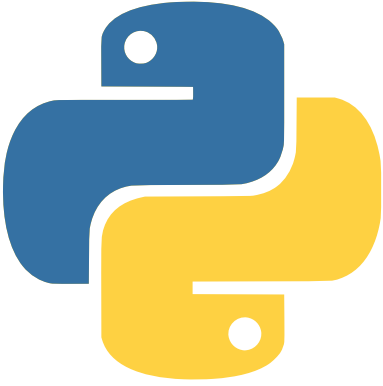
Universität Stuttgart
Projekt digit@L – BOOST. SKILLS. SUPPORT.



Programmierkurs Python

Lesen und Schreiben von Dateien

Dominik Göddeke

IANS SimTech

Stiftung Innovation in der Hochschullehre

Lesen und Schreiben von Dateien

Motivation

- Bisher: „Daten“ für Codeschnipsel **von Hand**

```
my_dict = {1:1,2:4,3:9,4:15}
print(my_dict)
```

Motivation

- Bisher: „Daten“ für Codeschnipsel **von Hand**

```
my_dict = {1:1,2:4,3:9,4:15}
print(my_dict)
```

- **Hochgradig impraktikabel** in der Praxis

Motivation

- Bisher: „Daten“ für Codeschnipsel **von Hand**

```
my_dict = {1:1,2:4,3:9,4:15}
print(my_dict)
```

- **Hochgradig impraktikabel** in der Praxis
- 4598573453 Datensätze? Sprichwörtliche 10 TB Machine Learning Daten?

Motivation

- Bisher: „Daten“ für Codeschnipsel **von Hand**

```
my_dict = {1:1,2:4,3:9,4:15}
print(my_dict)
```

- **Hochgradig impraktikabel** in der Praxis
- 4598573453 Datensätze? Sprichwörtliche 10 TB Machine Learning Daten?
- Ausweg: **Lesen und Schreiben von Dateien**

Beispiele

- Das **Lehrbuch-Standardbeispiel**

Beispiele

- Das **Lehrbuch-Standardbeispiel**
  - Sensor liefert Messdaten, Speicherung als Rohdaten

Beispiele

- Das **Lehrbuch-Standardbeispiel**
  - Sensor liefert Messdaten, Speicherung als Rohdaten
  - Aufgabe: Nachbehandlung und Verarbeitung, da Rohdaten nicht aussagekräftig

Beispiele

- Das **Lehrbuch-Standardbeispiel**
  - Sensor liefert Messdaten, Speicherung als Rohdaten
  - Aufgabe: Nachbehandlung und Verarbeitung, da Rohdaten nicht aussagekräftig
- Das **Studiums-Beispiel**

Beispiele

- Das **Lehrbuch-Standardbeispiel**
  - Sensor liefert Messdaten, Speicherung als Rohdaten
  - Aufgabe: Nachbehandlung und Verarbeitung, da Rohdaten nicht aussagekräftig
- Das **Studiums-Beispiel**
  - Abschlussarbeit mit empirischer Komponente

Beispiele

- Das **Lehrbuch-Standardbeispiel**
  - Sensor liefert Messdaten, Speicherung als Rohdaten
  - Aufgabe: Nachbehandlung und Verarbeitung, da Rohdaten nicht aussagekräftig
- Das **Studiums-Beispiel**
  - Abschlussarbeit mit empirischer Komponente
  - Ergebnis einer Studie: Excel-Tabelle mit Probant*innen-Daten

Beispiele

- Das **Lehrbuch-Standardbeispiel**
  - Sensor liefert Messdaten, Speicherung als Rohdaten
  - Aufgabe: Nachbehandlung und Verarbeitung, da Rohdaten nicht aussagekräftig
- Das **Studiums-Beispiel**
  - Abschlussarbeit mit empirischer Komponente
  - Ergebnis einer Studie: Excel-Tabelle mit Probant*innen-Daten
  - Aufgabe: statistische Aufbereitung und Visualisierung

