

## Python installieren

Wir installieren Python nicht von python.org, da wir damit allein nicht alle notwendigen Tools haben, um später bestimmte Libraries wie numpy auf allen Betriebssystemen zu installieren. Stattdessen verwenden wir eine Python Distribution namens Anaconda, die die nötigen Tools mitliefert. Anaconda gibt es in zwei Varianten: mit vorinstallierten Libraries und ohne. Da wir Libraries immer abhängig vom Projekt installieren, wählen wir Miniconda, die Variante ohne weitere Libraries.

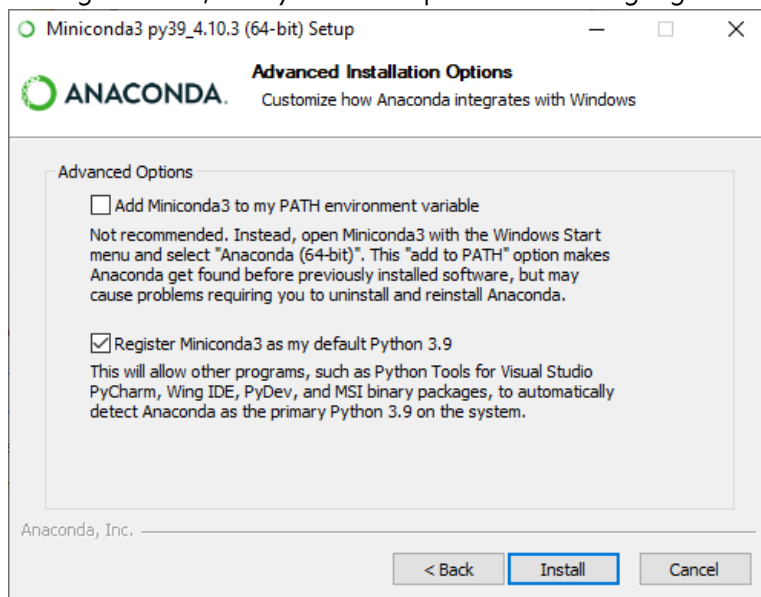
1. Downloade die für dich passende Version von Miniconda:

<https://docs.conda.io/en/latest/miniconda.html>

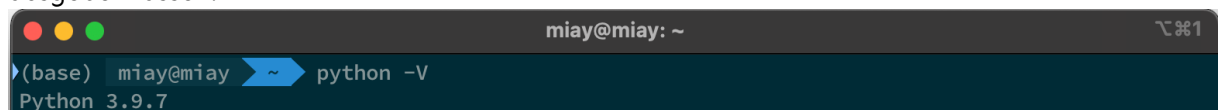
2. Folge der Installationsanleitung:

<https://conda.io/projects/conda/en/latest/user-guide/install/index.html>

Die default-Einstellungen unter Windows (siehe Screenshot) können beibehalten werden. Unter Windows wird Miniconda normalerweise nicht global installiert, sondern steht nur über eine besondere Kommandozeile namens Anaconda Prompt zur Verfügung. Programme wie Visual Studio Code können die Miniconda-Installation aber automatisch erkennen und stellen alle virtuellen Umgebungen, die mit Miniconda erzeugt werden, als Python Interpreter zur Verfügung.



3. Öffne entweder die Anaconda Prompt (Windows) oder dein Standard-Terminal. Wenn die Installation erfolgreich war, solltest du jetzt am Anfang der Zeile vor dem Prompt den Text (base) sehen. Die aktuell aktivierte Python-Version kannst du mit `python -V` ausgeben lassen.



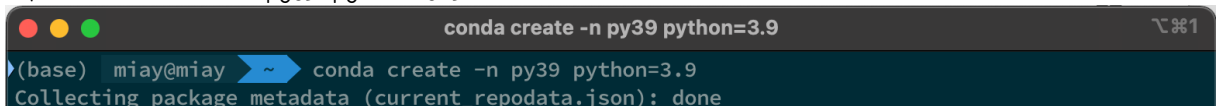
## Virtuelle Umgebungen

Anaconda stellt ein Packet- und Umgebungsmanager namens conda bereit. Damit können wir neue virtuelle Umgebungen für Python anlegen. Jede Umgebung hat ihre eigene Python-Version und eigene Libraries.

