Python Aufgaben
für den Informatik Vorkurs

Die Aufgabenblöcke orientieren sich an den jeweiligen Vorlesungstagen. Sie sind absichtlich umfangreich gehalten, damit ihr — falls gewünscht — genügend Aufgaben zur Verfügung habt. Sie müssen nicht alle zwingend bearbeitet werden.

Für Fortgeschrittene: Wenn ihr die Aufgaben bereits gelöst habt, oder ihr bereits wisst, wie es geht — bei allen Aufgaben, in denen vorher festgelegte Zahlen gewünscht sind, dürft ihr diese gerne durch Benutzereingaben ersetzen! Außerdem könnt ihr probieren, bei möglichen vorhersehbaren Fehlern (z.B. Teilen durch 0 bei Divisor aus Eingabe), diese Fehler zu verhindern und eine Fehlermeldung auszugeben.

1 Einstieg (Tag 0)

1.1 Gib den Text “Hallo Welt” aus.

1.2 Hänge zwei Strings aneinander.

1.3 Gib einen Text viermal aus.
Hinweis: Mit etwas herumprobieren geht das sehr schnell, ohne den Text 4-mal zu kopieren!

1.4 Multipliziere zwei Zahlen.

1.5 Dividiere zwei Zahlen. Was passiert, wenn du durch Null teilst?

1.5 Zeihe die Quadratwurzel aus einer Zahl.
Hinweis: Potzengesetze (siehe Matheteil)

2 If - Abfragen (Tag 1)

2.1 Sortiere zwei vorher festgelegte Zahlen nach Größe und gib sie in der richtigen Reihenfolge aus.

2.2 Bestimme die Größte aus drei vorher festgelegten Zahlen und gib diese Zahl aus.

2.3 Sortiere drei vorher festgelegte Zahlen und gib sie in der richtigen Reihenfolge aus. Zeichne dazu zuerst ein Ablaufdiagramm/Flussdiagramm, wie du es aus der Vorlesung kennst. *(Mit Paper+Stift)*

2.4 Überprüfe, ob ein vorher festgelegtes Jahr ein Schaltjahr ist. Dabei sind folgende Regeln zu beachten:

nicht durch 4 teilbar	kein Schaltjahr
durch 4 teilbar	Schaltjahr
durch 100 teilbar	kein Schaltjahr
durch 400 teilbar	Schaltjahr

Beispiele: 1900 kein Schaltjahr, 2000 Schaltjahr, 2004 Schaltjahr, 2006 kein Schaltjahr, ...

2.5 Lass den Benutzer Temperatur (warm, kalt) und Wetter eingeben (regnerisch, verschneit, sonnig) und gib ihm einen Vorschlag zurück, wie er sich dem Wetter entsprechend kleiden soll.
Beispiel:
Eingabe: *warm* und *sonnig*
Ausgabe: *Ein T-Shirt reicht für heute völlig!*

3 for / while - Schleifen (Tag 1)

Die folgenden Aufgaben dürft ihr entweder mithilfe von for- oder while-Schleifen realisieren. Schleifen braucht ihr hier wahrscheinlich auf jeden Fall. Wem das leicht fällt oder wer schon fertig ist, darf sich gerne noch an der jeweils anderen Schleife versuchen.

3.1 Lass dir für eine festgelegte Basis und Exponenten eine feste Potenz berechnen.
Ergebnis für $2^{10} = 1024 \quad 3^4 = 81$

3.2 Bilde folgende Summe für ein von dir vorher festgelegtes n (z. B. $n = 1\,000\,000\,000$)

$$\sum_{k=1}^{n} \frac{1}{k}$$

Ergebnis: für $n = 10^9$: 21.30048150134855.

3.3 Berechne die obige Summe, in dem Du rückwärts von $n = 1\,000\,000\,000$ aufsummierst. Ist das Ergebnis dasselbe? (Später lernt ihr in Rechnerarchitektur, warum...)

