Einführung in Jupyter

Jupyter Notebooks sind eine Mischung aus Programmcode und Text, die zum Erkunden einladen.

Aufbau

Jedes Notebook besteht aus einer Sequenz von Zellen. Diese enthalten entweder Text, der mit Hilfe der Auszeichnungssprache *Markdown* formatiert wird, oder Code. Beide Arten von Zellen können dabei beliebig gemischt werden, um beispielsweise Erklärungen einzubringen, die mit einfachen Kommentaren im Quellcode nicht anschaulich darstellbar wären.

Der Quellcode kann dabei in verschiedenen Programmiersprachen verfasst sein. Offiziell unterstützt Jupyter dabei über 40 Sprachen durch sogenannte *Kernel*. Im Folgenden werden wir uns jedoch auf Python - eine einfach zu erlernende und gut lesbare Sprache, die sich einer großen Gemeinschaft von Programmierern erfreut - beschränken.

Jede Zelle hat dabei die Möglichkeit, eine Ausgabe (*Output*) zu erzeugen. Bei Textzellen ist dies der formatierte Text, bei Codezellen das Ergebnis der letzten Anweisung, Textausgaben, Tabellen oder Grafiken. Jede Zelle lässt sich unabhängig von anderen Zellen im Notebook ausführen. Da Programmierer ungern ihre Maus verwenden, stehen die zwei Tastaturkürzel `STRG + ENTER`, um eine Zelle auszuführen, und `SHIFT + ENTER`, um zusätzlich zur nächsten Zelle zu springen, bereit.

Installation und Start

Möglicherweise verstehen Sie **noch** nicht alle Anweisungen, die im Folgenden ausgeführt werden. Es ist jedoch essentiell, dass Sie die bereitgestellten Jupyter Notebooks öffnen, bearbeiten und ausführen können.

JupyterHub

JupyterHub ist eine Anwendung, die es ermöglicht, Jupyter Notebooks mehreren Benutzern auf einem gemeinsamen Server bereitzustellen. Dieser Server übernimmt dabei auch die Ausführung aller Code-Zellen. Zur Verwendung benötigen Sie lediglich einen Webbrowser, sodass mit dieser Methode beispielsweise auch der Zugriff über iPadOS möglich ist.

Die Verknüpfung zum JupyterHub findet über Moodle statt. Klicken Sie auf einen dort hinterlegten Link, wird automatisch eine Benutzerumgebung für Sie vorbereitet und die benötigten Notebooks heruntergeladen bzw. aktualisiert.

JupyterHub ist der bevorzugte Weg für die Arbeit mit den Jupyter Notebooks.

Docker

Alternativ bietet Docker die Möglichkeit, reproduzierbare Umgebungen an andere Computer zu senden und dort zu nutzen. Für diese Veranstaltung existiert ein sogenanntes *Image*, das alle benötigten Abhängigkeiten enthält. Führen Sie zum Starten den folgenden Befehl aus:

```
docker run -it --rm -v "$(pwd)":/home/jovyan/work -p 8888:8888 troebs/jupyter-dsmt
```

Unter Windows ist abermals eine Anpassung notwendig:

```
docker run -it --rm -v "%cd%":/home/jovyan/work -p 8888:8888 troebs/jupyter-dsmt
```

In der Ausgabe erscheint kurz darauf ein Link der folgenden Form. Kopieren Sie diesen in die Adresszeile Ihres Browsers, um auf Jupyter Lab zuzugreifen.

```
http://127.0.0.1:8888/lab?token=<48 zufällige Zeichen>
```

Kommandozeile

Am fehleranfälligsten ist die Verwendung von Jupyter mit einer virtuellen Umgebung. Mit installiertem Python führen Sie dazu in einer Eingabeaufforderung im korrekten Ordner die folgenden Befehle aus:

```
python -m venv venv
source venv/bin/activate
pip install tui_dsmt
jupyter-lab
```

Unter Windows müssen Sie einen der Pfade anpassen:

```
python -m venv venv
venv\Scripts\activate
pip install tui_dsmt
jupyter-lab
```

Jupyter Lab sollte anschließend automatisch geöffnet werden. Ist dies nicht der Fall, kopieren Sie bitte den der folgenden Form entsprechenden Link aus der Ausgabe in die Adresszeile Ihres Browsers:

```
http://127.0.0.1:8888/lab?token=<48 zufällige Zeichen>
```

Programmzustand

Weiter oben wurde kurz erwähnt, dass Zellen unabhängig voneinander ausgeführt werden können. Dies ist korrekt, wenn man sich auf die Reihenfolge und Anzahl bezieht, um sich damit vom üblichen Ablauf sequentiell ablaufender Programme zu distanzieren. Der Zustand des Programms - also unter anderem die Menge der gespeicherten Variablen - wird jedoch akkumuliert und unter allen Zellen geteilt. Auf anderem Wege wäre es auch nicht möglich, aus einer Zelle auf Daten zuzugreifen, die in einer anderen Zelle geladen oder verändert wurden.

Ein Vorteil wird Ihnen im Verlauf der Programmierübungen vermutlich schnell klar werden: Das Programm muss nicht für jede noch so kleine Änderung von vorn ausgeführt werden. Ändern Sie also einen Teil, der zwei Sekunden benötigt, aber auf Daten aus einem Teil zurückgreift, der zehn Sekunden benötigt, warten Sie in Jupyter zwei Sekunden auf das Ergebnis ihrer Änderung, sofern die länger laufende Zelle bereits mindestens einmal ausgeführt wurde. In einem sequentiellen Programm vergehen dagegen zwölf Sekunden und Ihnen zeitnah die Lust, kleine Änderungen mit großen Wartezeiten mehrmals auszuprobieren.

Ein Nachteil hingegen wird Ihnen vermutlich ebenfalls begegnen. Sie selbst entscheiden, welche Zellen in welcher Reihenfolge, welcher Häufigkeit und mit welchen nachträglich eingefügten Änderungen ausgeführt werden. Im Nachhinein ist vor allem bei längeren Verbünden von Zellen nicht immer ganz eindeutig, welche Anweisungen wann ausgeführt wurden. Daraus resultierende Fehler sind schwer zu verstehen, weshalb Jupyter selbst den Versuch einer Auflösung mitliefert. Der Schalter ▶▶ in der oberen Leiste startet den Kernel neu (löscht also alle Variablen) und führt anschließend alle Zellen in der Reihenfolge im Notebook exakt einmal aus.

Eine englischsprachige Präsentation von Joel Grus, der darüber hinaus auch ein Buch zur Data Science veröffentlichte, stellt die Probleme aus seiner Sicht humoristisch dar. Mit etwas Erfahrung im Umgang mit Notebooks lohnt sich eventuell ein Blick.