Christian Lück

XML und Textkodierung

Mit einer Einführung in die Programmierung für Geisteswissenschaftler*innen

Fakultät für
Kultur- und Sozialwissen-schaften

FernUniversität in Hagen

Inhalt

2. Glyphen

2.1. Glyphe vs. Graph

Bislang wurde lediglich die Repräsentation von Zeichen in Nullen und Einsen behandelt. Schriftzeichen haben natürlich auch noch eine graphische Darstellung, die sogenannte Glyphe (von gr. γλυφή, wörtlich *Eingeritztes*). Die ägyptischen Hieroglyphen, die demotische Inschrift und die griechische Übersetzung auf der Stehle von Rosetta: alles Glyphen, wenngleich aus ganz verschiedenen Schriftsystemen. (Abb. 2.2) Nicht nur die Schriftzeichen zur Repräsentation von Sprache haben eine graphische Darstellung: Auch die graphischen Darstellungen der arabischen Ziffern sind Glyphen.

Abb. 2.1.: Verschiedene Glyphen des großen lateinischen Buchstaben <A>. In der Mitte die Glyphe aus der in Studienbriefen verwendeten Frutiger-Schrift. Ein gute Wahl? Oder nicht?

Zum Begriff der Glyphe gehört, dass es eine *konkrete graphische Erscheinungsform* eines Zeichens ist. (Abb. 2.1) Hingegen handelt es sich beim *Zeichen* um ein abstraktes Konzept, auch wenn man den Begriff Zeichen im Sinne des Graphs (von gr. γραφή, Schrift) auf die graphische Erscheinung verengt. Für den Computer (und für die alte Gutenberg-Druckerei) heißt das also: Eine Glyphe z. B. für den kleinen lateinischen Buchstaben <a> ändert sich mit der Schriftart, in der es am Bildschirm ausgegeben oder gedruckt wird. Das <a> in Times New Roman ist eine andere Glyphe als in Helvetica, in Garamond oder Frutiger (der Schrift, aus der dieser Kurs gesetzt ist). Auch mit der Schriftgröße ändert sich die Glyphe.

Um eine in Bytes kodierte Zeichenkette am Bildschirm ausgeben oder drucken zu können, muss also ein Satz von Glyphen zur Verfügung stehen, aus dem die zu einer Bitfolge passende graphische Darstellung ausgewählt werden kann. Diese graphische Erscheinung war bei einer Reihe alter Peripherie-Geräte in denselben abgelegt: Man denke etwa an den Typenraddrucker. Video-Controller (Grafik-Karten) haben Glyphen in Form einer Punkt-Matrix in einem nichtflüchtigen Baustein, in einem ROM, gespeichert; sie werden für den Textmodus, der wenigstens direkt nach dem Start eines PC aktiv ist, benötigt. Im Grafik-Modus hingegen, in dem jedes Pixel eines Displays individuell adressierbar ist, übernimmt die Software eines Computers die Ausgabe von Glyphen. In diesem Modus sind Glyphen beliebiger Schriftarten miteinander auf dem Bildschirm kombinierbar. Im Textmodus hingegen ist die Ausgabe auf die schönen Glyphen in der Punktmatrix der Grafikkarte festgelegt.

2.2. Glyphenlos: Textdateien

Im Dateiformat eines Textverarbeitungsprogramms wird (wenigstens potentiell) zu jedem Zeichen auch noch gespeichert, mit welcher Glyphe (Schriftart, Schriftgrad) es dargestellt werden soll. Bei sogenannten Textdateien, in denen die Zeichen zugelassen sind, die in einer Kodierung (vgl. Kapitel 1) definiert sind,

