Coding for lawyers?!

Tobias Pesch/Victoria Fricke, Düsseldorf*

Muss jede Juristin programmieren lernen? Ganz sicher nicht. Können alle Juristen davon profitieren? Auf jeden Fall! Warum es für Juristinnen sinnvoll ist, sich mit dem Programmieren auseinanderzusetzen und wie man damit anfangen kann.

Es gibt Juristen,[1] für die wäre es eine reine Zeitverschwendung, sich mit dem Thema Programmieren auseinanderzusetzen. Wer sich nicht für Technologie begeistern kann und im beruflichen Alltag wenig Bezug dazu hat, kann seine wertvolle Freizeit sinnvoller verbringen. Bei vorhandener Neugier bezüglich Legal Tech können wir aber nur dazu ermuntern, sich trotz der immer mächtiger werdenden No-Code Tools auch einmal mit den Grundzügen der Programmierung zu befassen. Es lohnt sich!

Zunächst stellt sich die Frage, wo Programmieren anfängt. Mag man zunächst an dunkle Kellerräume und bärtige Nerds in kurzärmligen Karohemden denken, die vor dem dunklen Bildschirm komplexe mathematische Zahlenreihen produzieren, so ist dies sicherlich nicht das Bild, was dem modernen Arbeiten entspricht. Um vorweg direkt eine Sorge vieler Juristen zu nehmen: Beim Programmieren geht es nicht ums Rechnen, denn die Rechenarbeit übernimmt der Computer.[2] Für Juristinnen mit einer Betätigung auf Schnittstellenpositionen, die auch die Entwicklung von Software beinhalten, dürfte es im Regelfall nicht um die Programmierung von Blockchain-Anwendungen und Smart Contracts oder das Trainieren von selbstlernenden Algorithmen gehen, sondern um die Übersetzung von Workflows in kleinere Tools zur Prozessoptimierung und dem daraus resultierenden komfortablerem und effizienterem Arbeiten.

A. Warum lohnt es sich, programmieren zu lernen?

Programmieren können bedeutet in erster Linie, zu verstehen, wie Programme und Programmiersprachen funktionieren. Die tägliche Arbeit wird sich im Weiteren oft weniger mit dem Programmieren selbst beschäftigen, sondern eher einen Vorteil dadurch erzielen, dass Prozesse verstanden und infolgedessen vereinfacht, verbessert und automatisiert werden können. Dies fängt schon damit an, das Optimum aus den gängig genutzten Programmen wie Microsoft Word und Excel/Power-Query herauszuholen – oder eventuell übergriffige E-Mail-Verteiler mit Outlook-Regeln zu bändigen –, kann aber noch bedeutend weitergehen.

Dabei geht es nicht um die Vorstellung, dass ein Jurist dauerhaft an Software-Projekte gebunden ist. Vielmehr bringt das Verständnis der Programmiersprache und -denkweise mit sich, dass man in der Lage ist, bei neuen juristischen Projekten und Mandaten im Vorhinein den effizienten Einsatz von informatorischen Hilfsmitteln zu bedenken, um den Projektfortschritt dadurch zu beschleunigen.[3] Was man nicht kennt, kann man schließlich auch nicht nutzbar machen.

I. Interdisziplinäre Zusammenarbeit

Dass auch die Arbeit in interdisziplinären Teams einwandfreier funktioniert, wenn man sich gegenseitig versteht und die Sprache des anderen spricht, liegt auf der Hand. Aus Unternehmen, aber auch aus Kanzleien und selbst der Verwaltung ist die Arbeit mit Informatikern und IT-Spezialisten nicht fortzudenken. Selbiges gilt in alltäglichen Situationen, wie der Arbeit mit dem IT-Support. Das Problem kohärent beschreiben zu können ist meist schon der halbe Weg zu einer schnellen Lösung. Dabei in einer ähnlichen Art wie die Expertinnen zu denken, vereinfacht die Kommunikation und beschleunigt die Lösung immens.