3.4 Schreibe ein Programm, das die eulersche Zahl e näherungsweise berechnet (Siehe Matheskript: $e = \sum_{k=0}^{\infty} \frac{1}{k!}$.)

3.5 Schreibe ein Programm, das die Zahl π näherungsweise berechnet (Siehe Matheskript: $\frac{\pi^2}{6} = \sum_{k=1}^{\infty} \frac{1}{k^2}$.)
Wie gut sind die Näherungen in 3.4 und 3.5? (Die ersten 50 korrekten Nachkommastellen von e und π finden sich jeweils auf wikipedia)

4 Listen (Tag 1)

4.1 Füge zwei vorher festgelegte Listen zu einer zusammen und gib diese zurück.
Beispiel: *[3,8,9,2]* zusammengefügt mit *[4,6]* ergibt *[3,8,9,2,4,6]*

4.2 Entferne ein bestimmtes Element aus einer festgelegten Liste.
Hinweis: Kommt ein Element mehr als einmal vor, sollte es überall entfernt werden.
Beispiel: *[3,8,9,5,1,3,6,4]* ohne 3 ergibt *[8,9,5,1,6,4]*

4.3 Trenne eine Liste an einem Index in zwei Teillisten.
Hinweis: Der Index beginnt bei 0. Getrennt wird vor dem Eintrag an der Indexstelle. Soll eine Liste an Index 0 getrennt werden, dann ist die erste Liste leer.
Beispiel: *[1,3,5,8,9,3]* getrennt an Index 3 ergibt *[1,3,5] [8,9,3]*

4.4 Drehe eine festgelegte Liste um.
Hinweis: *list.reverse()* ist zwar nett, aber nicht Sinn dieser Aufgabe!
Beispiel: *[1,3,5,8,9,3]* ergibt *[3,9,8,5,3,1]*

4.5 Lass den Benutzer einen Satz und ein Wort eingeben. Finde heraus ob das Wort im Satz vorkommt und gib das Ergebnis zurück.
Hinweis: *list[4:7]* gibt die Listenelemente von 4 bis 7 zurück. Ein Text kann eine Liste aus Buchstaben sein.
Beispiel: Eingabe: "Hallo du da am PC!' und "PC" Ausgabe: "True"

4.6 Oma Liesel braucht immer sehr lange an der Kasse, um die richtigen Münzen in ihrer Geldbörse zu finden (Ihr habt sicher alle Oma Liesel bereits kennen gelernt) Helft ihr und anderen Supermarktkunden, indem ihr:

a) ein Ablaufplan eines Programmes zeichnet *(Mit Papier+Stift ;)*, welches Oma Liesel die Münzen nennt, die zusammen ihren Betrag (in Cent!) bilden.

b) ein Programm zu diesem Diagramm schreibt.

Hinweis: Verwende keine Kommazahlen.
Beispiel: für 1553 Cent: *['2 Euro', '2 Euro', '2 Euro', '2 Euro', '2 Euro', '2 Euro', '2 Euro', '1 Euro', '50 Cent', '2 Cent', '1 Cent']*
Für Fortgeschrittene:

- Oma Liesel bezahlt auch mit Geldscheinen.
- Gib die Münzen nicht einzeln zurück sondern ihre jeweilige Anzahl.

5 Vertauschen (Tag 2)

Aufgabe: Schreibe eine Funktion, die die Werte zweier Variablen tauscht.
Zum Vorgehen:

- Wie viele Parameter nimmt die Funktion entgegen?
- Wie realisiert man den Tausch?
- Wie vergewissert man sich, dass der Tausch geklappt hat?

6 Eisenwarenhändler (Tag 2)

Ein Eisenwarenhändler bietet verschiedenste Waren zu unterschiedlichen Preisen an. Er führt unter anderem auch Schrauben, Muttern und Unterlegscheiben.