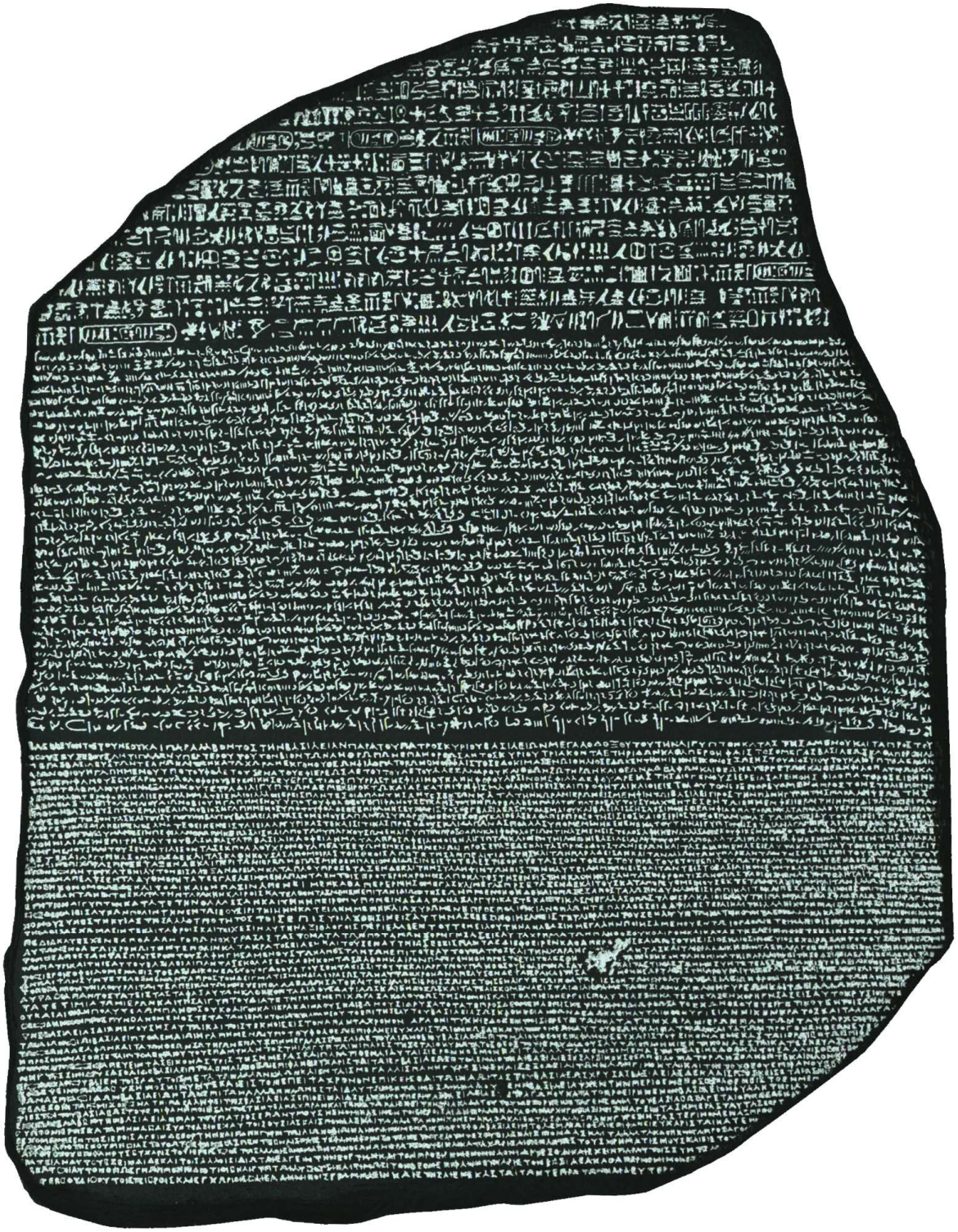

Abb. 2.2.: Der Stein von Rosetta hat die Entzifferung der ägyptischen Hieroglyphen (oberes Drittel) ermöglicht, weil er dasselbe Gesetz in drei verschiedenen Schriftsystemen in Stein präsentiert. Insbesondere an Namen konnte man so die teilweise phonographische Verwendung der Hieroglyphen nachweisen. Quelle: `https://de.wikipedia.org/wiki/Stein_von_Rosette` Die spannende Geschichte der Dechiffrierung der Hieroglyphen können Sie nachlesen bei David Kahn: The Codebreakers. A Story of Secret Writing. New York: Scribner, 1991, S. 905–910.

ist dies nicht der Fall. Dort werden nur die Bitfolgen der Zeichen gespeichert. Von der graphischen Erscheinung ist also abstrahiert. Das Programm, mit dem man sie bearbeitet – vorzugsweise ein Text-Editor, vgl. Anhang A.1 – bestimmt den Satz der zur Anzeige verwendeten Glyphen dann selbst.