Häufig wird in dieser Debatte angeführt, dass Juristen selbst nicht Programmieren können müssen, aber in der Lage sein sollten, sich in die Denkweise von Program-

* Tobias Pesch ist Associate im Kartellrecht bei White & Case LLP in Düsseldorf und Lehrbeauftragter für Legal Technology am College of Europe in Brügge. Victoria Fricke ist wissenschaftliche Mitarbeiterin im Insolvenzrecht bei White & Case LLP in Düsseldorf und Studentin im achten Semester an der Leibniz Universität Hannover.

1 Mit Juristinnen sind, der Situation der Autoren geschuldet, vor allem Anwältinnen gemeint. Auf alle anderen Berufszweige sind die Wertungen jedoch zu übertragen.

2 *Maurer*, Wie ich zum Coding Lawyer wurde, allaboutlegaltech.de, www.allaboutlegaltech.de/2018/10/wie-ich-zum-coding-lawyer-wurde, Abruf v. 13.7.2021.

3 *Miki*, Programming for Lawyers: Why Lawyers Make Good Programmers, clio.com, www.clio.com/blog/programming-for-lawyers, Abruf v. 13.7.2021.

mierern hineinzuversetzen.[4] Doch was bedeutet das? Man kann sich schlecht in etwas hineinversetzen, das man nicht versteht, geschweige denn durchdrungen hat. Oft erkennt man gar nicht, dass ein Problem existiert, weil man nicht sieht, wo eines liegen könnte.[5]

II. Mentalität

Beim Programmieren ist nicht nur das Ergebnis von großem Nutzen, bereits der Prozess lehrt einen sehr viel. Programmieren schult das ausnahmslos strukturierte, logische und problemlösende Denken und lehrt auf dem Weg hin zu ebendieser Lösung Kreativität zu zeigen. Oft kommt es vor, dass einem am Ende des Vorgangs ein Error ausgespuckt wird und man es erneut versuchen muss, bis am Ende – unter Umständen nach mehreren Anläufen – ein solider Programmcode entsteht. Dabei hilft Geduld und die in der Regel bereits aus dem juristischen Ausbildungsweg bekannte Frustrationstoleranz.

Zudem können wir Juristinnen uns eine wichtige Fertigkeit aus der Informatik-Branche abschauen: Beim Programmieren wird eine konstruktive Fehlerkultur geschult. Wenn ein Fehler in einem Programm gefunden wird, dann geht es nicht darum, den Schuldigen ausfindig zu machen, sondern vielmehr darum, (gemeinsam) zu einer effektiven Lösung des Problems zu gelangen. Auch die Konfrontation mit dem aus der Startup-Welt stammenden Shipping-beats-perfection-Ansatz[6] kann durchaus bereichernd sein.

III. Effizienzsteigerung

Grundkenntnisse im Programmieren sind keineswegs nur für Juristinnen relevant, die beispielsweise zu Haftungsfragen im IT-Recht oder zu Feinheiten des Datenschutzrechts beraten, sondern können grundsätzlich in jedem Rechtsgebiet von Bedeutung sein; eine besondere Eignung dürften wirtschaftsrechtliche Großprojekte in Rechtsgebieten wie dem Kartellrecht oder der Insolvenzverwaltung aufweisen. Vor allem beim Umgang mit großen Datenmengen, z. B. in Massenverfahren, ist es kaum noch denkbar oder erstrebenswert, die Arbeit vollständig manuell vorzunehmen. Zu groß wäre der Zeitaufwand, die Fehleranfälligkeit und zu wertvoll – gleichbedeutend mit teuer – ist die (Lebens-) Zeit der Juristinnen.

B. Können Juristen überhaupt programmieren lernen?

Regelmäßig hört man das Argument, Juristinnen hätten neben dem Studium, Referendariat oder Beruf überhaupt nicht ausreichend Zeit zur Verfügung, um Programmieren zu lernen – und im Zweifel werden es die IT-Spezialistinnen ohnehin immer besser können.

Der Umfang, den das Erlernen einer Programmiersprache aufweist, lässt sich kaum abstrakt definieren und bestimmen. Dabei kommt es natürlich auch darauf an, was die eigene Zielvorstellung ist. Zweifellos kann es kein realistisches – oder sinnvolles – Ziel für die meisten Juristen sein, den Aufwand eines vollen Bachelor- oder Master-Studiums der Informatik zu investieren, um sich im Zweitberuf als Softwareentwicklerin zu verwirklichen. Dafür sind beide Materien zu anspruchsvoll.
Bereits das Phänomen der sog. Coding Bootcamps zeigt, dass auch etwas weniger reichen kann: Dabei versprechen kommerzielle Schulungsanbieter,[7] dass sie innerhalb von etwa sechsmonatigen Vollzeitkursen zu Themen wie Web Development oder Data Science ihren (fachfremden) Kursteilnehmerinnen so viel Wissen vermitteln, dass diese danach realistische Chancen auf Bewerbungen für Einstiegsjobs als Junior Software Developer bzw. Junior Data Engineer haben. Aber auch dieser Zeit- und Kostenaufwand dürfte für die meisten Juristinnen, die keinen Berufswechsel in die IT-Branche auf Entwicklerseite anstreben, abschreckend und letztlich unrentabel sein.

Nach dieser Eingrenzung der Obergrenze wollen wir uns nun mit unserer unwissenschaftlichen Einschätzung hervorwagen, welche (Zeit-)Investition für Juristen realistisch und im Hinblick auf das Kosten/Nutzen-Verhältnis angemessen erscheint. Natürlich hängt der Aufwand auch ganz maßgeblich von etwaigen Begabungen im logischen Denken, Vorerfahrungen und dem Willen, Tempo und der Begeisterung zum Lernen ab.

Nach unserer Einschätzung genügt bereits eine Aktivlernzeit von 100 bis 150 Stunden – also ca. drei Stunden pro Woche über ein Jahr – um dauerhaft nützliche Fähigkeiten im Bereich der Programmierung zu erlernen. Dies dürfte in etwa dem Aufwand entsprechen, um in einer neuen Fremdsprache das GER-Niveau A2 zu erreichen;[8] auch die fachspezifische Fremdsprachenausbildung im Studium oder der theoretische Teil eines Fachanwaltskurses nehmen einen vergleichbaren Zeitrahmen in Anspruch. Mit anderen Worten: Alleine der Zeitmangel, sich neben der juristischen Betätigung noch fortzubilden, macht das Erlernen des Programmierens nicht unmöglich.

---

4 *Rose*, „Man muss ja nicht gleich anfangen zu coden – es reicht zu wissen, wie Digitalisierung funktioniert", lawtechrose.com, www.lawtechrose.com/post/man-muss-ja-nicht-gleich-anfangen-zu-coden-es-reicht-zu-wissen-wie-digitalisierung-funktioniert, Abruf v. 13.7.2021.

5 *Busch*, 5 Fragen an: Johannes Maurer, legaltechlab.de, legaltechlab.de/5-fragen-an-johannes-maurer, Abruf v. 13.7.2021.

6 Grob zu verstehen als „Lieber eine erste Version mit kleineren Fehlern veröffentlichen und aus dem Feedback der Nutzer lernen, statt den Code endlos perfektionieren zu wollen".

7 Bekannte europäische Anbieter sind unter anderem Le Wagon, Ironhack, CareerFoundry, General Assembly.

8 www.europaeischer-referenzrahmen.de, Abruf v. 13.7.2021.

Der größere Hinderungsgrund dürfte bei vielen Juristen die eigene (Fehl-)Vorstellung sein, mit den eigenen Talenten, Neigungen und Vorbildungen nicht für eine Karriere als Programmiererin prädestiniert zu sein. Dies ist jedoch gerade nicht die Zielvorstellung, die mit einem Coding Lawyer verbunden wird. Darüber hinaus sind sich die beiden Materien weniger fremd, als man meinen könnte: Bereits das juristische Studium schult in nicht zu unterschätzender Weise die Fähigkeit, in Strukturen zu denken.[9] Die Art und Weise, wie Gesetze aufgebaut sind und wie sie angewendet werden, ähnelt dem Grundgerüst eines Programms: Die Tatsache, dass zunächst alle Tatbestandsvoraussetzungen vorliegen müssen, um zur Rechtsfolge zu kommen, lässt sich in ähnlicher Art auch beim Programmieren in Wenn-Dann-Strukturen wiederfinden. Deshalb ist der Weg nicht mehr weit und vieles erscheint schnell vertraut.

