1. (4 Punkte) (a) Erkläre den Unterschied zwischen Syntax und Semantik einer Sprache.

   (b) Welche der folgenden Python-Zeilen enthalten einen Syntaxfehler? Was passiert bei den anderen beim Ausführen?

```
x = 3 +
print(10 / 0)
y = 'a' + 2
if x > 2 print(x)
z = [1, 2, 3][5]
```

Lösung:

(a) Die Syntax legt fest, welche Zeichenfolgen korrekt gebildet sind (die Form). Die Semantik legt fest, was eine korrekt gebildete Zeichenfolge bedeutet bzw. bewirkt (den Inhalt). Grammatiken beschreiben nur die Syntax.

(b) Syntaxfehler: `x = 3 +` (nach dem Plus fehlt ein Ausdruck) und `if x > 2 print(x)` (der Doppelpunkt fehlt). Die anderen Zeilen sind syntaktisch korrekt, führen aber beim Ausführen zu Fehlern: Division durch null (`ZeroDivisionError`), String plus Zahl (`TypeError`), Index 5 gibt es nicht (`IndexError`).

2. (5 Punkte) Gegeben ist die Grammatik mit $V = \{S, B\}$, $\Sigma = \{a, b, c\}$, Startsymbol S und den Regeln

$S \to aSc \mid B$
$B \to bB \mid b$

   (a) Gib eine Linksableitung für das Wort *aabcc* an.

   (b) Zeichne den Ableitungsbaum zu *aabcc*.

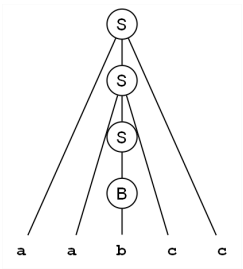
   (c) Welche der Wörter *abc*, *aabc*, *abbbc*, *ac*, *aabbcc* gehören zur Sprache?

   (d) Beschreibe die Sprache in Mengenschreibweise.

Lösung:

(a) $S \Rightarrow aSc \Rightarrow aaScc \Rightarrow aaBcc \Rightarrow aabcc$

(b)



(c) *abc*, *abbbc*, *aabbcc*. Bei *aabc* und *ac* passt die Anzahl der c nicht bzw. es fehlt das b.

(d) $L = \{a^n b^m c^n \mid n \geq 0,\ m \geq 1\}$

3. (4 Punkte) Gegeben ist die Grammatik mit $V = \{S, A, B\}$, $\Sigma = \{a, b\}$, Startsymbol S und den Regeln

   $S \to AB$
   $A \to aS \mid a$
   $B \to bA$

   (a) Welche der Wörter *aba*, *abab*, *abaaba*, *aabba*, *aaababaaba* gehören zur Sprache? Gib für ein Wort, das dazugehört und mehr als drei Zeichen hat, eine Ableitung an.

   (b) Begründe, warum in jedem Wort der Sprache doppelt so viele a wie b vorkommen.

Lösung:

(a) *aba*, *abaaba*, *aaababaaba*. *abab* endet auf b, aber jedes Wort endet auf a (das letzte Nichtterminal ist immer ein A). *aabba* enthält bb, aber auf jedes b folgt ein A, also ein a.
$S \Rightarrow AB \Rightarrow aB \Rightarrow abA \Rightarrow abaS \Rightarrow abaAB \Rightarrow abaaB \Rightarrow abaabA \Rightarrow abaaba$

(b) Jede Anwendung von $S \to AB$ und $B \to bA$ erzeugt ein b und zwei A. Jedes A wird zu genau einem a (mit $A \to aS$ entsteht dabei ein neues S). Also gehören zu jedem b genau zwei a.

4. (3 Punkte) Gegeben sind das Alphabet $\Sigma = \{a, b, c\}$ und die Sprache $L = \{a^i b^j c^k \mid i, j, k > 0\}$. Gib eine Grammatik an, die L erzeugt.

