Übungsblatt 7 zur Vorlesung Grundlagen der Programmierung

Übung 1
Ein einfacher Taschenrechner

Schreiben Sie ein Programm, welches den Nutzer nach zwei Eingabezahlen fragt, sowie nach einer auszuführenden Rechenoperation (Addition, Subtraktion, Multiplikation, Division).

Legen Sie in Ihrem Programm für jede Rechenoperation eine eigene Funktion an, die das Ergebnis zurückgibt. Hierbei implementieren Sie zunächst bitte die Addition und Multiplikation. Für die Implementierung der Subtraktion und Division nutzen Sie bitte intern die Funktionen für die Addition und Multiplikation.

Bitte vergessen Sie nicht, Ihre Funktion im Code zu dokumentieren!

Beispielausgabe:

```
Bitte geben Sie eine Zahl a ein: 84
Bitte geben Sie eine Zahl b ein: 2
Welche Rechenoperation soll ich ausführen? Wählen Sie 'a' für
Addition, 's' für Subtraktion, 'm' für Multiplikation und 'd'
für Division: d
Rechnung: 84/2 = 42
```

Übung 2
Kreise und Kugeln

Erzeugen Sie in einem Ordner zwei Dateien mit Namen `kreis_kugel.py` sowie `user_interaction.py`

Implementieren Sie in der Datei `kreis_kugel.py` einen Satz an Funktionen, der für einen gegebenen Radius

- die Fläche des Kreises ($A(r) = r^2\,\pi$) mit Namen `flaeche`,
- den Umfang des Kreises ($U(r) = 2\,r\,\pi$) mit Namen `umfang`,
- die Oberfläche der Kugel ($O(r) = 4\,A(r)$) mit Namen `oberflaeche`, sowie
- das Volumen der Kugel ($V(r) = 4/3\,\pi\,r^3$) mit Namen `volumen`

zurückgibt.

Starten Sie nun die Datei `user_interaction.py` folgendermaßen:

```
import kreis_kugel

print (kreis_kugel.flaeche(42))
print (kreis_kugel.umfang(42))
print (kreis_kugel.oberflaeche(42))
print (kreis_kugel.volumen(42))
```

Adaptieren Sie die Datei `user_interaction.py` derart, dass der Nutzer nach einem zu bestimmenden Wert (Fläche, Umfang, Oberfläche, Volumen) sowie dem Radius gefragt wird und die entsprechende Antwort ausgegeben wird.

Übung 3
Palindrome

Palindrome sind Zeichenketten, die von vorne und von hinten gelesen das identische Wort ergeben. Beispiele sind Wörter wie *Lagerregal, Elle, Ebbe, Ehe, Neffen, Retsina-Kanister, Reliefpfeiler* oder auch *Rentner*.

Schreiben Sie bitte eine rekursive Funktion, die prüft, ob es sich bei einer Zeichenkette um ein Palindrom handelt.

Satzpalindrome sind Wortketten, die abgesehen von Leerzeichen und Sonderzeichen sowohl von vorne als auch von hinten den identischen Satz ergeben. Beispiele sind Sätze wie *„Lesen Esel?“*, *„Vitaler Nebel mit Sinn ist im Leben relativ.“* oder *„Bei der Edna redete der andere Dieb.“*

Schreiben Sie bitte eine weitere Funktion, die prüft, ob es sich bei einem Text um ein Palindrom handelt. Bitte nutzen Sie für Ihre Implementierung die zuletzt entwickelte Funktion zum Prüfen von Palindromen.

Übung 4
Addition Rekursiv

In der Vorlesung haben wir die Multiplikation von zwei Zahlen rekursiv implementiert. Bitte vergegenwärtigen Sie sich nochmals die Zerlegung der Multiplikation in eine rekursive Addition.

In ähnlicher Form lässt sich auch die Addition rekursiv darstellen. Betrachten Sie zur Herleitung des rekursiven Algorithmus die Addition der Zahlen 3 und 4 und überlegen, wie Sie die Rekursion aufbauen könnten:

```
var_a + var_b      = (3) + 4
                   = (3 + 1) + 3
                   = (3 + 1 + 1) + 2
                   = (3 + 1 + 1 + 1) + 1
                   = (3 + 1 + 1 + 1 + 1) + 0
```

Übung 5
Potenzierung Rekursiv

Die Potenz x^n kann rekursiv dargestellt werden. Hierbei gilt: $x^0 = 1$. Bitte implementieren Sie die Funktion in Python. Betrachten Sie hierfür folgendes Beispiel:

```
var_a^var_b         = 3^4
                    = 3 * 3^3
                    = 3 * 3 * 3^2
                    = 3 * 3 * 3 * 3^1
                    = 3 * 3 * 3 * 3 * 3^0
```