C. Wie fange ich an?

Online gibt es eine Vielzahl von Anbietern, die einem, angepasst an den Zeitaufwand, den man bereit ist, zu investieren, kostenlose und kostengünstige Kurse zur Verfügung stellen.[10] Speziell für Juristinnen entwickelt wurde der Kurs CS50 for Lawyers der Harvard University, der einen spannenden Gesamtüberblick in die Thematik vermittelt.[11]

Zur Frage der besten Programmiersprache für Einsteiger gibt es erbittert geführte Diskussionen, die der Schärfe von juristischen Meinungsstreitigkeiten in nichts nachstehen. Am häufigsten genannt werden Python und Javascript, die sich zweifellos beide hervorragend eignen. Womit man beginnt, ist grundsätzlich unerheblich, da sich die Grundstrukturen der verschiedenen Sprachen ohnehin ähneln. Ein kleiner Vorteil für Python ist, dass mit Docassemble[12] ein sehr mächtiges Tool zum Abbilden juristischer Entscheidungsbäume als Open-Source-Projekt verfügbar ist, das seinerseits auf Python aufbaut.

D. Kleines Beispiel

Um zu demonstrieren, dass Programmieren nicht schwer ist, möchten wir an dieser Stelle einen kleinen Teaser basierend auf Python 3 anbieten:

I. First Steps

Direkt im Browser: Nutze Online-Umgebungen wie https://replit.com/languages/python3 oder http://colab.research.google.com.

Wer lieber auf dem eigenen PC arbeitet, muss zunächst die Python-Umgebung installieren.[13]

II. Programmcode

Der Programmcode besteht aus diversen Einzelteilen:

1. Kommentare

Darin enthalten sind die mit # begonnenen Zeilen, welche nicht direkt Teil des Codes sind, sondern lediglich Kommentare als Erklärungen enthalten.

2. Variablen

Variablen sind Datenfelder, die mit Inhalt gefüllt werden können, indem ihnen mittels = ein Wert zugewiesen wird.

Hierbei wird zwischen Datentypen wie int (Dezimalzahlen) und string (Zeichenkette, Text) unterschieden. Ein Beispiel: Wenn Variable1 die Zahl 4 ist (Variable1 = 4), ergibt die Rechnung 3 * Variable1 = 12; wenn Variable1 das Zeichen „4“ ist (Variable1 = “4“), ergibt die Rechnung 3 * Variable1 = „444“.

3. Ausgabe

Die Ausgabe des Codes erfolgt in der sog. Konsole. Der Befehl hierfür ist print().

4. Eingabe

Die Eingabe des Codes erfolgt im Eingabefeld. Der Befehl hierfür ist input().

5. Bedingungen

Der eigentlich (für uns) wichtige Part sind die Bedingungen. Wenn die Bedingung X erfüllt ist, dann passiert Y. Das ist vergleichbar mit dem Vorliegen aller Tatbestandsvoraussetzungen, die eine Rechtsfolge nach sich ziehen. Wenn nicht, dann passiert etwas Anderes entspricht der Verneinung einer Tatbestandsvoraussetzung. In den Programmiersprachen werden hierfür if und else verwendet.

Mögliche Bedingungsprüfungen können sein:
a == b (ist a gleich b?)
a != b (ist a ungleich b?)
a < b (ist a kleiner als b?)
a > b (ist a größer als b?)

[9] *Stamenov*, MMR-Aktuell 2020, 433995.

[10] Angebote zum Einstieg finden sich unter anderem bei: open.hpi.de/courses/javaeinstieg2020; freecodecamp.org; codecademy.com; codeavengers.com; teamtreehouse.com; sololearn.com/home, Abruf v. 13.7.2021.

[11] Vgl. online-learning.harvard.edu/course/cs50-lawyers?delta=0, Abruf v. 13.7.2021.

[12] Vgl. www.docassemble.org, Abruf v. 13.7.2021.

[13] www.python.org/downloads, Abruf v. 13.7.2021.

a <= b (ist a kleiner oder gleich b?)
a >= b (ist a größer oder gleich b?)

Wenn die Frage jeweils mit Ja beantwortet werden kann, wird die Bedingung als wahr dargestellt, falls nicht, als unwahr.

III. Anwendung

Um das Programm anzuwenden, muss es gestartet werden. Der Computer geht von Beginn des Dokuments an Zeile für Zeile durch und führt aus, was in der Zeile steht. Sollte sich das Programm nicht ausführen lassen, zeigt sich wieder einmal, dass der Computer es mit Regeln sehr streng nimmt. Wenn eine Art und Weise der Darstellung nicht richtig gewählt wurde oder ein notwendiges Zeichen fehlt, dann zeigt der Interpreter (der Teil des Computers, der aus dem menschenlesbaren Quelltext maschinenlesbaren Code erzeugt) einen Error, weil er nicht versteht, was er in der betroffenen Zeile tun soll.

Im Folgenden findest du den Programmcode eines ersten kleinen Programms:

```
# Ausgabe (Text in Anführungszeichen)
print("Herzlich Willkommen!")
# Eingabe Name (Zuweisung in Variablennamen)
name = input("Gib hier deinen Namen ein:")
#Eingabe des Alters (Zuweisung in Variable alter)
alter = input("Wie alt bist du?")
# Wenn Alter größer/gleich 18
if int(alter) >=18:
  # Ausgabe
   print("Hallo",name,". Du bist", alter, "Jahre alt.",
   "Du bist volljährig.")
  # Ansonsten
else:
  print("Hallo",name,". Du bist",alter,"Jahre alt.",
  "Du bist nicht volljährig.")
```

D. Fazit

Wir hoffen, mit diesem kurzen Beitrag Neugier auf das Programmieren geweckt und aufgezeigt zu haben, wie leicht sich die ersten Schritte gehen lassen. Motivator könnte zugleich die Perspektive auf das sein, was sich langfristig mithilfe von Technologien umsetzen lässt. Ein wesentlicher Teil der juristischen Arbeit besteht neben der juristischen Recherche in der Ausfertigung von Schriftsätzen und Verträgen mit immer wiederkehrenden Klauseln. Damit das wiederholte – insbesondere zeitintensive – Einpflegen dieser Standardformulierungen in Zukunft einen kleineren Raum im Terminplan in Anspruch nimmt, bietet es sich an, diese zu automatisieren. Schließlich wird der Wettbewerb selbst bewirken, dass dort, wo automatisiert werden kann, auch automatisiert wird.

Hierfür müssen Juristinnen es schaffen, die Hemmschwelle zu überwinden, sich einem vermeintlich völlig neuen Themenfeld zuzuwenden. Vor allem aber müssen Anwälte mit zunehmendem Einsatz von Assistenzsystemen wissen, wo die Schwächen der Technik sind, um Fehler zu erkennen und Risikovorsorge zu betreiben. Dazu muss man aber wenigstens in Grundzügen verstehen, welchen Regeln diese Programme folgen, um etwaige Fehler zu beheben oder gänzlich vorzubeugen. Wenn Techniken der automatisierten Rechtsfindung auf allen Ebenen zunehmen und bisherige Streitbeilegungsprozesse ergänzen, wandelt sich in naher Zukunft auch die rechtsmethodische Arbeitsweise. Damit sollten sich Anwältinnen, Studierende, Rechtsreferendare und die Rechtswissenschaft auseinandersetzen.[14] Unumgänglich bleibt jedenfalls die Offenheit gegenüber den zu erwartenden Veränderungen wie Projektmanagement in der Mandatsarbeit und Prozessautomatisierungen – damit es bald bei viel mehr Juristinnen heißt: print(“Hello World!“).

[14] *Anzinger*, Juristenausbildung in Zeiten der Digitalisierung: Müssen Anwälte der Zukunft programmieren können?, legal-tech.de, www.legal-tech.de/juristenausbildung-in-zeiten-der-digitalisierung-muessen-anwaelte-der-zukunft-programmieren-koennen, Abruf v. 13.7.2021.