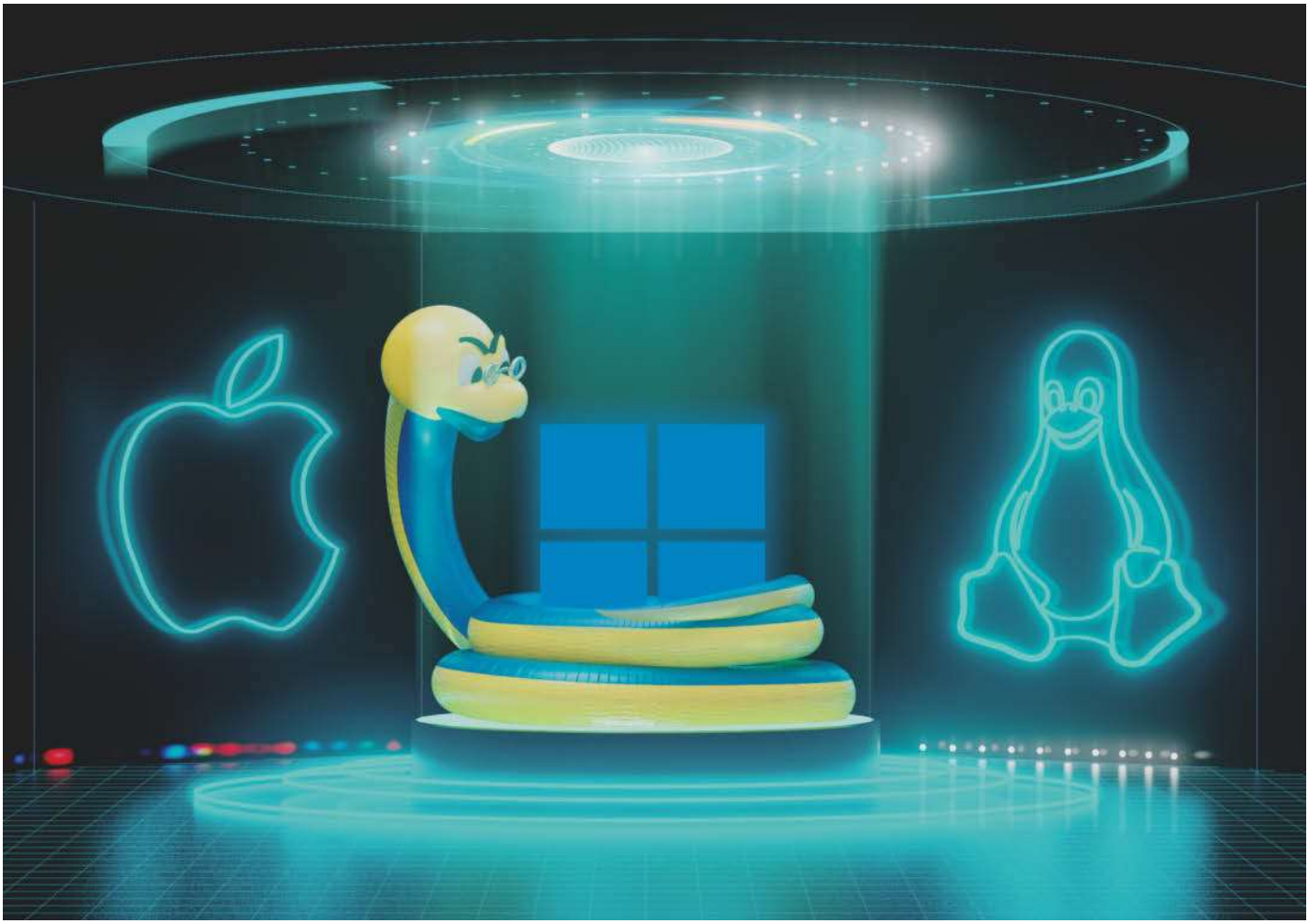


Bild: Andreas Martini

Draufgebeamt

So richten Sie Python schnell und einfach ein

Ihr Python-Einstieg ist nur wenige Minuten entfernt, ganz gleich, ob Sie Windows, Linux oder macOS nutzen. Nach der Installation können Sie aus Hunderttausenden Modulen wählen, die Ihre Skripte mit allerlei Funktionen ausstatten. Durch virtuelle Umgebungen verlieren Sie dabei nicht den Überblick.