Lösung: Variablen $V = \{S, A, B, C\}$, Startsymbol S, Regeln:

$S \to ABC$
$A \to a \mid aA$
$B \to b \mid bB$
$C \to c \mid cC$

5. (3 Punkte) Gegeben sind das Alphabet $\Sigma = \{a, b, c\}$ und die Sprache $L = \{a^i b^j c^{i+j} \mid i, j \geq 0,\ i + j > 0\}$. Gib eine Grammatik an, die L erzeugt.

Lösung: Variablen $V = \{S, B\}$, Startsymbol S, Regeln:

$S \to aSc \mid B \mid ac$
$B \to bBc \mid bc$

Jedes a und jedes b wird zusammen mit einem c erzeugt, von außen nach innen.

6. (3 Punkte) Gegeben ist das Alphabet $\Sigma = \{0, 1, 2\}$. Gib eine kontextfreie Grammatik für die Sprache $L = \{0^n 1^{2n} 2^m \mid n, m \geq 0\}$ an.

Lösung: Variablen $V = \{S, A, B\}$, Startsymbol S, Regeln:

$S \to AB$
$A \to 0A11 \mid \varepsilon$
$B \to 2B \mid \varepsilon$

A erzeugt den Teil $0^n 1^{2n}$, B den Teil 2^m.

7. (4 Punkte) Gegeben ist das Alphabet $\Sigma = \{a, b, c, d\}$. L ist die Sprache aller Wörter, in denen nie ein b direkt auf ein a folgt, nie ein c direkt auf ein b und nie ein d direkt auf ein c. Zum Beispiel liegt *acadbbda* in L, *acab* nicht. Gib eine Grammatik an, die L erzeugt.

Lösung: Variablen $V = \{S, A, B, C, D\}$, Startsymbol S. Das Nichtterminal merkt sich das zuletzt erzeugte Zeichen: Nach A darf kein b kommen usw.

$S \to aA \mid bB \mid cC \mid dD \mid \varepsilon$
$A \to aA \mid cC \mid dD \mid \varepsilon$
$B \to aA \mid bB \mid dD \mid \varepsilon$
$C \to aA \mid bB \mid cC \mid \varepsilon$
$D \to aA \mid bB \mid cC \mid dD \mid \varepsilon$

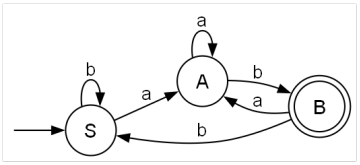
8. (3 Punkte) Beschreibe die Sprache, die von der folgenden Grammatik erzeugt wird: $V = \{S, A, B\}$, $\Sigma = \{0, 1\}$, Startsymbol S, Regeln:

$S \to 0B \mid 1A$
$A \to 0 \mid 0S \mid 1AA$
$B \to 1 \mid 1S \mid 0BB$

Lösung: Die Sprache aller nichtleeren Wörter über Σ, in denen gleich viele Nullen wie Einsen vorkommen.

Aus A entstehen genau die Wörter mit einer Null mehr als Einsen, aus B die mit einer Eins mehr als Nullen. ε gehört nicht dazu, weil jede Regel für S ein Zeichen erzeugt.

9. (4 Punkte) Der DEA akzeptiert die Wörter über $\{a, b\}$, die auf *ab* enden.



(a) Gib eine reguläre Grammatik an, die dieselbe Sprache erzeugt.

(b) Gib eine Ableitung für *baab* an und erkläre, wie sie mit dem Lauf des Automaten zusammenhängt.

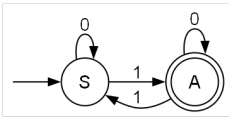
Lösung:

(a) Jeder Zustand wird ein Nichtterminal, jeder Übergang eine Regel, der Endzustand bekommt eine ε-Regel. Startsymbol S.
$S \to aA \mid bS$
$A \to aA \mid bB$
$B \to aA \mid bS \mid \varepsilon$

(b) $S \Rightarrow bS \Rightarrow baA \Rightarrow baaA \Rightarrow baabB \Rightarrow baab$
Das Nichtterminal am Ende jeder Satzform ist der Zustand, in dem der Automat nach dem Lesen der bisher erzeugten Zeichen ist: S, S, A, A, B. Weil B ein Endzustand ist, darf es mit ε verschwinden.

10. (3 Punkte) Gegeben ist die reguläre Grammatik mit $V = \{S, A\}$, $\Sigma = \{0, 1\}$, Startsymbol S und den Regeln
$S \to 0S \mid 1A$
$A \to 0A \mid 1S \mid \varepsilon$
Zeichne einen endlichen Automaten, der dieselbe Sprache akzeptiert, und beschreibe die Sprache.

Lösung: Nichtterminale werden Zustände, S der Startzustand. $A \to \varepsilon$ macht A zum Endzustand.



Die Sprache besteht aus allen Wörtern mit einer ungeraden Anzahl von Einsen.

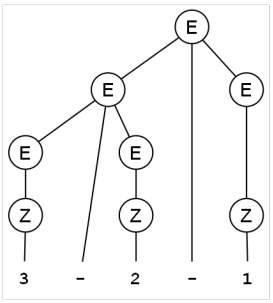
11. (5 Punkte) Gib für jede Grammatik den größten Typ in der Chomsky-Hierarchie an (3, 2, 1 oder 0) und begründe. Startsymbol ist jeweils S.
G_1: $S \to aS \mid bA, \quad A \to b \mid \varepsilon$
G_2: $S \to aSb \mid ab$
G_3: $S \to aSBc \mid abc, \quad cB \to Bc, \quad bB \to bb$
G_4: $S \to AB, \quad AB \to a$
   (a) Typen von G_1 bis G_4
   (b) Welche Sprache erzeugt G_4? Welchen Typ hat diese Sprache?

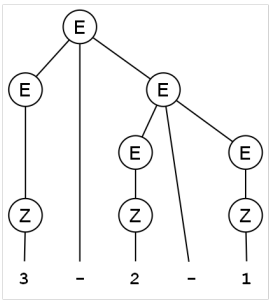
Lösung:
   (a) G_1: Typ 3, alle Regeln haben die Form $A \to aB$ oder $A \to \varepsilon$ bzw. $A \to b$.
   G_2: Typ 2, links steht immer ein Nichtterminal, aber aSb hat ein Terminal hinter dem Nichtterminal.
   G_3: Typ 1, links stehen teilweise zwei Zeichen (cB, bB), aber keine Regel verkürzt. (Die Grammatik erzeugt $\{a^n b^n c^n \mid n \geq 1\}$.)
   G_4: Typ 0, die Regel $AB \to a$ verkürzt die Satzform.
   (b) $L(G_4) = \{a\}$. Diese Sprache ist regulär (Typ 3), z.B. mit $S \to a$. Der Typ einer Sprache ist der größte Typ einer Grammatik, die sie erzeugt.

12. (5 Punkte) Gegeben ist die Grammatik mit Startsymbol E und den Regeln $E \to E - E \mid Z$, $Z \to 1 \mid 2 \mid 3$.
    (a) Zeichne zwei verschiedene Ableitungsbäume für das Wort $3 - 2 - 1$.
    (b) Welche Werte ergeben sich, wenn man die Bäume als Rechenweg liest? Welcher Baum passt zur üblichen Rechnung?
    (c) Gib eine eindeutige Grammatik für dieselbe Sprache an, deren Ableitungsbaum zur üblichen Rechnung passt.

Lösung:

(a)





(b) Links: $(3 - 2) - 1 = 0$. Rechts: $3 - (2 - 1) = 2$. Üblich wird von links nach rechts gerechnet, das ist der linke Baum.

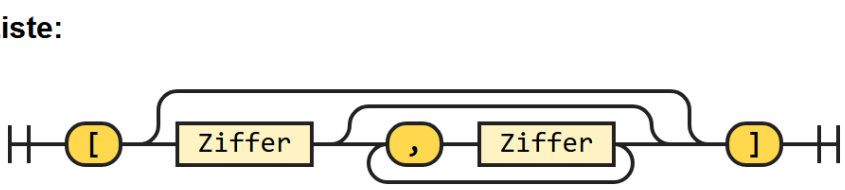
(c) $E \to E - Z \mid Z$, $Z \to 1 \mid 2 \mid 3$. Rechts vom Minus steht immer nur eine Zahl, deshalb wird zuerst links gerechnet.