Bei der Kodierung von Texten für die Digital Humanities setzt man ganz auf solche Textdateien. – Dennoch muss hier geklärt werden, was Glyphen sind: einerseits um den Unterschied zwischen abstrakten Zeichen (Graph) und konkreten graphischen Erscheinungen (Glyphe) zu verstehen, andererseits um zu den Prinzipien der Textkodierung in den Digital Humanities zu gelangen.

Die Gestalt der Zeichen wird in den Digital Humanities im Allgemeinen für kein wichtiges Merkmal eines Textes gehalten.[1] Es geht ihnen nicht um die Gestalt der Zeichen, sondern um die Zeichen als abstrakte Idee; nicht um Zeichen als Glyphen, sondern als Symbole. Der Symbolwert, der darin begründet ist, dass man die Zeichen voneinander unterscheiden kann (Differenz), ist mit einer der im vorigen Kapitel beschriebenen Kodierungen vollständig repräsentiert.

Aufgabe 28: Geben Sie Beispiele für Medien, in denen Schrift bzw. Text ohne Glyphen übertragen wird.

2.3. PDF

Das *Portable Document Format*, kurz PDF, erfüllt manche Anforderungen, die die Digital Humanities an ein Format für ein Forschungskorpus stellen: Für praktisch alle Betriebssysteme gibt es Programme, die PDFs darstellen können, und seitdem es in einer ISO-Norm normiert worden ist, kann es, auch wenn die Firma Adobe es erfunden hat, nicht mehr als proprietär gelten: Die akademische Community würde sich also nicht in Abhängigkeit einer Firma begeben.[2]

Man könnte also auf die Idee kommen, dass das PDF ein geeignetes Format für die Kodierung, Archivierung und (maschinelle) Weiterverarbeitung bzw. Auswertung von Text sei. Aber dem ist nicht so. PDF ist ein Format, mit dem insbesondere das Aussehen, das typographische *Erscheinungsbild* festgelegt wird. Es werden Glyphen und ihre Positionen auf einer Seite festgelegt. Und das Format stellt insbesondere auch Mittel bereit, sicherzustellen, dass die Glyphen bei der Anzeige im PDF-Betrachter auch verwendet werden, nämlich durch die sogenannte Einbettung von Schriftarten, aus denen die Glyphen ausgewählt worden sind. Anders als bei Textverarbeitungsprogrammen wie MS Word oder LibreOffice spielt es dann keine Rolle mehr, ob die Schriftart auf dem betreffenden Computersystem installiert ist oder nicht: Sie wird vielmehr in der PDF-Datei selbst übermittelt. PDF ist kein Format zum langfristigen Speichern von Texten, sondern zum Fixieren ihrer graphischen Präsentation.

Im Zentrum der Funktionsweise dieses Formats steht ein noch älteres Format namens *PostScript*. Dies ist eine Programmiersprache, die speziell auf das exakte Beschreiben des Aufbaus bzw. Aussehens einer Seite (Buchseite, zu druckende Seite) hin entworfen worden ist. Werfen wir einen kurzen Blick in den Programmcode dieser Seitenbeschreibungssprache. Hier ein kurzer Auszug aus einer beliebigen Datei. Jedes PDF enthält solche Ausdrücke.

```
14 0 obj
<</Length 8031>>stream
q
BT
/F2B 9.963 Tf 14.4 652.437 Td[(\334ber)-250(dieses)-250(Buch)]TJ/F2 9.963
Tf 0 -21.669 Td[(Dies)-296(ist)-297(ein)-296(digitales)
```

[1] Freilich gibt es Ausnahmen. Sofern die Gestalt der Zeichen doch für ein wichtiges Merkmal gehalten wird, soll sie möglichst mit symbolischen Mitteln beschrieben, aber nicht nur foto-mechanisch reproduziert werden. Zumindest muss dann mit symbolischen Mittel auf die Reproduktion hingewiesen werden.