Um eine neue Umgebung für Data Science anzulegen:

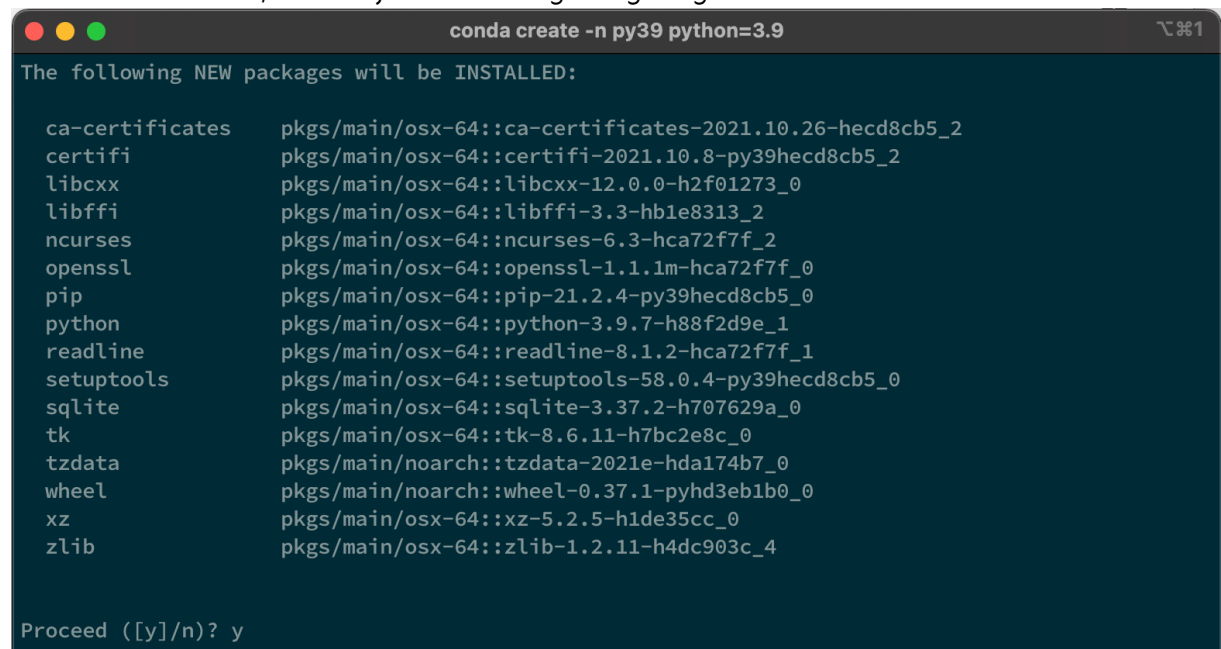
1. Öffne die Anaconda Prompt (Windows) oder ein Terminal, indem conda zur Verfügung steht ((base)-Text am Anfang).
2. Nutze den conda create Befehl zum Anlegen einer neuen Umgebung. Mit -n kannst du den Namen der Umgebung festlegen und mit python=3.x kannst du die Python-Version festlegen, die in der neuen Umgebung installiert werden soll.

```
$ conda create -n py39 python=3.9
```



```
conda create -n py39 python=3.9
(base) miay@miay ~$ conda create -n py39 python=3.9
Collecting package metadata (current_repodata.json): done
```

3. Bestätige die Installation mit y. conda create wird standardmäßig eine Reihe von Paketen installieren, mit der jede neue Umgebung ausgestattet wird.



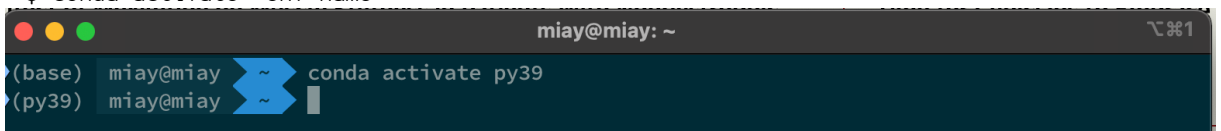
```
conda create -n py39 python=3.9
The following NEW packages will be INSTALLED:

ca-certificates      pkgs/main/osx-64::ca-certificates-2021.10.26-hecd8cb5_2
certifi              pkgs/main/osx-64::certifi-2021.10.8-py39hecd8cb5_2
libcxx               pkgs/main/osx-64::libcxx-12.0.0-h2f01273_0
libffi               pkgs/main/osx-64::libffi-3.3-hb1e8313_2
ncurses              pkgs/main/osx-64::ncurses-6.3-hca72f7f_2
openssl              pkgs/main/osx-64::openssl-1.1.1m-hca72f7f_0
pip                  pkgs/main/osx-64::pip-21.2.4-py39hecd8cb5_0
python               pkgs/main/osx-64::python-3.9.7-h88f2d9e_1
readline             pkgs/main/osx-64::readline-8.1.2-hca72f7f_1
setuptools           pkgs/main/osx-64::setuptools-58.0.4-py39hecd8cb5_0
sqlite               pkgs/main/osx-64::sqlite-3.37.2-h707629a_0
tk                   pkgs/main/osx-64::tk-8.6.11-h7bc2e8c_0
tzdata               pkgs/main/noarch::tzdata-2021e-hda174b7_0
wheel                 pkgs/main/noarch::wheel-0.37.1-pyhd3eb1b0_0
xz                   pkgs/main/osx-64::xz-5.2.5-h1de35cc_0
zlib                 pkgs/main/osx-64::zlib-1.2.11-h4dc903c_4

Proceed ([y]/n)? y
```