Beispiele

- Das **Lehrbuch-Standardbeispiel**
  - Sensor liefert Messdaten, Speicherung als Rohdaten
  - Aufgabe: Nachbehandlung und Verarbeitung, da Rohdaten nicht aussagekräftig
- Das **Studiums-Beispiel**
  - Abschlussarbeit mit empirischer Komponente
  - Ergebnis einer Studie: Excel-Tabelle mit Probant*innen-Daten
  - Aufgabe: statistische Aufbereitung und Visualisierung
- Das **Pragmatismus-Beispiel**

Beispiele

- Das **Lehrbuch-Standardbeispiel**
  - Sensor liefert Messdaten, Speicherung als Rohdaten
  - Aufgabe: Nachbehandlung und Verarbeitung, da Rohdaten nicht aussagekräftig
- Das **Studiums-Beispiel**
  - Abschlussarbeit mit empirischer Komponente
  - Ergebnis einer Studie: Excel-Tabelle mit Probant*innen-Daten
  - Aufgabe: statistische Aufbereitung und Visualisierung
- Das **Pragmatismus-Beispiel**
  - Eingabe- und Ausgabedaten persistent archivieren

Beispiele

- Das **Lehrbuch-Standardbeispiel**
  - Sensor liefert Messdaten, Speicherung als Rohdaten
  - Aufgabe: Nachbehandlung und Verarbeitung, da Rohdaten nicht aussagekräftig
- Das **Studiums-Beispiel**
  - Abschlussarbeit mit empirischer Komponente
  - Ergebnis einer Studie: Excel-Tabelle mit Probant*innen-Daten
  - Aufgabe: statistische Aufbereitung und Visualisierung
- Das **Pragmatismus-Beispiel**
  - Eingabe- und Ausgabedaten persistent archivieren
  - Sinnfrei: Kopieren aus IDE, Speichern über Datei-Explorer

Beispiele

- Das **Lehrbuch-Standardbeispiel**
  - Sensor liefert Messdaten, Speicherung als Rohdaten
  - Aufgabe: Nachbehandlung und Verarbeitung, da Rohdaten nicht aussagekräftig
- Das **Studiums-Beispiel**
  - Abschlussarbeit mit empirischer Komponente
  - Ergebnis einer Studie: Excel-Tabelle mit Probant*innen-Daten
  - Aufgabe: statistische Aufbereitung und Visualisierung
- Das **Pragmatismus-Beispiel**
  - Eingabe- und Ausgabedaten persistent archivieren
  - Sinnfrei: Kopieren aus IDE, Speichern über Datei-Explorer
- Das **Prinzip-Beispiel**

Beispiele

- Das **Lehrbuch-Standardbeispiel**
  - Sensor liefert Messdaten, Speicherung als Rohdaten
  - Aufgabe: Nachbehandlung und Verarbeitung, da Rohdaten nicht aussagekräftig
- Das **Studiums-Beispiel**
  - Abschlussarbeit mit empirischer Komponente
  - Ergebnis einer Studie: Excel-Tabelle mit Probant*innen-Daten
  - Aufgabe: statistische Aufbereitung und Visualisierung
- Das **Pragmatismus-Beispiel**
  - Eingabe- und Ausgabedaten persistent archivieren
  - Sinnfrei: Kopieren aus IDE, Speichern über Datei-Explorer
- Das **Prinzip-Beispiel**
  - Reproduzierbarkeit von Experimenten als Kernelement wissenschaftlicher Arbeit

Beispiele

- Das **Lehrbuch-Standardbeispiel**
  - Sensor liefert Messdaten, Speicherung als Rohdaten
  - Aufgabe: Nachbehandlung und Verarbeitung, da Rohdaten nicht aussagekräftig
- Das **Studiums-Beispiel**
  - Abschlussarbeit mit empirischer Komponente
  - Ergebnis einer Studie: Excel-Tabelle mit Probant*innen-Daten
  - Aufgabe: statistische Aufbereitung und Visualisierung
- Das **Pragmatismus-Beispiel**
  - Eingabe- und Ausgabedaten persistent archivieren
  - Sinnfrei: Kopieren aus IDE, Speichern über Datei-Explorer
- Das **Prinzip-Beispiel**
  - Reproduzierbarkeit von Experimenten als Kernelement wissenschaftlicher Arbeit
  - Auch schon im Studium

