→ Ein Edit-Fenster öffnet sich.

(2) Geben Sie im Edit-Fenster das Ihnen aus der Vorlesung bekannte **Programm** ein. Beachten Sie dabei die farbige Darstellung:

```
# Erstes Programm: Hello world
print "Hello world"
```

(Das Programm (ohne Funktion) besteht aus dem Kommando aus Abschnitt a).)

(3) - Im Edit-Fenster ausführen: `File/Save As...`
- In *Save As*-Fenster eingeben:
  `Save in:` *<Hier Ihr Arbeitsverzeichnis auswählen>*
  `File name:` `hello.py` *(Name: frei, Typ: py)*
  `Save as type:` `*.py`

(4) Im Edit-Fenster ausführen: `Run/Check Module`
Als Folge gibt es zwei Möglichkeiten:
a) Falls im Quelltext *hello.py* ein **Syntax**fehler vorhanden ist, öffnet sich in der IDLE ein *Syntax error*-Fenster für den *ersten* erkannten Syntaxfehler mit einer entsprechenden Fehlerausschrift.

- Klicken Sie dann mit der Maus of *OK*. Der Kursor steht nun im *Edit*-Fenster. Die Quelltextstelle, an der der Syntaxfehler ***erkannt*** wurde, ist rot markiert. In der Regel liegt der Fehler ***vor*** der roten Markierung. Verbessern Sie den Fehler.
→ Setzen Sie wieder mit (4) fort (Schleife: solange bis kein Syntaxfehler mehr gefunden wird)

(Hinweis: Im schwarzen Kommandozeileninterpreter *python.exe* wird die Stelle, an der der Fehler *erkannt* wurde, durch einen kleinen „Pfeil“ markiert.)

oder b) Es wurde kein Syntaxfehler erkannt und der Kursor steht (ohne eine neue Ausschrift) im Python Shell-Fenster.

Die vom System gefundenen Syntaxfehler wurden also korrigiert.

(5) Abarbeitung des Programms *hello.py*
Im Edit-Fenster: `Run/Run Module`
→ Die Ausgabe des Programms erscheint im Python Shell-Fenster.

d) Als Python-Programm mit Funktion:
Alle Schritte **entsprechend** zu Abschnitt **c)**, d.h. führen Sie auch hier die **Schritte (1) bis (5)** aus !!!

Geben Sie dabei als Python-Programm ein:

```
#"Hello world" als Programm mit Funktion und Funktionsaufruf
def hello():                    #Funktion ohne formale Parameter
    print "Hello world"
#Hauptprogramm:
hello()                         #Aufruf der Funktion hello()
```

Speichern Sie das Programm z.B. als *hellofkt.py* **usw. (d.h. Schritt (1) bis (5) wie unter c))**

(Das Python-Programm mit Funktion besteht also aus den beiden Einzelkommandos aus b), d.h. vom interaktiven Modus mit Funktion. Im Programm wurden noch Kommentare hinzugefügt.)

e) ***(Hausaufgabe:)*** **Abarbeitung des Hello-Programms vom MS-DOS-Command Prompt *oder* von Start/Run sowie vom Windows-Explorer**
Ein Interpreter speichert, anders als ein Compiler, bekanntlich **kein** ausführbares exe-File. Das heißt: zur Ausführung wird stets der Interpreter benötigt!

(1) Man kann aber den Python-Interpreter auch **vom MS-DOS-Command Prompt oder von Start/Run** aufrufen und ihm beim Aufruf den Bezeichner des Programms als Parameter übergeben.
→ Funktioniert ohne Pfade nur, wenn Pfade richtig gesetzt sind. Sonst Pfade angeben. In den Pools sind die Pfade kompliziert - deshalb hier nur die allgemeine Aufrufform:

<pfad>\python <pfad>hello.py

Dann wird der Python-Interpreter aufgerufen und das Programm interpretierend ausgeführt.

(2) Wenn der Typ *.py* umter MS-Windows korrekt mit dem Python-Interpreter python.exe verbunden ist, dann kann man das Gleiche wie unter (1) auch im Windows-Explorer durch Doppelklicken auf *hello.py* erreichen:
der Python-Interpreter wird gestartet und *hello.py* als Parameter übergeben.

Probieren Sie es aus!

→ Oooh, das klappt nicht richtig…:

Das schwarze Fenster des Python-Interpreters wird geöffnet, Hello world *ausgeschrieben und das Fenster sofort wieder geschlossen.*
Das geht so schnell, dass man nichts erkennen kann!

Die Ausgabe hält allerdings im schwarzen Fenster an (und man kann *Hello world* in Ruhe lesen), wenn man als letzte Programmanweisung eine *raw_input()*-Anweisung hinzufügt.