4. Aktiviere die neue Umgebung mit conda activate. Nach erfolgreicher Aktivierung ändert sich der Name der Umgebung im Terminal (im Beispiel zu (py39)).

```
$ conda activate <env-name>
```



```
miay@miay: ~$ conda activate py39
(py39) miay@miay ~$
```

5. Du kannst alle Pakete einer Umgebung aktualisieren, indem du conda update --all verwendest. Bestätige die Installation wie gewohnt mit y.

## Pakete installieren

Pakete bzw. Libraries können entweder mit Hilfe von conda oder pip installiert werden. Grundsätzlich empfehlen wir pip, da damit auch Pakete installiert werden können, die es nicht in Anaconda gibt. Alle Pakete werden immer in die aktuell ausgewählte Umgebung installiert.

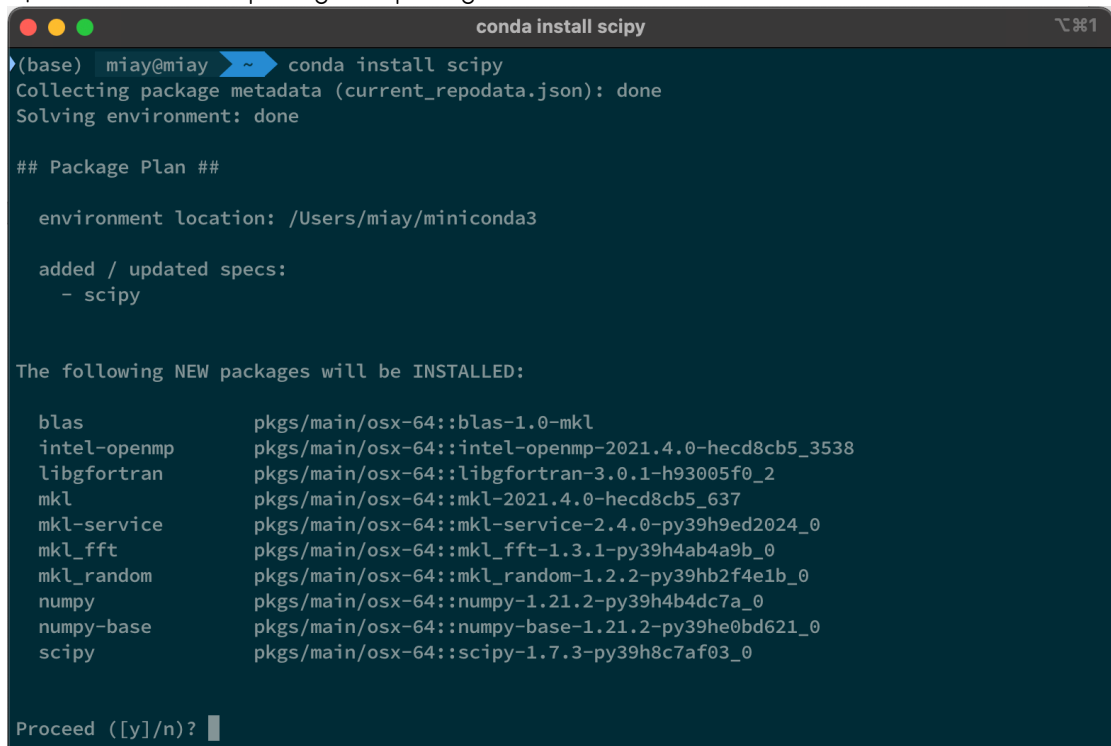
Vorbereitung:

1. Öffne die Anaconda Prompt (Windows) oder ein Terminal, indem conda zur Verfügung steht ((base)-Text am Anfang).
2. Optional: Aktiviere die gewünschte virtuelle Umgebung, in der du das Paket installieren möchtest:  
\$ conda activate <env-name>

### conda

1. Führe conda install auf der Konsole aus, um ein Paket zu installieren. conda erkennt alle dependencies (Abhängigkeiten) des zu installierenden Paketes und installiert diese ebenfalls. Die Installation muss mit y bestätigt werden. Du kannst mehrere Pakete mit Leerzeichen getrennt angeben.