[2] »[B]e easy for researchers to use without special-purpose software«, ist eines der Ziele der Initiative der geisteswissenschaftlichen Forschungs-Community zur Textkodierung, Text Encoding Initiative, ed.: Design Principles for Text Encoding Guidelines. 1988. `https://www.tei-c.org/Vault/ED/edp01.htm` [Stand: 09. 05. 2019], Abschnitt 4

Sie sehen eine eher kryptische Kette von Zeichen. Das sind Anweisungen in der Programmiersprache PostScript. Vielleicht haben sie bemerkt, dass mitten in den Anweisungen ein paar deutsche Wörter auftauchen, die jeweils in Klammern stehen: »dieses«, »Buch«, »Dies«, »ist«, »ein« und »digitales«. Den Inhalt des ersten Klammer-Ausdrucks könnten wir als »Über« lesen, wenn wir die »\334« als Kodierung des Umlauts interpretieren.

Aber was fällt verglichen mit den uns bislang begegneten Zeichenketten auf? Richtig: Es gibt kein Leerzeichen zwischen den Wörtern! An die Stelle des Leerzeichens sind hier Ausdrücke aus Minus und einer Zahl getreten (und gegebenenfalls sind auch die Klammern Indizien für Wortgrenzen). Wenn wir die Zahlen-Ausdrücke zwischen den Klammer-Ausdrücken als Spatien auffassen, wobei die Zahl den Abstand zum nächsten eingeklammerten Wort beschreibt, dann haben wir es hier mit einem anderen Konzept des Spatiums zu tun, als es das Leerzeichen war. Hier sind die Spatien kein Symbol, nicht $0x20$, sondern ganz ähnlich wie in Gutenbergs Druckerei wird hier Blindmaterial zwischen die Wörter eingefügt. – Vielleicht ist folgender Satz etwas überpointiert, aber im Hinblick auf den in Abschnitt 1.3 geschärften Begriff des Leerzeichens durchaus richtig: **Das PDF kennt das Konzept des Leerzeichens nicht.** <SP> ($0x20$) wird hier als Trennzeichen für die PostScript-Befehle verwendet.

Schon das allein schließt das PDF-Format für die Digital Humanities aus. Es bringt nämlich dann erhebliche Schwierigkeiten mit sich, wenn man aus einer PDF-Datei den Text als Reihe von Symbolen auslesen will (Text-Extraktion), z. B. um Wörter und Sätze mit (computer-) linguistischen Methoden zu untersuchen. Programme zur Text-Extraktion müssen eine Heuristik verwenden, um die Wortgrenzen zu bestimmen. Denn bisweilen werden die Zeichen eines Wortes mit etwas größerem Abstand gesetzt, wie im Falle einer Hervorhebung mit S p e r r d r u c k. Dies soll auch als ein einziges Wort erkannt werden, führt aber regelmäßig zu unakzeptablen Ergebnissen bei der Text-Extraktion. Die Liste der Phänomene eines Textes, in denen die bloßen Lage-Daten von Zeichen, die das PDF enthält, als Information nicht hinreichen, ist oft noch viel länger. Für Schriftstücke, die nicht digital-born sind, etwa die von Google-Books gescannten Bücher, sind die Schwierigkeiten bei der Text-Extraktion noch viel größer. Deswegen gibt es derzeit auch kaum Forschungsprojekte, die mit dem sehr großen Archiv von Google-Books arbeiten.[3]

Aufgabe 29: Suchen Sie mit der Suchfunktion Ihres PDF-Betrachters im PDF dieses Kurses nach dem in Sperrdruck gesetzten Wort Sperrdruck! Finden Sie es?

Um an dieser Stelle bereits einmal auf spätere Abschnitte des Kurses vorzugreifen: Textrepräsentation in den Digital Humanities soll eher folgendermaßen aussehen.

```
im Fall einer Hervorhebung durch <gesperrt>Sperrdruck</gesperrt>.
```

[3] Für gescannte Bücher hat das PDF allerdings den Vorteil, dass das gescannte Bild der Buchseite und die durch Optical Character Recognition (OCR) bestimmten Zeichen zusammen in einer Datei und somit im Verbund auswertbar bleiben. Das ist für diverse Aufgaben der Text-Extraktion, z. B. von Tabellen, unerlässlich.