Also:
(1) Führen Sie von der Python-Shell aus: `File/Open...`
Öffnen Sie *hello.py* aus Ihrem Arbeitsverzeichnis (in einem Edit-Fenster)

(2) Ergänzen Sie das Programm etwa wie folgt:

```
#Erstes Programm: Hello world
print "Hello world"
raw_input("Programmende: Taste ENTER druecken")
```

(3) Speichern Sie das geänderte Programm vom Edit-Fenster aus mit *File/Save As…* als *hello2.py* in Ihr Arbeitsverzeichnis.

→ Arbeiten Sie jetzt *hello2.py* vom Windows-Explorer aus ab durch Doppelklicken auf *hello2.py* .
Und diesmal geht alles klar!
(Vergessen Sie nicht als Programmende die Taste ENTER zu drücken!
Erst dann schließt sich das schwarze Fenster.)
---

*(Nochmals der Hinweis: Groß- und Kleinschreibung ist in Python **signifikant**.)*

3) Volumen und Oberfläche einer Kugel (vgl. Struktogramme, Aufg. 2) *(mit IDLE)*

- Schreiben Sie ein Python-Programm (ohne Funktion) für die Berechnung des **Volumens und der Oberfläche einer Kugel** in folgender Form („EVA“ ... Eingabe, Verarbeitung, Ausgabe):

-- Reellen Radius r eingeben (r>=0) (ohne Behandlung von Eingabefehlern)
-- Berechnung der Oberfläche $S = 4*Pi*r^2$ und des Volumens $V = 4/3*Pi*r^3$
(dabei *pi* aus dem Modul *math* verwenden)
-- Ausgabe der berechneten Ergebnisse

- Testen Sie die syntaktische Korrektheit („Check Module“)
- Arbeiten Sie das Programm ab („Run Module“)
- Testen Sie unbedingt die logische Korrektheit durch gut gewählte **Testbeispiele;**
  dabei müssen die Resultate im Voraus bekannt sein (z.B. Taschenrechner) !!!
  (z.B. dreimal abarbeiten mit folgender Eingabe: r=1.0, dann r=0, dann r=24)

---

*(Achtung: Blöcke sind in Python stets **einzurücken**, z.B. auch der Block einer if-Anweisung.)*

4) Bestimmung des Wertes eines Maximums (vgl. Struktogramme, Aufg. 5) *(mit IDLE)*

Achtung: In dieser Aufgabe ist NICHT die Python-Standardfunktion *max( )* zu benutzen!

a) Schreiben Sie ein Python-Programm für die Bestimmung des maximalen Wertes **zweier** beliebiger reeller Zahlen. *(einfache Selektion)*

b) Schreiben Sie ein Python-Programm für die Bestimmung des maximalen Wertes **dreier** beliebiger reeller Zahlen. *(geschachtelte Selektion)*

5*) *(Hausaufgabe)* Ein simulierter Minicomputer *(mit IDLE)*

(vgl. Struktogramme, Aufg.12)

Schreiben Sie ein Python-Programm für einen simulierten "**Minicomputer**", der einfache arithmetische Ausdrücke der Form

<operand_1> <operator> <operand_2>

in folgender Weise berechnet *(geschachtelte Selektion)*:

- Einzugeben sind: 1. und 2. Operand, Operator
- Zugelassene Operatoren:
  - -- Addition: +
  - -- Subtraktion: -
  - -- Multiplikation: * *oder* . (d.h. Stern oder Satzpunkt)
  - -- Division / oder : (d.h. Slash oder Doppelpunkt)
- Fehlerausschriften bei Division durch 0
  sowie wenn ein nicht zugelassener Operator eingegeben wurde
- Die Ergebnisausgabe soll nur einmal im Programm erfolgen.

(**Hinweis:** Verwenden Sie statt der ganzzahligen Variablen *ok* im Struktogramm in Python besser eine *logische* Variable *ok* (Python-Datentyp *bool*).)

- Testen Sie unbedingt die semantische Korrektheit durch gut gewählte **Testbeispiele:**
Sie sollten das Programm mit jedem Operator testen sowie auch einmal einen unzulässigen Operator eingeben.

Hausaufgabe 2:

- Aufgaben, die in der 6. Übung *nicht* geschafft werden, sind als **Hausaufgabe** zu lösen.
Sie werden dann am Beginn der 7. Übung kurz referiert.
- Die 5*) ist als Hausaufgabe zu lösen.

Hinweis: Die Lösungen zu den Übungen erscheinen auf den Platten und im WWW, nachdem die letzte Übungsgruppe diese Aufgaben behandelt hat.