13. (4 Punkte) Gesucht ist eine Beschreibung für Python-Listen aus einzelnen Ziffern ohne Leerzeichen, z.B. `[]`, `[3]`, `[1,2,7]`. Nicht erlaubt sind z.B. `[1,]`, `[,]` und `[12]`.
    (a) Gib eine EBNF an. Du darfst `Ziffer ::= [0-9]` verwenden.
    (b) Zeichne das Syntaxdiagramm.

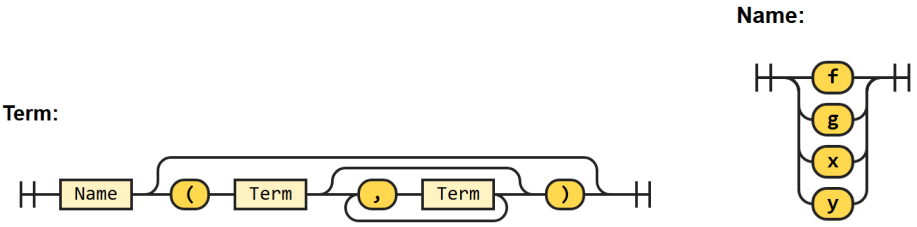
Lösung:

(a)
```
Liste  ::= '[' ( Ziffer ( ',' Ziffer )* )? ']'
Ziffer ::= [0-9]
```

(b)



14. (5 Punkte) Gegeben sind die Syntaxdiagramme:



(a) Welche der folgenden Wörter sind ein `Term`? `f(x)`, `f()`, `g(x,f(y,x))`, `f(x,)`, `(x)`, `fg(x)`, `x`

(b) Gib eine EBNF an.

(c) Kann man die Sprache mit einem regulären Ausdruck beschreiben? Begründe.

Lösung:

(a) Ein Term sind `f(x)`, `g(x,f(y,x))`, `x`. Keine Terme sind `f()` (in der Klammer muss mindestens ein Term stehen), `f(x,)` (nach dem Komma fehlt ein Term), `(x)` (ein Term beginnt mit einem Namen) und `fg(x)` (ein Name ist nur ein Zeichen).

(b)
```
Term ::= Name ( '(' Term ( ',' Term )* ')' )?
Name ::= 'f' | 'g' | 'x' | 'y'
```

(c) Nein. Terme können beliebig tief geschachtelt sein, z.B. $f(f(f(\ldots x \ldots)))$. Dann müssen gleich viele öffnende wie schließende Klammern vorkommen. Das kann ein endlicher Automat nicht prüfen, die Sprache ist nicht regulär. Man sieht es auch an der Rekursion: `Term` benutzt sich selbst.

15. (4 Punkte) Erweitere den Parser aus dem Notebook so, dass er auch Subtraktion und Division kann:

```
Ausdruck ::= Term ( ( '+' | '-' ) Term )*
Term     ::= Faktor ( ( '*' | '/' ) Faktor )*
```

Die Funktionen `faktor`, `zahl` und `auswerten` bleiben unverändert.

Was ergeben `auswerten('10-4-3')` und `auswerten('8/4/2')`?

Lösung:

```
def ausdruck(s, i):
    wert, i = term(s, i)
    while i < len(s) and s[i] in '+-':
        op = s[i]
        w, i = term(s, i + 1)
        if op == '+':
            wert = wert + w
        else:
            wert = wert - w
    return wert, i

def term(s, i):
    wert, i = faktor(s, i)
    while i < len(s) and s[i] in '*/':
        op = s[i]
        w, i = faktor(s, i + 1)
        if op == '*':
            wert = wert * w
        else:
            wert = wert / w
    return wert, i
```

`auswerten('10-4-3')` ergibt 3 und `auswerten('8/4/2')` ergibt 1.0: Die Schleife rechnet von links nach rechts, $(10-4)-3$ und $(8/4)/2$.