Die Preise für diese Artikel sind:

- Schrauben: 7 ct
- Muttern: 4 ct
- Unterlegscheiben 2 ct

Aufgabe: Schreibe eine Funktion *hornbach*, die für eine gegebene Anzahl Schrauben, Muttern und Unterlegscheiben den Endpreis berechnet.
Zum Vorgehen:

- Wie viele Parameter nimmt die Funktion entgegen?
- Wie genau errechnet sich der Preis?
- Was gibt die Funktion als Rückgabewert?

7 Domino (Tag 2)

Aufgabe: Schreibe eine Funktion, welche Dominosteine bis zu einer gewissen Grenze erzeugt. Die Dominosteine können z.B. so aussehen:

$$(1|1), (1|2), (1|3), (2|2), (2|3), (3|3), \ldots$$

Es dürfen dabei jedoch **keine** Duplikate auftreten! Ist bspw. (1|2) schon erzeugt, darf (2|1) nicht nochmals erzeugt werden, usw.

Zum Vorgehen:

- Welche Parameter kriegt die Funktion?
- Wie erzeugt man die Domino-Steine geschickt?
- Wie verhindert man das Auftreten von Duplikaten?
- Welchen Rückgabewert hat die Funktion?

8 Lotto (Tag 2)

Das gute, alte „6 aus 49“.

Aufgabe: Schreibe eine Funktion *lotto*, welche sechs Zufallszahlen von 1 bis 49 ausgibt.

Zum Vorgehen:

- Wie generiert man Zufallszahlen?
- Wie garantiert man, dass diese im gewünschten Zahlenbereich sind?
- Wie stellt man sicher, dass jede Zahl nur ein einziges Mal gezogen wird?
- Wie garantiert man, dass nur sechs Zahlen gezogen werden?
- Welchen Rückgabewert hat die Funktion?

9 Median (Tag 2)

Aufgabe: Schreibe eine Funktion, die den Median berechnet.

Zur Erinnerung: Gegeben seien Zahlen $x_1, \ldots, x_n$, wobei $x_1 \leq x_2 \leq \cdots \leq x_n$.
Dann ist der Median der o.g. Zahlen:

$$\tilde{x} = \begin{cases} x_{\frac{n+1}{2}} & \text{, falls } n \text{ ungerade} \\ \frac{1}{2}\left(x_{\frac{n}{2}} + x_{\frac{n}{2}+1}\right) & \text{, falls } n \text{ gerade} \end{cases}$$

Zum Vorgehen:

- Was für Parameter nimmt die Funktion entgegen?
- Wie kann man die Funktion für beliebig viele Zahlen anwenden?
- Wie wird gewährleistet, dass die Zahlen sortiert sind?
- Wie unterscheidet man, ob n gerade/ungerade?
- Was gibt die Funktion zurück?

10 Taschenrechner (Tag 2)

Aufgabe: Schreibe einen kleinen Taschenrechner.
Der Taschenrechner sollte die Methoden:

- Addieren
- Subtrahieren
- Multiplizieren
- Dividieren

beherrschen.

Eine Beispieleingabe kann z.B. so aussehen:
Gebe eine Zahl ein: 42
Gebe eine andere Zahl ein: 6
Gebe eine Rechenoperation ein: /"

Das Ergebnis lautet: 7

Zum Vorgehen:

- Wie stellt man sicher, dass der Nutzer nur Zahlen eingeben darf?
- Mit welcher Funktion muss die Eingabe eingelesen werden?
- Wann ruft man welche Methoden auf? (Addieren, Subtrahieren,. . .)
- Darf man durch Null teilen? Wenn ja, was passiert? Wenn nein, wie verhindert man das?

11 Caesar-Verschlüsselung (Tag 3)

Programmiere eine Funktion, die einen String und eine Zahl k als Parameter entgegennimmt und die Caesar-Verschlüsselung zurückliefert. Bei der einfachen Verschlüsselungsverfahren wird jeder Buchstabe zyklisch mit dem Buchstaben ersetzt, der im Alphabet k Buchstaben weiter hinter steht.

- Wie behandele ich Groß- und Kleinschreibung[1]?