2*. Programmieren

In diesem Kapitel werden zunächst die Zeichenketten, Listen und Wörterbücher vorgestellt. Bei allen handelt es sich um (dynamische) Datenstrukturen beliebiger Länge, die jeweils mehrere Objekte (einzelne Zeichen bei Zeichenketten) speichern können. Sie werden beim Programmieren immer wieder benötigt. Anschließend wird mit der For-Schleife die erste sogenannte Kontrollstruktur eingeführt, also ein Gebilde, das den Ablauf des Programms steuert. Die For-Schleife dient zur Verarbeitung der vorher eingeführten Datenstrukturen beliebiger Länge. Die Programmbeispiele dienen zum Zählen von Elementen bzw. Merkmalen von Elementen. Zum Abschluss werden die Schleifen wiederverwendbar gemacht, indem sie in selbst definierten Funktionen abgelegt werden.

2*.1. Strings: Länge, Index, Slices

Eine Zeichenkette ist eine Folge von Zeichen. Es wäre praktisch zu wissen, wie lang eine Zeichenkette ist und auf einzelne Zeichen oder Bereiche zugreifen zu können. Das geht natürlich alles. Die eingebaute Funktion `len()` gibt die Länge einer Zeichenkette (oder einer Liste) an; die Funktion wertet zum Typen Integer aus:

```
s = "Pythons gehören zur Familie der großen Schlangen."
len(s)
```

```
49
```

Mit einer Ganzzahl n in einer eckigen Klammer hinter der Zeichenkette lässt sich auf das Zeichen Nummer $n+1$ der Zeichenkette zugreifen. (Indexing) Das erste Zeichen der Zeichenkette hat also den Index 0. Man spricht auch von null-indizierten Strings.

```
s[0]
```

```
'P'
```

```
s[1]
```

```
'y'
```

```
s[len(s)-1]
```

```
'.'
```

Aufgabe 30:

a) Erklären Sie die letzte Code-Zeile und das Ergebnis ihrer Auswertung.
b) Was passiert bei `s[len(s)]`?

Ein negativer Index zählt von hinten:

```
s[-1]
```

```
'.'
```

```
s[-2]
```

```
'n'
```

```
s[-len(s)]
```

```
'P'
```

Aufgabe 31: Erklären Sie die letzte Code-Zeile! Vergleichen Sie mit der Code-Zeile `s[len(s)-1]`!

Mit zwei Zahlen, getrennt durch ein Colon, kann man Start- und End-Index eines Bereichs angeben:

```
s[0:6]
```

```
'Python'
```

```
s[-10:-2]
```

```
'Schlange'
```

```
s[-10:len(s)]
```

```
'Schlangen.'
```

Aufgabe 32: Erklären Sie erneut die letzte Code-Zeile und deren Auswertung!

Es gibt eine ganze Reihe von Funktionen, die auf Zeichenketten angewandt werden können. Wir werden sie hier nicht weiter behandeln, sondern nach und nach einige davon selbst implementieren. Das dient der Übung im algorithmischen Denken. Erst ab Abschnitt 4*.3 werden die vorhandenen String-Funktionen verwendet.

2*.2. Listen

Strings sind der Datentyp für Folgen von Zeichen. Das Konzept lässt sich noch verallgemeinern zu Typen von Folgen beliebiger Objekte. Allgemein spricht man (zumindest im Kosmos von Python) von Sequenztypen, engl. *sequence types*. Die wichtigsten Sequenztypen neben der Spezialsequenztyp String sind Listen und Tupel. Listen und Tupel können Folgen beliebiger Objekte speichern. Alle Sequenztypen zeichnen sich dadurch aus, dass die Objekte eine Reihenfolge haben (im Unterschied zu Mengen).

Man konstruiert Listen u. a. mit Hilfe von eckigen Klammern und Kommata als Trennzeichen zwischen den Listen-Elementen:

```
l = [0, 2, 4, 6, 8, 10]
```

```
l
```

```
[0, 2, 4, 6, 8, 10]
```

Wie bei Strings stehen der Operator `in` zur Verfügung, um das Enthaltensein eines Elements zu testen:

```
2 in l
```