Von Ronald Eikenberg und Jan Mahn

Um Python-Skripte zu entwickeln und auszuführen, benötigen Sie nur den Python-Interpreter, der auf fast allen Betriebssystemen läuft. Wir liefern Ihnen im folgenden Artikel Starthilfe für die Installation unter Windows, Linux und macOS. Zudem erklären wir, wie Sie sich aus dem großen Fundus an Python-Modulen bedienen und wie Sie für Ihre Projekte virtuelle Umgebungen mit venv einrichten. venv versorgt jedes Skript mit den passenden Modulen, damit Sie den Überblick nicht verlieren.

Wenn Sie bei null anfangen und selbst Skripte entwickeln möchten, greifen Sie am besten zur aktuellsten Python-Version (siehe ct.de/yknd). Zunächst sollten Sie jedoch herausfinden, ob Python bereits auf Ihrem System installiert ist – und wenn ja, in welcher Version. Hierzu können Sie die folgenden Befehle auf der Konsole ausprobieren:

```
python --version
python2 --version
python3 --version
```

Unter Windows listet der Python-Launcher py.exe gleich alle installierten Versionen auf, sofern vorhanden: `py -0`

Möglicherweise sind Sie jetzt bereits fündig geworden (Version 3.8 oder neuer) und können direkt zum Abschnitt „Versionen und Module ohne Chaos“ springen. Wahrscheinlicher ist jedoch, dass Sie keine oder eine alte Version vorfinden. In diesem Fall schaffen Sie gleich Abhilfe. Für den späteren Funktionstest können Sie mit einem Texteditor schon mal ein einfaches Python-Skript namens meinskript.py anlegen, das neben der Version einen kleinen Text ausgibt:

```
import sys
print(sys.version)
print("Hallo Python-Welt!")
```

Sie rufen es nach der Installation mit `python3 meinskript.py` auf – und wissen jetzt auch schon, wie man Skripte ausführt. Jetzt gehts erst einmal an die Installation von Python, die je nach Betriebssystem anders abläuft.

Python für Windows

Unter Windows ist Python zwar nicht vorinstalliert, das Betriebssystem hilft Ihnen jedoch zumindest auf die Sprünge: Solange Python nicht installiert ist, ist der Befehl `python3` ein Platzhalter, dessen Aufruf Sie geradewegs in den Microsoft Store befördert. Der Store preist anschließend eine Python-App zur Installation an, mit der Sie prinzipiell nichts falsch machen, da es sich beim Store um einen offiziellen Distributionskanal handelt. Allerdings schickt Sie Windows nicht zwangsläufig zur aktuellen Version, da die Verknüpfung statisch auf eine bestimmte Ausgabe zeigt. Suchen Sie im Store stattdessen lieber nach „python“ und wählen Sie die gewünschte Version selbst aus (momentan ist 3.10 aktuell).

Nach einem Klick auf „Herunterladen“ startet der rund 35 MByte große Download, anschließend läuft ohne weitere Rückfragen die Installation durch. Einfacher geht es nicht. Falls die aktuelle Python-Version noch nicht angeboten wird, oder Sie dem Microsoft Store nicht über den Weg trauen, können Sie sich den regulären Installer auf python.org besorgen. Der ist nur unwesentlich komplizierter zu bedienen und bietet zudem die ein oder andere Konfigurationsmöglichkeit.