Testet eure Funktion mit diesem Wort (und weiteren): caesar $\xrightarrow{k=3}$ fdhvdu

12 Tic-Tac-Toe (Tag 3)

In dieser Aufgabe soll das Spiel Tic-Tac-Toe programmiert werden. Spielregeln bekannt? — Wenn nicht : https://de.wikipedia.org/wiki/Tic-Tac-Toe
Wichtige Überlegungen vorab:

- Wie speichere ich das Spielfeld ab?
- Wie ist die Eingabe vom Spieler?
- Was muss für ein Ende des Spiels überprüft werden[2]?

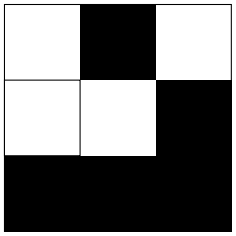
[1] nützliche Funktion str.upper() und str.lower()
[2] neun Bedingungen, für ein Spielende

13 Game of Life (Tag 3)

Programmiere das Spiel *Game of Life von Conway*[3]. In diesem Spiel werden Zellen simuliert, deren Leben von ihren Nachbarfeldern abhängen. Dafür sind die Regeln wie folgt:

- Eine tote Zelle wird lebendig, wenn genau drei Nachbarzellen leben.
- Eine lebendige Zelle stirbt, wenn sie weniger als zwei lebendige Nachbarzellen besitzt (Einsamkeit).
- Eine lebendige Zelle bleibt am Leben, wenn zwei oder drei Nachbarn lebendig sind.
- Eine lebendige Zelle stirbt, wenn mehr als drei Nachbarzellen lebendig sind (Überbevölkerung).

Ein interessantes Muster ist z.B. der Gleiter:



Dafür mache dir am Anfang Gedanken, wie du vorgehst.

- Wie kann ich das Spielfeld speichern?
- Brauche ich eine Kopie?
- Wie gebe ich das Spielfeld aus?
- Wie behandele ich die Ränder?
- Wie lese ich die Nachbarfelder aus?
- Wie realisiere ich die Regeln?

⇒ Welche Funktionen brauche ich?

14 Drachenkurve (Tag 3)

Schreibe ein Programm, das für eine ganze Zahl n die Befehle zum Zeichnen der Drachenkurve (siehe Wikipedia) der Ordnung n ausgibt. Die Befehle sind Strings aus F, L und R Zeichen, wobei F bedeutet "zeichne eine Linie eine Einheit vorwärts" und L bzw. R für "drehe dich um 90° nach links bzw. rechts" steht. Die Drachenkurve der Ordnung n entsteht, wenn ein Streifen Papier n mal in der Hälfte gefaltet wird und dann in rechten Winkeln entfaltet wird. Der Schlüssel zum Lösen des Problems ist die Beobachtung, dass die Kurve der Ordnung n genau eine Kurve der Ordnung $n-1$ gefolgt von einem L gefolgt von der Kurve der Ordnung $n-1$ in umgekehrter Reihenfolge ist. Hier bedeutet umgekehrt, dass jedes L mit einem R getauscht wird und umgekehrt.

[3] https://de.wikipedia.org/wiki/Conways_Spiel_des_Lebens

Die Kodierung der Drachenkurven der Ordnung 0,1,2,3:
F
FLF
FLFLFRF
FLFLFRFLFLFRFRF
Diese Kurve kann gezeichnet werden mit dem Paket `turtle`. Nach `from turtle import *` kennt python die Befehle `forward(5)` (zeichne eine Linie der Länge 5), `left(90)` (drehe um 90 Grad nach links) und `right(90)` (dito nach rechts). Mit

```
penup()
setpos(200,100)
pendown()
```

kann der Stift zu einem geeigneten Startpunkt (hier: 200,100) bewegt werden.

