Verwaltungsgericht Cottbus
Vom-Stein-Straße 27
03050 Cottbus
Telefax: 0355 4991-6499
Doppelte Ausführung
AZ: VG 8 K 518/21

Von:
Marcel Langner
[redacted]

Sehr geehrte Vorsitzende Richterin, Sehr geehrte Kammer,
ich erhielt am 03.09.2021 Ihr Schreiben vom 31.08.2021 mit Anhang der Hochschule und Bitte um eventuelle Stellungnahme. Vielen Dank für diese Möglichkeit. Ich kann nur hoffen, dass Ihnen nicht auch bald der Geduldsfaden reißt.

Ich entnehme dem Schreiben der Hochschule, dass diese weiterhin der Meinung ist ermessensfehlerfrei Auskunft erteilt zu haben. Auf meine Argumente, warum welche Unterlagen fehlen müssen, geht sie nicht ein. Ich möchte daher vermuten, dass meine Ausführungen (zumindest größtenteils) zutreffend sind.

Ich halte an allen meinen Anträgen aus meinen Schreiben vom 03.08.2021 fest.

Die Hochschule schlägt nun vor, dass ich vor Ort Einsicht nehmen könne. Bisher hat die Hochschule nicht dargelegt, dass die von mir vorgeschlagene Form der Einsicht (26.07.2021) nicht durchführbar ist und welche Gründe nach AIG dem entgegenstehen. Tatsächlich gibt sie doch auch selbst an, dass der Akteneinsicht keine weiteren Gründe entgegenstehen (28.07.2021). Wo ist also das Problem? Eine Einsicht vor Ort führt auch auf Seiten der Hochschule zu mehr Aufwand. Es stellt sich also die Frage nach der eigentlichen Motivation, für welche sie bereit ist diesen Mehraufwand in Kauf zu nehmen.
Ich bestehe daher weiterhin auf der von mir vorgeschlagenen Art der Akteneinsicht und werde Ihnen, wertes Gericht in diesem Schreiben darlegen, warum dies so auch erforderlich ist und auch offensichtlich von anderen, wie der bpb (26.06.2021), von selbst aus so durchgeführt wird.
Jede andere Form sehe ich, für einem vergleichbaren Fall, als ermessensfehlerhaft an.

Quellcode unterscheidet sich von anderen Akteninhalten dergestalt, als dass dieser wesentlich kompliziertere weitere Verständnisebenen aufweist. Man kann hier auch nicht so argumentieren, dass eben immer ein bestimmtes Fachwissen erforderlich ist, um den Inhalt von Fachakten verstehen zu können. Selbst wenn ein routinierter Programmierer eine bestimmte Programmiersprache beherrscht, benötigt er Zeit, um ein darin geschriebenes Programm verstehen zu können. Es ist nicht möglich, durch eine einmalige Betrachtung die Funktionsweise genau zu verstehen. Ansonsten gäbe es ja auch keine Programmierfehler. Diese kommen nicht daher, dass der Programmierende besonders „schlecht“ in seinem Fach ist. Nein, sie rühren daher, dass die heute geschriebenen Programme einen Umfang/Komplexität aufweisen, die an die Grenzen unserer Gehirnleistung heranreicht. Nicht immer, aber doch immer häufiger.