\$ conda install <package1> <package2> ...



```

conda install scipy
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /Users/miay/miniconda3

  added / updated specs:
    - scipy

The following NEW packages will be INSTALLED:

blas                    pkgs/main/osx-64::blas-1.0-mkl
intel-openmp            pkgs/main/osx-64::intel-openmp-2021.4.0-hecd8cb5_3538
libgfortran             pkgs/main/osx-64::libgfortran-3.0.1-h93005f0_2
mkl                     pkgs/main/osx-64::mkl-2021.4.0-hecd8cb5_637
mkl-service             pkgs/main/osx-64::mkl-service-2.4.0-py39h9ed2024_0
mkl_fft                 pkgs/main/osx-64::mkl_fft-1.3.1-py39h4ab4a9b_0
mkl_random              pkgs/main/osx-64::mkl_random-1.2.2-py39hb2f4e1b_0
numpy                   pkgs/main/osx-64::numpy-1.21.2-py39h4b4dc7a_0
numpy-base              pkgs/main/osx-64::numpy-base-1.21.2-py39he0bd621_0
scipy                   pkgs/main/osx-64::scipy-1.7.3-py39h8c7af03_0

Proceed ([y]/n)? 

```

2. Alternativ können Pakete auch direkt bei der Erzeugung einer neuen Umgebung mit angegeben werden:  
\$ conda create -n <name> python=3.x <package1> <package2> ...
3. Führe conda update auf der Konsole aus, um ein Paket zu aktualisieren.  
\$ conda update <package1> <package2> ...

## pip

1. Führe `pip install` auf der Konsole aus, um ein Paket zu installieren. `pip` erkennt alle dependencies (Abhängigkeiten) des zu installierenden Paketes und installiert diese ebenfalls. Du kannst mehrere Pakete mit Leerzeichen getrennt angeben.  
`$ pip install <package1> <package2> ...`
2. Mit dem Zusatz `--upgrade` kannst du mit `pip install` auch Pakete aktualisieren:  
`$ pip install --upgrade <package1> <package2> ...`

## Jupyter Notebooks starten

Um Notebooks im Format `.ipynb` öffnen zu können, muss die Erweiterung `jupyter` installiert werden. Damit steht der Befehl `jupyter notebook` zur Verfügung. Eine Erweiterung ist `jupyterlab`, welches das reine Jupyter Notebook noch um weitere Features ergänzt, darunter einen Dateibrowser, mehrere Notebooks im Tab sowie ein integriertes Inhaltsverzeichnis. Mit dem Befehl `jupyter lab` kann Jupyterlab gestartet werden. Wichtig ist bei allen beiden Befehlen, dass der Server immer in dem Verzeichnis startet, in dem man sich zum Zeitpunkt des Befehlsaufrufs in der Konsole befand.

1. Öffne die Anaconda Prompt (Windows) oder ein Terminal, indem `conda` zur Verfügung steht ((`base`)-Text am Anfang).
2. Optional: Aktiviere die gewünschte virtuelle Umgebung, in der du Jupyter starten möchtest:  
`$ conda activate <env-name>`
3. Installiere `jupyter` und, wenn gewünscht, `jupyterlab`:  
`$ pip install jupyter jupyterlab`
4. Optional: Sofern du nicht in deiner (`base`)-Umgebung bist, musst du die virtuelle Umgebung noch als Kernel registrieren:  
`(venv) $ python -m ipykernel install --user --name <name> --display-name <display-name>`  
 Als Name `<name>` kannst du eine Kurzform verwenden, unter der der Kernel intern angesprochen werden kann. Der Anzeigename `<display-name>` ist der Name, unter dem der Kernel dann im Notebook ausgewählt werden kann.
5. Wechsle mit dem `cd`-Befehl in das Verzeichnis, in dem du ein Notebook starten bzw. öffnen möchtest.
6. Starte `jupyter notebook` oder `jupyter lab`:  
`$ jupyter notebook`  
`$ jupyter lab`  
 Daraufhin öffnet sich dein Standardbrowser und ruft die Seite `localhost:8888` auf.
7. Wähle deinen Kernel aus. Wenn du ein Notebook zum ersten Mal startest oder öffnest, musst du den Kernel auswählen, mit dem das Notebook ausgeführt werden soll. Dazu kannst du aus der Liste deinen gewünschten Kernel auswählen.