Beispiele

- Das **Lehrbuch-Standardbeispiel**
  - Sensor liefert Messdaten, Speicherung als Rohdaten
  - Aufgabe: Nachbehandlung und Verarbeitung, da Rohdaten nicht aussagekräftig
- Das **Studiums-Beispiel**
  - Abschlussarbeit mit empirischer Komponente
  - Ergebnis einer Studie: Excel-Tabelle mit Probant*innen-Daten
  - Aufgabe: statistische Aufbereitung und Visualisierung
- Das **Pragmatismus-Beispiel**
  - Eingabe- und Ausgabedaten persistent archivieren
  - Sinnfrei: Kopieren aus IDE, Speichern über Datei-Explorer
- Das **Prinzip-Beispiel**
  - Reproduzierbarkeit von Experimenten als Kernelement wissenschaftlicher Arbeit
  - Auch schon im Studium
  - Bereitstellung für Überprüfung und Validierung durch andere

Dateiformate

Dateiformate

- Vielzahl von **Dateiformaten**

Dateiformate

- Vielzahl von **Dateiformaten**
  - Textdateien, und auch Tabellen als ASCII-Daten

Dateiformate

- Vielzahl von **Dateiformaten**
  - Textdateien, und auch Tabellen als ASCII-Daten
  - Bilddateien, Musikdateien, Videodateien

Dateiformate

- Vielzahl von **Dateiformaten**
  - Textdateien, und auch Tabellen als ASCII-Daten
  - Bilddateien, Musikdateien, Videodateien
  - zip/rar/tar/gz Archive

Dateiformate

- Vielzahl von **Dateiformaten**
  - Textdateien, und auch Tabellen als ASCII-Daten
  - Bilddateien, Musikdateien, Videodateien
  - zip/rar/tar/gz Archive
  - Ausführbare Programme (executables)

Dateiformate

- Vielzahl von **Dateiformaten**
  - Textdateien, und auch Tabellen als ASCII-Daten
  - Bilddateien, Musikdateien, Videodateien
  - zip/rar/tar/gz Archive
  - Ausführbare Programme (executables)
  - Dateiformate aus Anwendungsprogrammen (Office-Anwendungen, . . .)

Dateiformate

- Vielzahl von **Dateiformaten**
  - Textdateien, und auch Tabellen als ASCII-Daten
  - Bilddateien, Musikdateien, Videodateien
  - zip/rar/tar/gz Archive
  - Ausführbare Programme (executables)
  - Dateiformate aus Anwendungsprogrammen (Office-Anwendungen, . . .)
- **In diesem Kurs: nur Textdateien**

Dateiformate

- Vielzahl von **Dateiformaten**
  - Textdateien, und auch Tabellen als ASCII-Daten
  - Bilddateien, Musikdateien, Videodateien
  - zip/rar/tar/gz Archive
  - Ausführbare Programme (executables)
  - Dateiformate aus Anwendungsprogrammen (Office-Anwendungen, . . .)
- **In diesem Kurs: nur Textdateien**
- Binäre Dateien oftmals **viel** effizienter, aber weniger allgemein

Öffnen von Dateien in Python

Öffnen von Dateien in Python

- Grundlegender Befehl: `open(file, mode)`

Öffnen von Dateien in Python

- Grundlegender Befehl: `open(file, mode)`
- Argument `file`: Dateiname als Zeichenkette

Öffnen von Dateien in Python

- Grundlegender Befehl: `open(file, mode)`
- Argument `file`: Dateiname als Zeichenkette
- Regeln