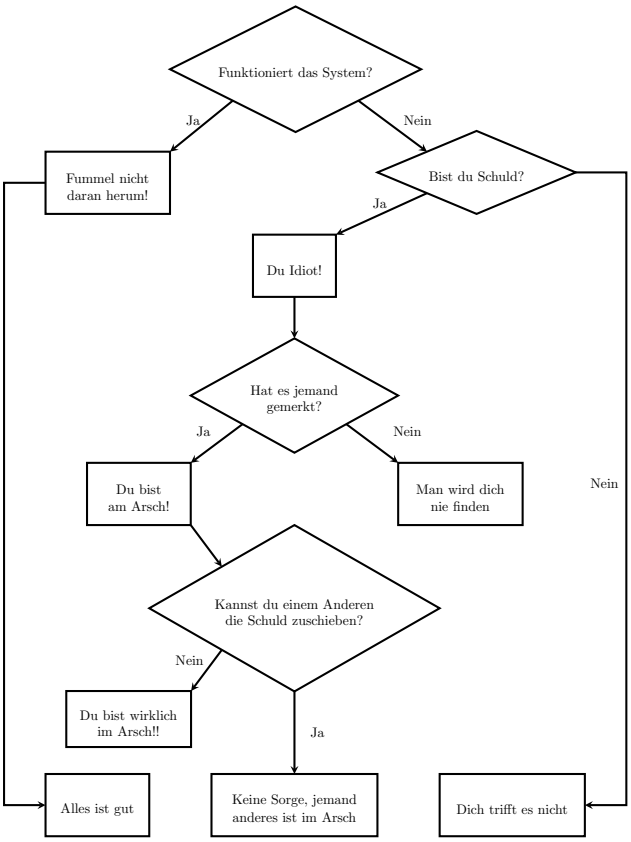
15 Kochkurve (Tag 3)

Schreibe ein Programm, das die Befehle zum Zeichnen der Kochkurve (siehe Wikipedia) der Ordnungen 0 bis 5 ausgibt. Die Befehle sind Strings aus F, L und R Zeichen, wobei F bedeutet "zeichne eine Linie eine Einheit vorwärts" und L bzw. R für "drehe dich um 60° nach links bzw. rechts" steht. Der Schlüssel zum Lösen des Problems ist die Beobachtung, dass die Kurve der Ordnung n genau eine Kurve der Ordnung $n-1$ gefolgt von einem L gefolgt von der Kurve der Ordnung $n-1$ gefolgt von einem RR gefolgt von der Kurve der Ordnung $n-1$ gefolgt von einem L gefolgt von der Kurve der Ordnung $n-1$ ist.
Die Kodierung der Kochkurven der Ordnung 0,1,2:
F
FLFRRFLF
FLFRRFLFLFLFRRFLFRRFLFRRFLFLFLFRRFLF

(Noch besser sieht das Ergebnis aus, wenn man mit FRRFRRF startet)

16 Flussdiagramm (Bonus Tag 1)

Die Firma *It wasn't me* hat folgende Anweisung für ihre Mitarbeiter ausgegeben.



Erstelle ein Computerprogramm anhand des Ablaufplans.

17 Primzahlen sieben (Bonus Tag 1)

Finde alle Primzahlen von 2 bis 1000
Hinweis: Falls du das Prinzip nicht kennst, es nennt sich:"Sieb des Eratostenes". Wikipedia kann auch hier hilfreich sein.

18 Vigenere-Code (Bonus Tag 3)

Eine sehr viel sicherere Verschlüsselungsmethode als der Cäsarcode ist der Vigenere-Code (s. wikipedia). Dabei wird ganz analog zum Cäsarcode verschlüsselt (siehe Aufgabe 11), aber es werden statt eines festen ks verschiedene ks verwendet, die aus einem Schlüsswelwort hervorgehen. Haben wir als Schlüsselwort z.B. 'key', dann verschlüsseln wir den ersten Buchstaben des zu verschlüsselnden Worts mit dem Cäsarcode mit $k = 10$ (für 'k'), den zweiten Buchstaben des zu verschlüsselnden Worts mit dem Cäsarcode mit $k = 4$ (für 'e'), den dritten Buchstaben mit $k = 24$ (für 'y'), den vierten Buchstaben wieder mit $k = 10$ (für 'k'), usw.