Übung 6

Die Fibonacci-Folge

Leonardo Fibonacci beschrieb im Jahr 1202 das Wachstum einer Kaninchenpopulation als Summenfolge, bei der jede Zahl der Folge als Summe der beiden vorangehenden Zahlen definiert ist:

$$f_n = f_{n-1} + f_{n-2}\,, n > 3$$

$$f_1 = f_2 = 1$$

Im Folgenden der Anfang der unendlichen Reihe:

$$0, 1, 1, 2, 3, 5, 8, 13, \ldots$$

Die Fibonacci-Zahlen beschreiben aber tatsächlich viele verschiedene Wachstumsvorgänge in der Natur. So lässt sich beispielsweise die Struktur eines Schneckenhauses mithilfe der Fibonacci-Reihe beschreiben:

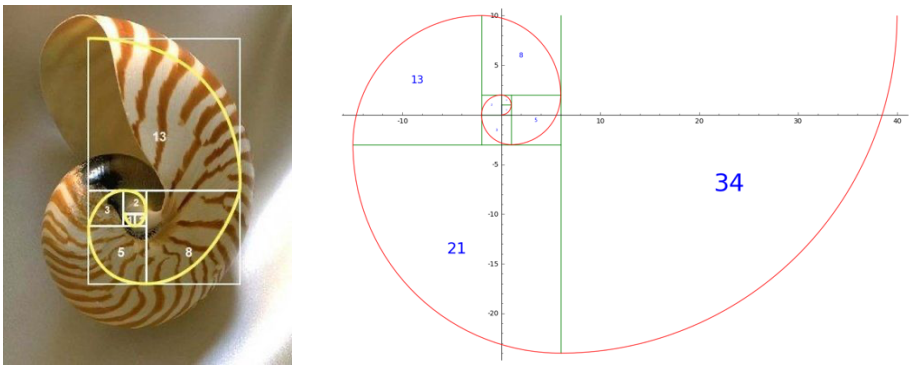


Abb. 1: Die Fibonacci-Folge und die Struktur eines Schneckenhauses

Bitte implementieren Sie eine Funktion zur Bestimmung der n-ten Zahl in der Fibonacci-Folge unter Verwendung eines rekursiven Ansatzes.

Übung 7

Das Pascal'sche Dreieck und eine Fingerübung zur Textausgabe

Das Pascal'sche Dreieck ist folgendermaßen definiert:

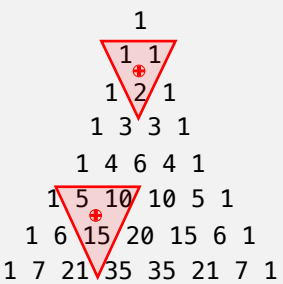
$$p(z,s) = \begin{cases} 1, & s = 1 \vee s = s \\ p(z-1, s-1) + p(z-1, s), & sonst \end{cases}$$

z bezeichnet hier die Zeile und s die Spalte. Hierbei ergibt sich der Wert einer Zahl im Dreieck *anschaulich* immer über die Summe der beiden *grafisch* über der Zahl liegenden Werte in der vorhergehenden Zeile

Hier der Anfang des Dreiecks …

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
```

… und hier nochmal in „schön“ 😉



Bitte implementieren Sie die Funktion zur Bestimmung des Werts $p(z,s)$ im Pascal'schen Dreieck rekursiv und erzeugen eine „schön“ formatierte Ausgabe als Dreieck unter Berücksichtigung der Breite der Zahlen. Hierbei darf der Nutzer angeben, wie viele Zeilen des Pascal'schen Dreieck er sehen möchte.

Übung 8

Größter gemeinsamer Teiler (ggT)

Der größte gemeinsame Teiler (ggT) von zwei Zahlen a und b bezeichnet die Zahl, durch die sowohl a als auch b ohne Rest teilbar ist. Lösungsmöglichkeiten zur Bestimmung des ggT beinhalten die Primfaktorzerlegung aber auch den Euklidischen Algorithmus. Die Idee des Euklidischen Algorithmus besteht darin, dass der gemeinsame Teiler von a und b dem Teiler von b und dem Rest der Division von a und b entspricht.

```
Beispiel – Suche den größten gemeinsamen Teiler von 70 und 42:

        70 : 42 = 1 Rest 28
        42 : 28 = 1 Rest 14
        28 : 14 = 2 Rest 0
        14 : 0 >> Ergebnis

Gesuchtes Ergebnis: 14.
```

Implementieren Sie die Lösung zur Bestimmung des ggT bitte zunächst iterativ und dann rekursiv unter der Angabe der Werte in jeder Iteration.

Übung 4 von letzter Woche – zweiter Versuch 😉

Runden binärer Zahlen auf eine feste Anzahl Nachkommastellen

Erweitern Sie den Code zur Umrechnung dezimaler Zahlen zu binären Zahlen aus dem Skript dahingehend, dass Sie die binäre Zahl auf eine feste Anzahl Nachkommastellen runden. Zusätzlich soll die Differenz zwischen der gerundeten Binärzahl und der ursprünglichen Dezimalzahl berechnet und angezeigt werden.

Hinweis zur Lösung: zerlegen Sie das Problem gedanklich zunächst in mehrere Teilprobleme. Erstellen Sie pro Teilproblem eine Funktion und rufen diese auf.