Öffnen von Dateien in Python

- Grundlegender Befehl: `open(file, mode)`
- Argument `file`: Dateiname als Zeichenkette
- Regeln
  - Datei liegt im gleichen Verzeichnis wie das Python-Programm (das Jupyter Notebook) das die Datei öffnet

Öffnen von Dateien in Python

- Grundlegender Befehl: `open(file, mode)`
- Argument `file`: Dateiname als Zeichenkette
- Regeln
  - Datei liegt im gleichen Verzeichnis wie das Python-Programm (das Jupyter Notebook) das die Datei öffnet
  - Dateiname enthält Verzeichnisangabe (relativ zum ausführenden Programm)

Öffnen von Dateien in Python

- Grundlegender Befehl: `open(file, mode)`
- Argument `file`: Dateiname als Zeichenkette
- Regeln
  - Datei liegt im gleichen Verzeichnis wie das Python-Programm (das Jupyter Notebook) das die Datei öffnet
  - Dateiname enthält Verzeichnisangabe (relativ zum ausführenden Programm)
  - Linux/Mac: `file = 'my_directory/text.txt'`

Öffnen von Dateien in Python

- Grundlegender Befehl: `open(file, mode)`
- Argument `file`: Dateiname als Zeichenkette
- Regeln
  - Datei liegt im gleichen Verzeichnis wie das Python-Programm (das Jupyter Notebook) das die Datei öffnet
  - Dateiname enthält Verzeichnisangabe (relativ zum ausführenden Programm)
  - Linux/Mac: `file = 'my_directory/text.txt'`
  - Windows `file = 'my_directory\text.txt'`

Öffnen von Dateien in Python

- Grundlegender Befehl: `open(file, mode)`
- Argument `file`: Dateiname als Zeichenkette
- Regeln
  - Datei liegt im gleichen Verzeichnis wie das Python-Programm (das Jupyter Notebook) das die Datei öffnet
  - Dateiname enthält Verzeichnisangabe (relativ zum ausführenden Programm)
  - Linux/Mac: `file = 'my_directory/text.txt'`
  - Windows `file = 'my_directory\text.txt'`
  - Dateiname enthält absolute Verzeichnisangabe

Öffnen von Dateien in Python

- Grundlegender Befehl: `open(file, mode)`
- Argument `file`: Dateiname als Zeichenkette
- Regeln
  - Datei liegt im gleichen Verzeichnis wie das Python-Programm (das Jupyter Notebook) das die Datei öffnet
  - Dateiname enthält Verzeichnisangabe (relativ zum ausführenden Programm)
  - Linux/Mac: `file = 'my_directory/text.txt'`
  - Windows `file = 'my_directory\text.txt'`
  - Dateiname enthält absolute Verzeichnisangabe
  - Windows: `file = 'C:\Eigene Dateien\Pythonkurs\text.txt'`

Öffnen von Dateien in Python

- Grundlegender Befehl: `open(file, mode)`
- Argument `file`: Dateiname als Zeichenkette
- Regeln
  - Datei liegt im gleichen Verzeichnis wie das Python-Programm (das Jupyter Notebook) das die Datei öffnet
  - Dateiname enthält Verzeichnisangabe (relativ zum ausführenden Programm)
  - Linux/Mac: `file = 'my_directory/text.txt'`
  - Windows `file = 'my_directory\text.txt'`
  - Dateiname enthält absolute Verzeichnisangabe
  - Windows: `file = 'C:\Eigene Dateien\Pythonkurs\text.txt'`
  - Windows/Mac: `file = '~/Pythonkurs/myfile.txt'`

Öffnen von Dateien in Python

- Grundlegender Befehl: `open(file, mode)`
- Argument `file`: Dateiname als Zeichenkette
- Regeln
  - Datei liegt im gleichen Verzeichnis wie das Python-Programm (das Jupyter Notebook) das die Datei öffnet
  - Dateiname enthält Verzeichnisangabe (relativ zum ausführenden Programm)
  - Linux/Mac: `file = 'my_directory/text.txt'`
  - Windows `file = 'my_directory\text.txt'`
  - Dateiname enthält absolute Verzeichnisangabe
  - Windows: `file = 'C:\Eigene Dateien\Pythonkurs\text.txt'`
  - Windows/Mac: `file = '~/Pythonkurs/myfile.txt'`
- Im Notebook: Hinweise zur Plattformunabhängigkeit

Öffnen von Dateien in Python

- Grundlegender Befehl: `open(file, mode)`

Öffnen von Dateien in Python

- Grundlegender Befehl: `open(file, mode)`
- Argument `mode`: Art des Dateizugriffs

Öffnen von Dateien in Python

- Grundlegender Befehl: `open(file, mode)`
- Argument `mode`: Art des Dateizugriffs
  - `mode='r'`: reiner Lesezugriff

Öffnen von Dateien in Python

- Grundlegender Befehl: `open(file, mode)`
- Argument `mode`: Art des Dateizugriffs
  - `mode='r'`: reiner Lesezugriff
  - `mode='w'`: Überschreiben des Dateiinhalts

Öffnen von Dateien in Python

- Grundlegender Befehl: `open(file, mode)`
- Argument `mode`: Art des Dateizugriffs
  - `mode='r'`: reiner Lesezugriff
  - `mode='w'`: Überschreiben des Dateiinhalts
  - `mode='a'`: Weiterschreiben ab Dateiende

Öffnen von Dateien in Python

- Grundlegender Befehl: `open(file, mode)`
- Argument `mode`: Art des Dateizugriffs
  - `mode='r'`: reiner Lesezugriff
  - `mode='w'`: Überschreiben des Dateiinhalts
  - `mode='a'`: Weiterschreiben ab Dateiende
  - `mode='r+'` oder `mode='w+'`: Lesen und Schreiben

Beispiel

- Anlegen der Datei `test.txt` mit einem Texteditor

```
Testdatei fuer Dateioperationen
bla
blub
1234
5.6789
[22,33,44,55]
{name:"Dominik", nachname:"Goeddeke", alter:43, rolle="Dozent"}
```

Beispiel

- Anlegen der Datei `test.txt` mit einem Texteditor

```
Testdatei fuer Dateioperationen
bla
blub
1234
5.6789
[22,33,44,55]
{name:"Dominik", nachname:"Goeddeke", alter:43, rolle="Dozent"}
```

- Abspeichern im Ordner der Python-Datei, die zugreifen soll

Codebeispiel

Zusammenfassung: Öffnen von Dateien

Zusammenfassung: Öffnen von Dateien

- `f = open(file, mode)`: wie bereits diskutiert

Zusammenfassung: Öffnen von Dateien

- `f = open(file, mode)`: wie bereits diskutiert
- `read()`: Einlesen des gesamten Inhalts der Datei als Zeichenkette

Zusammenfassung: Öffnen von Dateien

- `f = open(file, mode)`: wie bereits diskutiert
- `read()`: Einlesen des gesamten Inhalts der Datei als Zeichenkette
- `f.close()`: Freigabe der Datei, d.h. Rückgabe der Kontrolle an Betriebssystem

Zusammenfassung: Öffnen von Dateien

- `f = open(file, mode)`: wie bereits diskutiert
- `read()`: Einlesen des gesamten Inhalts der Datei als Zeichenkette
- `f.close()`: Freigabe der Datei, d.h. Rückgabe der Kontrolle an Betriebssystem
- **Daumenregel**: jedes Vorkommen von `f = open(...)` muss mit `f.close()` gepaart werden

Zusammenfassung: Öffnen von Dateien

- `f = open(file, mode)`: wie bereits diskutiert
- `read()`: Einlesen des gesamten Inhalts der Datei als Zeichenkette
- `f.close()`: Freigabe der Datei, d.h. Rückgabe der Kontrolle an Betriebssystem
- **Daumenregel**: jedes Vorkommen von `f = open(...)` muss mit `f.close()` gepaart werden
- Potentielle Komplikation bei Sonderzeichen wie ö in Göddeke: siehe Notebook

Lesen von Dateiinhalten

Lesen von Dateiinhalten

- `read()` ohne Argument: Einlesen des kompletten Inhalts der Datei „auf einen Schwung“

Lesen von Dateiinhalten

- `read()` ohne Argument: Einlesen des kompletten Inhalts der Datei „auf einen Schwung“
- Problematisch bei sehr großen Dateien

Lesen von Dateiinhalten

- `read()` ohne Argument: Einlesen des kompletten Inhalts der Datei „auf einen Schwung“
- Problematisch bei sehr großen Dateien
- Oft wenig sinnvoll, besser Verarbeitung der Inhalte einer Datei direkt beim Einlesen

Lesen von Dateiinhalten

- `read()` ohne Argument: Einlesen des kompletten Inhalts der Datei „auf einen Schwung“
- Problematisch bei sehr großen Dateien
- Oft wenig sinnvoll, besser Verarbeitung der Inhalte einer Datei direkt beim Einlesen
- Erlaubt „Überspringen“ nicht benötigter Inhalte

Lesen von Dateiinhalten

- `read()` ohne Argument: Einlesen des kompletten Inhalts der Datei „auf einen Schwung“
- Problematisch bei sehr großen Dateien
- Oft wenig sinnvoll, besser Verarbeitung der Inhalte einer Datei direkt beim Einlesen
- Erlaubt „Überspringen“ nicht benötigter Inhalte
- Performance: Zugriff auf Datei typischerweise 100–1000 Mal langsamer als auf Daten im Speicher

Lesen von Dateiinhalten

- Für Textdateien: Zeile als „natürliche Granularität“

Lesen von Dateiinhalten

- Für Textdateien: Zeile als „natürliche Granularität“
- Wenig verblüffend: `file` Datentyp iterierbar per `for` Schleife

Lesen von Dateiinhalten

- Für Textdateien: Zeile als „natürliche Granularität“
- Wenig verblüffend: `file` Datentyp iterierbar per `for` Schleife
- Alternative: `f.readline()`, Einlesen einer Zeile

Lesen von Dateiinhalten

- Für Textdateien: Zeile als „natürliche Granularität“
- Wenig verblüffend: `file` Datentyp iterierbar per `for` Schleife
- Alternative: `f.readline()`, Einlesen einer Zeile
- `f.readlines()`, Einlesen in Liste der Zeilen

Lesen von Dateiinhalten

- Für Textdateien: Zeile als „natürliche Granularität“
- Wenig verblüffend: `file` Datentyp iterierbar per `for` Schleife
- Alternative: `f.readline()`, Einlesen einer Zeile
- `f.readlines()`, Einlesen in Liste der Zeilen
- `read(n)` Einlesen von `n` Zeichen

Codebeispiele

Schreiben von Dateien

Schreiben von Dateien

- `f.write(s)` schreibt Zeichenkette `s`

Schreiben von Dateien

- `f.write(s)` schreibt Zeichenkette `s`
- In eine bereits geöffnete Datei

Schreiben von Dateien

- `f.write(s)` schreibt Zeichenkette `s`
- In eine bereits geöffnete Datei
- Wichtig: Modus (siehe oben) muss Schreiben erlauben, also `w`, `w+` oder `a`

Codebeispiele

Impressum, Danksagung und Quellen

Stiftung Innovation in der Hochschullehre

Gefördert durch die Stiftung Innovation in der Hochschullehre im Rahmen des Projekts digit@L, https://stiftung-hochschullehre.de

SimTech

Gefördert mit Mitteln der Deutschen Forschungsgemeinschaft (EXC 2075 - 390740016) im Rahmen der Exzellenzstrategie

---

Autor: Dominik Göddeke, IANS, Universität Stuttgart

ZOERR

Veröffentlicht auf dem Zentralen OER Repositorium Baden-Württemberg, https://www.zoerr.de