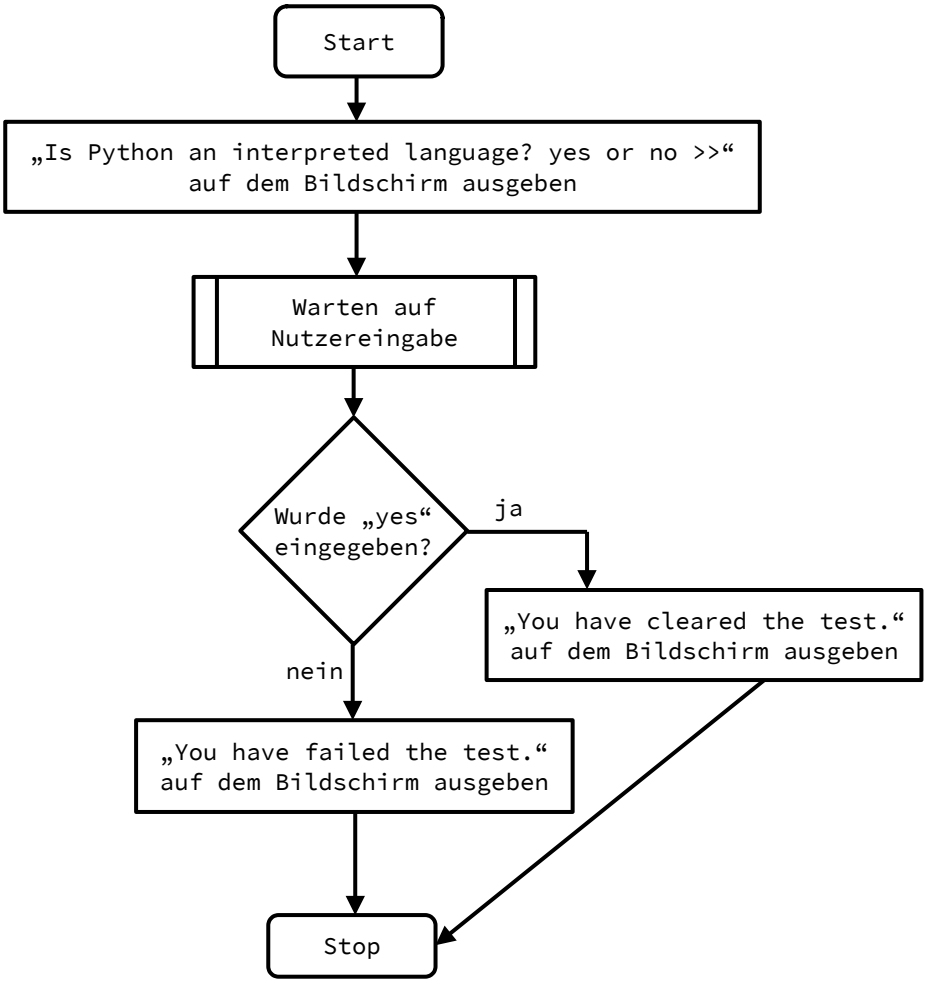
Beispielhaft möchte ich Ihnen Quellcode in der Programmiersprachen Python zeigen:

```
answer = input("Is Python an interpreted language? yes or no >> ")

if answer == "yes" :
    print("You have cleared the test.")
else :
    print("You have failed the test.")

print("Thanks!")
```

Dieser Quellcode ist durch jeden zu lesen, der der Sprache Englisch mächtig ist. Aber er ist dadurch noch nicht zu verstehen ohne Fachwissen und etwas Zeit zum überlegen. Bei der hier von mir verwendeten Programmiersprachen spielen die Einrückungen übrigens auch eine Rolle für den Ablauf des Programms und dienen nicht nur der Leserlichkeit.
Es hat sich bewährt, dass komplizierte Programmteile visualisiert werden. Dafür haben sich Standards herausgebildet, welche Formen und Farben zu verwenden sind, damit solche Visualisierungen weltweit zu verstehen sind. Ein solcher Standard ist zum Beispiel UML.
Visualisierungen werden genau dann durchgeführt, wenn der Quellcode einen Umfang oder Struktur aufweist, so dass der Programmablauf nicht mehr so ohne weiteres im Kopf des Programmierenden abgebildet werden kann. Sie dienen dann häufig auch der Dokumentation. Das obige Beispiel kann in seinem Ablauf wie folgt visualisiert werden, wobei ich hier die für Neueinsteiger eingängigere Symbolik der DIN 66001/ISO 5807 entlehne und die Sprache deutsch verwende:



Das hier gezeigte Beispiel ist ein sehr einfaches. Aber auch dieses besitzt bereits eine zweite Ebene des Verständnisses. Hier kann der Geübte nämlich erkennen, dass bei Eingabe jedes anderen Wortes als „yes“ auch die Fehlermeldung angezeigt wird, man hätte den Test nicht bestanden. Der etwas Erfahrene kann auch erkennen, dass das Wort „yes“ genau so in seiner Kleinschreibung eingetippt werden muss, um den Test zu bestehen.
Sie finden in Anlage 1 ein nur leicht umfangreicheres Beispiel. Die hier gezeigten Beispiele sind sehr einfach (10. Klasse Informatik).
Wann immer Quellcode durch einen Fachkundigen gelesen wird, wird im Kopf dieser Person der Computer simuliert und der Ablauf visualisiert. Was da im Kopf dann genau für Bilder entstehen ist bei jedem anders (Konstruktivismus). Letztlich versucht die Person aber zu ermitteln, wie das Programm sich bei Ausführung wohl verhalten wird. Bei großen Programmen, wie hier, macht/kann man das nicht, sondern führt es einfach aus und schaut was passiert. Es werden in der Folge dann nur Programmteile genauer untersucht, die eventuell Quellen für Fehler sein könnten.

