Datenumwandlung und Programmierung – eine experimentelle Einführung

Prof. Dr. Carsten Link

27. April 2022

1 Lernziele

In dieser Übung sollen Sie mittels einfacher kleiner Schritte einen Einblick in die Programmierung und das Umwandeln von Datendarstellungen gewinnen. Dabei sollen Sie aktiv mit kleinen Programmen experimentieren. Auch wenn Ihnen keine Einführung in die Programmierung gegeben wurde, werden Sie durch ausprobieren und mit Hilfe ihrer Kommilitonen und des Dozenten ein kleines Programm erstellen können.

Mit der Bearbeitung der Aufgaben erlangen Sie die folgenden Kompetenzen:

- Sie verstehen, wie einzelne Zeichen, Wörter und ganze Texte in Form von Zahlen in einem Programm gespeichert und verarbeitet werden können
- Ihnen ist bewusst, dass zwischen den verschiedenen Datenformaten hin- und hergewandelt werden muss
- Sie können einfache Berechnungen auf Zahlen, welche in Form von Zeichenketten vorliegen, programmieren
- Sie können komplexere Berechnungen auf Zahlen, welche in Form von Zeichenketten vorliegen, nachvollziehen

2 Informationsdarstellung

Computer können im Wesentlichen nur Zahlen verarbeiten (Nullen und Einsen). Um anders dargestellte Daten verarbeiten zu können, müssen diese Daten in Form von Zahlen dargestellt werden. Zur Darstellung von Buchstaben, Ziffern, Satz- und Sonderzeichen werden Tabellen verwendet.

Aus einzelnen Zeichen können Zeichenketten zusammengestellt werden, um Wörter, Sätze und ganze Texte in Form von Zahlen im Computer zu speichern und zu verarbeiten:

- Zeichen: `a`, `b`, `c`, ... `z`, `A`, ... Z, `0`, ... `9`, ...
- Zeichenketten: `Franz jagt im komplett verwahrlosten Taxi quer durch Bayern`, `12`, `zwölf`

2.1 ASCII-Tabelle

Die traditionelle Tabelle, die druckbare Zeichen mit Zahlen verknüpft, ist die ASCII-Tabelle. Überfliegen Sie Wikipedia: ASCII[1], um ein grobes Verständnis der Zuordnung von druckbaren Zeichen (Buchstaben, Ziffern, Satz- und Sonderzeichen) und Zahlenwerten zu bekommen. Besonders interessant ist die Zuordnung der druckbaren Ziffer `0` und dem dazugehörigen Zahlenwert `48`.

[1] https://de.wikipedia.org/wiki/American_Standard_Code_for_Information_Interchange

2.2 Zahlenwerte und Zeichenketten

In diesem Praktikum soll an der Oberfläche des Themas Programmierung gekratzt werden. Die konkrete Programmierpsprache ist hier nebensächlich – es geht lediglich darum, einen Eindruck zu vermitteln, wie dem Computer Anweisungen gegeben werden können, die dieser Ausführen kann.

Python ist eine einfache und mächtige Programmiersprache. Wir verwenden zur programmierung in Python das Programm Thonny – Python IDE for beginners[2]. Sie können alternativ auch auf dieser Web-Seite Python ausprobieren: repl.it: Python3[3] (Hinweis: schalten Sie über das Zahnrad links oben `autocomplete` aus). Falls Sie mit `repl.it` Probleme haben, versuchen Sie runpython.org[4], rextester.com[5] oder try it online[6].

2.3 Zeichen und Zeichenketten

Bringen Sie auf einer dieser Web-Seiten den folgenden Python-Code zur Ausführung:

```
# Listing 1

a="4711"
b="b ist eine kurze Zeichenkette mit Buchstaben"

print("a:")
print(a)
print(a[0])
print(a[1])
print(a[2])
print(a[3])

print("b:")
print(b)
print(b[0])
print(b[2])
print(b[3])
```

1. Frage: Was geschieht, wenn Sie `print()` mit der Variablen `a` oder `b` verwenden (aufrufen, also `print(a)`)?
2. Frage: Was geschieht, wenn Sie `print()` mit der Variablen `a[0]` oder `a[1]` verwenden (aufrufen, also `print(a[0])`)?
3. Frage: Warum ergeben `print(4711+4000)` und `print("4711+4000")` und `print("4711"+"4000")` unterschiedliche Ausgaben?

2.4 Ziffern und deren Indizes in der ASCII-Tabelle

Hinweis: beachten Sie die Einrückung mit Leerzeichen (z.B. vor `return`), da Python die Einrückung berücksichtigt, um die Zugehörigkeit von Zeilen festzulegen. Erweitern Sie nun Ihr Programm um diese Zeilen:

```
# Listing 2

def asciiIndexOfChar(character):
  return bytearray(character,"ASCII")[0]

print("asciiIndexOfChar :")
print(asciiIndexOfChar(a[0]))
print(asciiIndexOfChar(a[1]))
```

[2] https://thonny.org/
[3] https://repl.it/languages/python3
[4] https://runpython.org
[5] http://rextester.com/l/python3_online_compiler
[6] https://tio.run

```
print(asciiIndexOfChar(a[2]))
print(asciiIndexOfChar(a[3]))
```

1. Frage: Was wird ausgegeben (im Vergleich zur vorherigen Aufgabe)?

2.5 Ziffern und deren Zahlenwerte

Erweitern Sie nun Ihr Programm um diese Zeilen:

```
# Listing 3

def numberValueOfChar(character):
  offsetAsciiZero = 38
  return asciiIndexOfChar(character) - offsetAsciiZero

print("numberValueOfChar :")
print(numberValueOfChar(a[0]))
print(numberValueOfChar(a[1]))
print(numberValueOfChar(a[2]))
print(numberValueOfChar(a[3]))
```

1. Frage: Welchen Zweck könnte `numberValueOfChar` haben (Hinweis: ASCII-Tabelle)?
2. Frage: Welchen Fehler enthält `numberValueOfChar`?
3. Frage: Wie können Sie `numberValueOfChar` in `ZahlenWertVon` umbenennen?

2.6 Schleifen

Statt wie in der vorherigen Aufgabe die Ausgabe mehrfach hinzuschreiben, bieten Programmiersprachen Schleifen an:

```
# Listing 4

def printNumberValues(s):
  index = 0
  length = len(s)
  print("Zeichenkette ", s ," ist ", length, " Zeichen lang")
  while index < length :
    print("index = ", index, "   numberValueOfChar() = ", numberValueOfChar(s[index]))
    index = index + 1

printNumberValues(a)
```

Dies führt dazu, dass die Zeilen unterhalb von `while` mehrfach ausgeführt werden, jedoch der Wert von **index** jeweils ein anderer ist. Effektiv werden diese Zeilen ausgeführt (nicht abtippen – vergleichen Sie die Ausgabe unten mit der `while`-Schleife oben):

```
print("index = ", 0, "   numberValueOfChar() = ", numberValueOfChar(s[0]))
print("index = ", 1, "   numberValueOfChar() = ", numberValueOfChar(s[1]))
print("index = ", 2, "   numberValueOfChar() = ", numberValueOfChar(s[2]))
print("index = ", 3, "   numberValueOfChar() = ", numberValueOfChar(s[3]))
```

2.7 Putting it all together

Die folgende Funktion fasst alles oben Gelernte zusammen: sie automatisiert die einzelnen Zugriffe per Index auf Zeichen der Zeichenkette und summiert die Anzahl der Ziffern in der Zeichenkette auf:

```
# Listing 5

def countNumericChars(z):
  index = 0
  length = len(z)
  numberOfNumerics = 0
  while index < length:
    if (asciiIndexOfChar(z[index]) >= 48) and (asciiIndexOfChar(z[index]) <= 57) :
      numberOfNumerics = numberOfNumerics + 1
    index= index + 1
  return numberOfNumerics

f="1234 abcd"
print("f enthlt ", countNumericChars(f), " arabische Ziffern")
```

Aufgabe: Geben Sie den obigen Quelltext am Ende Ihres bisherigen Programmes ein und lassen ihn ausführen.

1. Frage: Was wird für `f` ausgegeben?
2. Frage: Wie arbeitet `countNumerics()` im Einzelnen?

3 Prüfsummen

Laut en.wikipedia: Checksum[7] ist eine Prüfsumme: A checksum is a small-size datum from a block of digital data for the purpose of detecting errors which may have been introduced during its transmission or storage.

Neben der Fehlererkennung werden Prüfsummen in vielen anderen Bereichen eingesetzt.

3.1 Berechnung von Quersummen

Die einfachste Prüfsumme ist die Quersumme. Wird zusätzlich zu einer zu übertragenden Zahl, deren Quersumme übertragen, so kann beim Empfang der Nachricht erkannt werden, ob bei der Übertragung eine Ziffer ihren Wert geändert (z. B. +1) hat.

1. Aufgabe: überlegen Sie, wie Veränderungungen mittels einer Quersumme konkret erkannt werden können
2. Aufgabe: Nehmen Sie `countNumericChars` als Vorlage und Versuchen Sie (ggf. mit Hilfe des Dozenten), eine Funktion `digitSum(z)` zu konstruieren, welche die Quersumme (engl. *digit sum*) einer als ASCII-Zeichenkette gegebenen Zahl `z` berechnet.

   Rufen Sie Ihre Funktion `digitSum` mit diesem Code auf:

   ```
   # Listing 6

   z="1231"
   print("Zahl in Form einer ASCII-Zeichenkette: ", z)
   print("Quersumme davon: ", digitSum(z))
   ```

   Falls Sie `digitSum()` korrekt programmiert haben, gibt das Programm folgendes aus:

   ```
   Zahl in Form einer ASCII-Zeichenkette: 1231
   Quersumme davon: 7
   ```

[7] https://en.wikipedia.org/wiki/Checksum

3.2 Berechnung von ISBN-Prüfsummen

Schauen Sie sich die Berechnung der ISBN-10-Prüfsumme (de.wikipedia: ISBN-10[8]) an.

1. Aufgabe: Welche Fehler kann die ISBN-Prüfsumme erkennen, die mit einer Quersumme nicht erkannt werden?

2. Aufgabe: Programmieren Sie die Berechnung der ISBN-10-Prüfsumme in ihrem Programm für `a="344642638"`. Nehmen Sie `digitSum` als Vorlage. Hinweis: den Divisionsrest können Sie mit dem Modulus-Operator `%` berechnen lassen. Das Ergebnis ist 8 (allerdings nicht die 8, die in **a** am Ende steht).

[8] https://de.wikipedia.org/wiki/Internationale_Standardbuchnummer#ISBN-10