Institut für Informatik
Dr. Tobias Neckel
M.Sc. Ivana Jovanovich
M.Sc. Friedrich Menhorn

Technische Universität München
WS 2020/2021

Einführung in die wissenschaftliche Programmierung

Übungsblatt 0: Installationsparty

In dieser Vorlesung bekommen Sie eine Einführung in die wissenschaftliche Programmierung, wofür wir die Programmiersprache Python nutzen werden. Daher ist das Ziel dieses Arbeitsblattes, dass Sie an dessen Ende mit einer funktionierenden Installation von Python bereit sind für die folgenden Wochen. Zögern Sie nicht bei allen nun folgenden Schritten ihren Tutor bei Problemen um Hilfe zu bitten.

1 Anaconda

Anaconda (`https://docs.conda.io/projects/conda/en/latest/user-guide/install/index.html`) ist eine vereinheitlichte Plattform für Python. Es funktioniert für Unix-System (Linux, iOS), sowie für Windows. Da es nicht nur Python, sondern auch die meisten für uns relevanten Pakete standardmäßig enthält, empfehlen wir die Installation. Nach erfolgreicher Installation können Sie ein Python Programm starten indem Sie

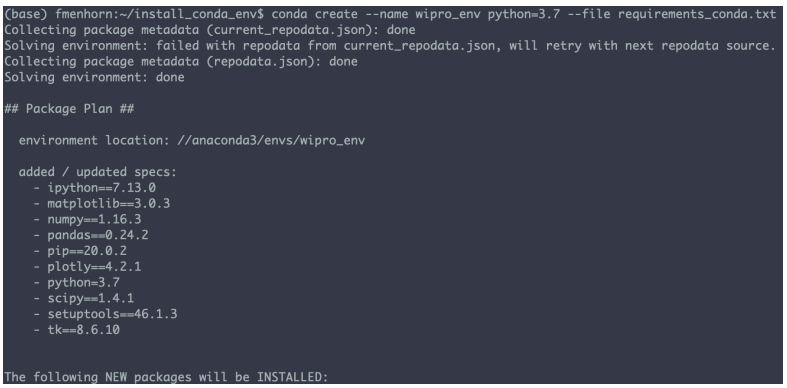
- in Windows die Anaconda Konsole öffnen (z.B. über das Startmenü)
- in Unix die Standardkonsole öffnen (oft `Terminal` genannt, Strg+Alt+T in Linux)

Beide sollten standardmäßig Anaconda mit dessen Standardumgebung `(base)` starten. Mehr dazu im nächsten Abschnitt.

1.1 Anaconda Umgebung (Environments)

In Anaconda gibt es sogenannte "Umgebungen" (eng. Environments). In einer Umgebung können Sie z.B. neue Pythonpakete (d.h. Bibliotheken) installieren, die dann nur in dieser Umgebung zur Verfügung stehen. Die Standardumgebung heißt `(base)`:

Mit dem Befehl `conda create --name <umgebungsname>` können Sie eine neue Umgebung erstellen. In unserem Fall sind wir noch etwas spezifischer: Laden Sie den Ordner `install_conda_env` von Moodle herunter. Darin finden Sie eine Datei namens `requirements_conda.txt`. Mit Hilfe solch einer Datei definieren wir Pakete, die wir in unserer Umgebung haben möchten. Um eine neue Umgebung aufzusetzen, zusammen mit dieser Datei, verwenden wir den Befehl `conda create --name <umgebungsname> python=3.7 --file requirements_conda.txt`. Zusätzlich setzen wir die Versionsnummer von Python auf 3.7. Achten Sie darauf, dass Sie in ihrem Terminal im Ordner sein müssen, indem die Datei `requirements_conda.txt` liegt.



Nach der Ausführung des Befehls sehen Sie auch, wie Sie die Umgebung aktivieren, nämlich `conda activate <umgebungsname>`:

Um zu `(base)` zurückzukehren, verwenden Sie einfach `conda deactivate`.
Wir empfehlen die Verwendung einer eigenen Umgebung für diese Vorlesung, in diesem Beispiel `(wipro_env)`, aber natürlich können Sie auch direkt in `(base)` arbeiten. Weitere nützliche Befehle:

- `conda env list`: Zur Auflistung von allen Umgebungen
- `conda remove --name <umgebungsname> --all`: Zum Entfernen einer Umgebung

Viele weitere Informationen zur Umgebung finden Sie hier.

1.2 Python Pakete

In Python gibt es sogenannte Pakete, um Ihnen die Programmierarbeit zu erleichtern. Vergleichbar mit Bibliotheken in anderen Programmiersprachen, finden sich darin Funktionalitäten, die Sie verwenden können. Sie müssen das Rad nicht (immer) neuerfinden. Das klassische Beispiel für wissenschaftliches Rechnen ist das Paket `numpy` (`https://numpy.org`). In Anaconda können Sie ein Paket ganz einfach über den Befehl `conda install <paketname>` installieren. Das Paket `numpy` (und ein paar andere) haben wir schon durch die Datei `requirements_conda.txt` installiert.
Eine Auflistung aller installierten Pakete in der aktuellen Umgebung finden Sie mit `conda list`:

```
(wipro_env) fmenhorn:~/install_conda_env$ conda list
# packages in environment at //anaconda3/envs/wipro_env:
#
# Name                    Version                   Build  Channel
appnope                   0.1.0                    py37_0
backcall                  0.2.0                      py_0
blas                      1.0                         mkl
ca-certificates           2020.10.14                   0
certifi                   2020.6.20          pyhd3eb1b0_3
cycler                    0.10.0                   py37_0
decorator                 4.4.2                      py_0
freetype                  2.10.4               ha233b18_0
intel-openmp              2019.4                     233
ipython                   7.13.0             py37h5ca1d4c_0
ipython_genutils          0.2.0                    py37_0
jedi                      0.17.2                   py37_0
kiwisolver                1.3.0              py37h23ab428_0
libcxx                    10.0.0                       1
libedit                   3.1.20191231         h1de35cc_1
```

1.3 Python Skript ausführen

Um nun ein Pythonprogramm auszuführen, wechseln Sie einfach zu dem jeweiligen Ordner, wo sich die Datei befindet (Windows mit `dir`, Linux mit `cd`). Danach können Sie die Datei ausführen mit `python <dateiname>`. Für ein konkretes Beispiel gehen sie zu Abschnitt 2.1.

2 Testbeispiele

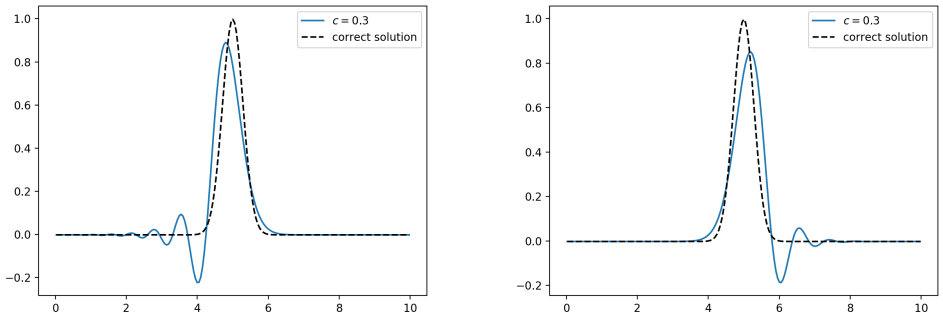
Sie finden auf Moodle vier Testbeispiele, die Sie am Ende dieser Übung auf ihrem Rechner laufen sollten. Wir gehen hier nicht darauf ein, was die Programme tatsächlich berechnen, sondern zeigen nur die gewünschten Resultate als Test für Sie zum Vergleich. Für das erste Beispiel zeigen wir auch, wie das Programm ausgeführt wird.

2.1 Advektion

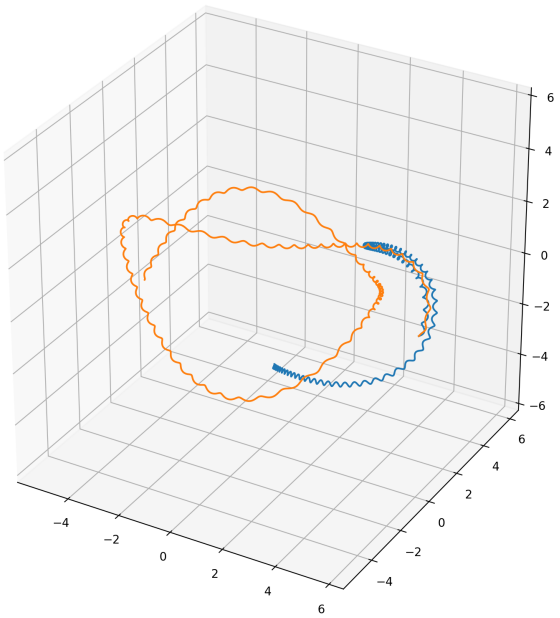
Start des Programs:

```
(wipro_env) fmenhorn:~/[redacted]/blatt00/advection$ python main.py
BeamWarming scheme used for space time discretization
0.0 0.015
0.195 0.015
0.39 0.015
0.585 0.015
0.78 0.015
0.9750000000000001 0.015
1.1700000000000002 0.015
1.3650000000000002 0.015
1.5600000000000003 0.015
1.7550000000000003 0.015
1.9500000000000004 0.015
```

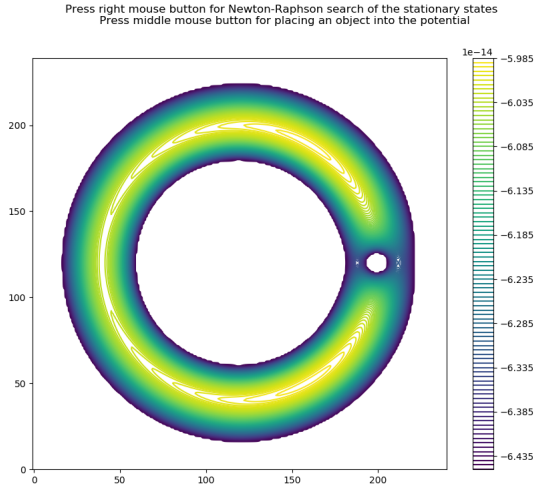
Resultat:



2.2 Drifts



2.3 LaGrange



2.4 PM