Der hier in Frage stehende Quellcode besteht aus mindestens 120 einzelnen Dateien, wobei jede wesentlich mehr Zeilen Code aufweist, als die Beispiele. Hinzu kommt die Ebene der Modellierung im Rahmen der Objektorientierung, die nicht immer alles leichter macht. Die unterschiedlichen Dateien (Programmteile) greifen dabei aufeinander zu und stehen in Beziehungen zueinander. Der Quellcode ist daher nicht von oben nach unten lesend verstehbar bzw. läuft so ab, sondern „springt“ im Ablauf zwischen den Dateien hin und her. Ein wenig mit Gesetzesparagraphenverweisen innerhalb unterschiedlicher Gesetze vergleichbar. Hier ist, wie auch in der Programmierung, das bestreben nicht alles nochmal hinzuschreiben, sondern bereits geregeltes einfach zu benutzen. Hier auch mit all den Folgen, wenn man an der einen Ecke eigentlich nur eine Kleinigkeit ändern wollte.

Und natürlich weiß das die Hochschule. Jeder der programmiert weiß das und würde die Auskunft, so wie bisher erteilt als unzweckmäßig erachten, sofern es darum geht auch wirklich zu verstehen, wie das Programm im Detail abläuft. Also die Akte auch einzusehen und nicht nur anzusehen. Und darum ging es meiner Lesart nach auch dem Gesetzesgeber (aus der Gesetzesbegründung des AIG):
„Auf der anderen Seite steht das Land Brandenburg mit diesem Gesetzgebungsvorhaben in guter Nachbarschaft zu anderen europäischen und außereuropäischen Ländern, die das Recht des Bürgers auf Hinsicht in staatliche Akten zur verbesserten Teilhabe an den politischen Mitgestaltungsrechten ausgestaltet haben.“
Nicht darum, dass man Akten anschauen kann (dass ist nur das Mittel zum Zweck), sondern darum, dass der Bürger die Möglichkeit hat Behördenhandeln (hier auch im Sinne von Programmhandeln auszulegen) transparent nachzuvollziehen, was Verstehen voraussetzt, und aus dieser Transparenz heraus Mitgestaltungsmöglichkeit zu erlangen.

Es scheint mir also nun doch die grundsätzliche Frage zu beantworten, ob das Ermessen einer Behörde nach §7 (1) AIG als erfüllt gilt, wenn eine Akteneinsicht so erfolgt, dass eine fachkundige Person nicht in der Lage ist, den Inhalt zu verstehen, es jedoch nur deswegen nicht kann, weil die Umstände der Akteneinsicht so gewählt worden sind, wie sie gewählt wurden. Hier in Tateinheit mit Erschwerung des Prozesses zur Last des Einsichtnehmenden (nur vor Ort) ohne sachliche Begründung und unter Mehraufwand auch auf eigener Seite, obwohl doch Alternativen vorliegen, zu denen sich die Behörde jedoch konkret noch nie eingelassen hat. Aufgrund jeglichen Fehlens von Rückmeldungen der Hochschule dahingehend, wird es auch Ihnen, wertes Gericht, schwer fallen, Kompromissvorschläge zu machen, auf die sich beide Seiten einlassen könnten.

[redacted]

Anlage 1: Leicht umfangreicheres Beispiel:
Ich möchte mit diesem Beispiel aufzeigen, dass sich aus dem Lesen eines Codes nicht ohne zusätzlichen Zeitaufwand dessen Sinn ergibt oder dessen Komplexität.

Quellcode:

```
x = 10
y = 20
z = 30

print("Start")
if x == 10:
    print(" Nested If")
    if y == 20:
        print(" End of Nested If Block ")
    else:
        print(" End of Nested If-Else Block ")
elif y == 20:
    print(" Elif block ")
else:
    print(" Nested If")
    if z == 30:
        print(" End of Nested If Block ")
    else:
        print(" End of Nested If-Else Block ")
print("Stop")
```

Visualisierung: