Übungsaufgaben 1
(Zahlen, Strings, Bedingungen, while-Schleifen)

Hausaufgabe: Lösen Sie mindestens 6 der folgenden Aufgaben. Sie dürfen die Aufgaben in Gruppen bearbeiten. Aufgaben mit Stern * zählen doppelt.

Abgabe im ISIS-Kurs, spätestes Abgabedatum wird dort angegeben, Gruppenpartner bitte bei der Abgabe vermerken.

Für einige Aufgaben benötigen Sie Zufallszahlen. Für Fließkomma-Zufallszahlen zwischen 0 und 1 können Sie die Funktion `random` des Moduls `random` benutzen oder die Funktion `rand` des Moduls `numpy.random`. Für ganze Zufallszahlen gibt es in den beiden Modulen eine Funktion `randint` (mit kleinen Unterschieden, sehen Sie sich help(...) an!).

Aufgaben aus dem Labor

Diese Aufgaben kennt ihr vielleicht schon aus dem Mathesis-Kurs ...

1. **Abstand von Fliegen auf einem Papier**

   Zwei Fliegen landen zufällig (gleichverteilt) auf einem Quadrat der Seitenlänge eins. Mit welcher Wahrscheinlichkeit ist der (euklidische) Abstand der Fliegen größer als 1? Ermitteln Sie einen Näherungswert durch eine Simulation über eine Million Landungen.

   Gelingt es Ihnen, den Wert theoretisch zu bestimmen (technisch nicht so leicht)?

2. **Irrfahrt (2d)**

   Simulieren mit Hilfe der Turtlegraphik ein Wesen, dass im Ursprung startet und in jedem Zeitschritt mit Wahrscheinlichkeit 1/4 nach N/W/S/O läuft (eine gewisse Strecke s). Beende die Simulation, wenn das Wesen wieder im Ursprung ist.

   Haben Sie Vermutungen darüber, mit welcher Wahrscheinlichkeit das Wesen überhaupt zurückkehrt oder über die mittlere Dauer der Irrfahrt? Wie könnten Sie solche Hypothesen mit einer Simulation auf Plausibilität überprüfen? (Beweisen könnten Sie sie allerdings nur mit Hilfe von Mathematik.)

3. **Genomanalyse*** Lesen Sie mit dem folgenden Stück Code die ganze Datei `protein_ecoli.txt` in einen String. Es handelt sich um den Teil der DNA von E. Coli, der das erste Protein (bezüglich einer in den großen Gendatenbanken festgelegten Reihenfolge) codiert.

```
f = open("protein_ecoli.txt", "r")
seq = ""

for line in f:
        seq = seq + line.rstrip()

f.close()
```

Schreiben Sie ein kleines Programm, das das längste Code-Stück sucht, das (überschneidungsfrei) doppelt in dieser Sequenz vorkommt (oder alle längsten Stücke).

4. **Schere-Stein-Papier**

Schreiben Sie ein Programm, das gegen einen menschlichen Spieler Schere-Stein-Papier spielt. Was würden Sie intuitiv für ein besonders gutes Programm gegen einen Spieler halten, über den nichts bekannt ist, außer, dass er mit gewisser (unbekannter) Wahrscheinlichkeit eines der drei Symbole zeigt, wenn das Ziel ist, möglichst oft zu gewinnen, egal was der menschliche Spieler tut ? Können Sie beweisen, dass Sie die optimale Strategie gefunden haben ?

Diese einfache Aufgabe lässt sich als Ausgangspunkt für die Spieltheorie verwenden, indem man sie etwas weiter treibt. Was ist die optimale Strategie, wenn Siege nicht gleichviel zählen, sondern eine Aus- oder Einzahlung stattfindet gemäß der folgenden Matrix ?

A	B	Ausz. an A	Ausz. an B
Schere	Papier	2	-2
Papier	Stein	3	-3
Stein	Schere	1	-1
Papier	Schere	-2	2
Stein	Papier	-3	3
Schere	Stein	-1	1

Zeigen beide das gleiche Symbol, erfolgt keine Auszahlung. Ein hierfür relevantes Stichwort aus der Spieltheorie lautet *beste gemischte Strategie* oder auch *Gleichgewicht gemischter Strategien*.

Frage zum Grübeln: Lässt sich die Strategie gegen einen *menschlichen* Spieler, der keine Hilfsmittel benutzt, verbessern ?

5. **Häufigkeit der Vokale**

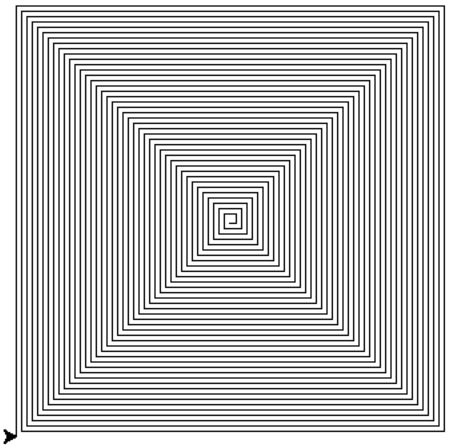
Schreiben Sie ein Programm, das für einen String (also Text) die Anzahl der 'e','a','i','o' und 'u' ermittelt. Sie können statt eines eingegebenen Strings auch einen ganzen Text verwenden. Dazu kopieren Sie den Text als Datei (auf ISIS stehen als Beispiele `das_urteil.txt` und `dickens_martinchuzzlewit.txt` .) in das Verzeichnis, wo sich ihr Programm befindet. Mit

```
with open("das_urteil.txt","r") as f:
    text=f.read()
```

wird der ganze Text in dem String 'text' gespeichert. Sie können daran etwa die Sprache erkennen, probieren Sie's aus.

6. **Ein Spiralornament**

   Zeichnen Sie mit der Turtle ein spiralartiges Ornament mit vierzig „Umrundungen".



7. **Diskretes Räuber-Beute-Modell**

   Dieses in der mathematischen Biologie betrachtete Modell beschreibt die Entwicklung der Populationsdichte x einer Beutespezies ('Kaninchen') und der Populationsdichte y einer Räuberspezies ('Füchse'). Die Populationen werden nur in gewissen Zeitabständen ermittelt, und das Modell macht auch nur für diese Zeiten Voraussagen. Man sagt, das Modell sei *diskret in der Zeit*.

$$x_{n+1} = ax_n(1-x_n) - bx_ny_n \qquad y_{n+1} = cx_ny_n \ .$$

   Schreiben Sie ein Programm, dass die Populationen mit den Parametern $a = 3.5$, $b = 2$, $c = 4.05$ und den Anfangswerten $x_0 = 0.5$ und $y_0 = 0.4$ hundert Schritte lang berechnet und dabei (x,y) mit turtle-Graphik visualisiert.

   Sehen Sie sich die Bilder auch für andere Werte der Parameter an.

Andere Aufgaben

Einige Aufgaben sind einfach, einige schwieriger. Um was zu lernen solltet ihr euch nicht nur solche Aufgaben suchen, die euch sehr einfach vorkommen.

8. **Buchstaben zu Bild**

   Schreiben Sie ein Programm, das um die Eingabe einer Zeichenkette aus den Buchstaben 'L','R'

und 'G' bittet. Anschließend zeichnet die Turtle-Graphik die Kurve, die durch die Anweisungen 'L' (nach links drehen), 'R' (nach rechts drehen) und 'G' (50 Schritte geradeaus gehen) gegeben ist, also liefert beispielsweise GLGLGLG ein Quadrat.

9. **Eine quadratische Gleichung**
   Schreiben Sie ein Programm, das die Koeffizienten a,b,c einer quadratischen Gleichung $ax^2 + bx + c = 0$ einliest und die reelle(n) Lösung(en) ausgibt, bzw. gegebenenfalls eine Meldung, dass es keine reelle Lösung gibt. – Schreiben Sie eine Variante, die die komplexen Lösungen ausgibt. In beiden Fällen brauchen Sie die Wurzelfunktion `sqrt(x)`, die Sie wieder aus `numpy` oder `math` importieren können.

10. **Fakultät berechnen**
    Schreiben Sie ein Programm, das eine natürliche Zahl einliest und deren Fakultät ausgibt.

11. **Quersumme**
    Schreiben Sie ein Programm, das als Eingabe eine Dezimalzahl erwartet und die Quersumme ausgibt.

12. **Collatz-Folge**
    Die Collatz-Folge $(a_n)_{n\in\mathbb{N}}$ zum Anfangswert $a_0 \in \mathbb{N}$ ist wie folgt rekursiv definiert: Ist a_n gerade, so ist $a_{n+1} = \frac{a_n}{2}$, andernfalls $a_{n+1} = 3 * a_n + 1$. Es ist nicht bekannt, ob diese Folge für alle Anfangswerte irgendwann bei 1 ankommt (und anschließend periodische weitergeht $1 \to 4 \to 2 \to 1 \to \cdots$.) Schreiben Sie ein Programm, dass für einen eingebenen Anfangswerd die ersten 100 Folgenglieder ausgibt. Schreiben Sie anschließend eine Variante, die alle Glieder bis zur ersten 1 und den Index dieser ersten 1 ausgibt. Schreiben Sie anschließend ein Programm, dass im Zahlenbereich bis zu 1 Million (oder gerne auch größer) den Anfangswert mit dem längsten Anfangsstück bis zur ersten 1 ausgibt.

13. **Verschlüsselung mit Caesarchiffre**

    Bei der Caesarchiffre, einem einfachen Verfahren zur Verschlüsselung von Nachrichten, ersetzt man die Buchstaben mit dem n. Nachfolger im Alphabet (zyklisch), also z.B. für $n = 3$ wird a zu d, b zu e, c zu f, ..., w zu z, x zu a, y zu b und z zu c. Aus *eswarschondunkel* wird dann *hvzduvfkrqgxqnho*. (Traditionell verwendet man bei den einfachen Verschlüsselungen keine Groß-/Kleinschreibung, Interpunktion und Leerzeichen – das „Knacken" des Codes wäre sonst zu einfach. Ebenso können hier Umlaute und ß besser als ae, oe, ue und ss dargestellt werden.)

    Schreiben Sie ein Programm, das ein n und den zu verschlüsselnden Klartext einliest und dafür die verschlüsselte Zeichenkette ausgibt.

14. **Pseudozufallszahlen**
    Wenn man in Programmen Zufallszahlen benötigt, so werden meist Folgen von Pseudozufallszahlen verwendet, d.h. eine Folge von Zahlen $a_1, a_2, a_3, \ldots$, die für statistische Analysen wie

das Ergebnis eines Zufallsprozesses aussehen. Ein einfaches Beispiel solcher Pseudozufallfolgen mit Folgegliedern aus dem Bereich von 0 bis 65535 erhält man durch $a_{i+1} = (25173a_i + 13849)\%65536$. (Dies ist eine übliche Weise für die Definition von Zahlenfolgen: Die Gleichung gibt an, wie man aus einem Folgeglied a_i das nächste Folgeglied a_{i+1} errechnet.) Schreiben Sie ein Programm, das das erste Folgeglied (den *seed*) vom Nutzer einliest und die nächsten 20 Folgeglieder ausgibt.
Zusatz: Veranschaulichen sie die Zufallszahlen, indem Sie durch diese Zahlen die Bewegung einer `turtle` steuern. Dafür gibt es verschiedene Möglichkeiten, lassen Sie sich was einfallen. Wenn das Ergebnis schön wirr aussieht, scheint der Zufall zufällig genug zu sein.

15. **Primzahltest**
    Schreiben Sie ein Programm, das eine natürliche Zahl einliest und prüft, ob es sich um eine Primzahl handelt.

16. **Klammerung prüfen**
    Schreiben Sie ein Programm, das für eine Zeichenkette überprüft, ob die enthaltenen (runden) Klammern den Regeln einer korrekten Klammerung folgen. Dazu muss für jede öffnende Klammer „(“ eine nachfolgende schließende Klammer „)“ existieren und es darf keine schließende Klammer ohne vorhergehende öffnende Klammer dazu vorkommen. Andere Zeichen als die runden Klammern sollen einfach ignoriert werden.

    Beispielsweise korrekt geklammert sind `'()'`,`''`, `'(()(a)( ()((b))))'`, nicht korrekt `'(()'` und `'a (()()) b)'`.

17. **Harshad-Zahlen**
    Eine natürliche Zahl heißt Harshad-Zahl, wenn sie durch ihre Quersumme (bezüglich der Dezimalschreibweise) teilbar ist. Beispielsweise ist für 777 die Quersumme $7+7+7=21$ und teilt 777. Schreiben Sie ein Programm, dass die Harshard-Zahlen von 1 bis 100 berechnet.

18. **Palindrom** Schreiben Sie zwei Programme, die einen String einlesen und prüfen, ob es sich um ein Palindrom handelt: 1. ein einzeiliges Programm, das ohne Schleifen auskommt und die besonderen String-Operationen verwendet. 2. ein Programm, das mit Schleifen arbeitet.

Mehr Turtle

Hier noch einige Aufgaben, die Turtlegraphik verwenden.

19. **regelmäßiges Polygon**
    Schreiben Sie ein Programm, das um Eingabe einer (natürlichen) Zahl n bittet und anschließend mit der Turtle-Graphik ein n-Eck zeichnet.

20. **Stern**
    Schreiben Sie ein Programm, das um Eingabe einer Zahl n bittet und anschließend einen n-zackigen Stern zeichnet.

21. **Mäander**
    Schreiben Sie ein Programm, das nach Eingabe einer Zahl n ein n-fach wiederholtes Mäander-Ornament zeichnet (s. (s. `https://de.wikipedia.org/wiki/Mäander_(Ornamentik)`).