Bei der manuellen Installation sollten Sie gleich nach dem Start des Setup-Programms ganz unten das Häkchen bei „Add Python 3.x to PATH“ setzen, damit das Installationsverzeichnis der Umgebungsvariable PATH hinzugefügt wird. So können Sie unter anderem den Befehl `python3` von jedem beliebigen Ort ausführen, ohne den vollständigen Pfad angeben zu müssen oder ins Installationsverzeichnis zu wechseln. Die übrigen Vorgaben passen in den meisten Fällen, Sie können die Installation also mit „Install Now“ anstoßen.

Befindet sich alles an Ort und Stelle, fragt Sie das Setupprogramm abschließend noch, ob es die Längenbeschränkung für Dateipfade in Windows aufheben soll („Disable file path length limit“). Das ist eine gute Idee, denn Windows erlaubt standardmäßig nur Dateipfade mit einer maximalen Länge von 260 Zeichen. Insbesondere, wenn Sie mit Python-Skripten aus dem Netz arbeiten, besteht die Gefahr, dass Sie durch verschachtelte Ordnerstrukturen an diese Begrenzung stoßen. Daher sollten Sie dieses Angebot annehmen. Negative Auswirkungen aufs System sind uns nicht bekannt. Wenn Sie die Längenbeschränkung nachträglich von Hand aufheben möchten, können Sie in der Registry den DWORD-Wert `Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\FileSystem\LongPathsEnabled` auf 1 setzen (siehe ct.de/yknd). Bei der Installation über den Store müssen Sie in jedem Fall selbst aktiv werden, wenn Sie das Limit deaktivieren möchten.

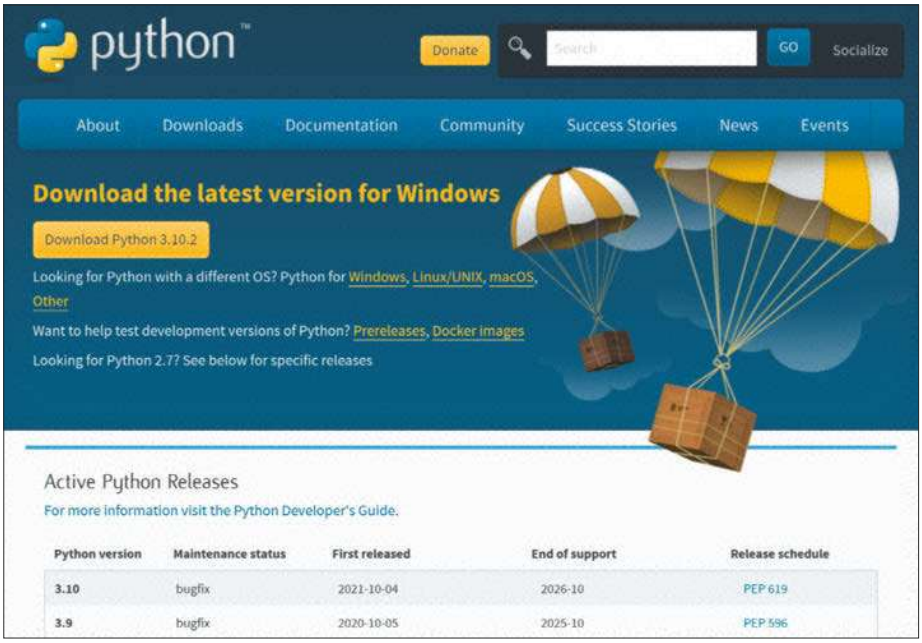
Nach Abschluss der Installation können Sie die Python-Shell über das Startmenü ausführen. Sie finden dort außerdem die rudimentäre Lernumgebung IDLE Shell, welche die ersten Schritte etwas erleichtern soll. Auch der Python-Paketmanager pip, auf den wir weiter unten noch eingehen, ist Teil der Installation. Windows-Nutzer bekommen zudem den Python-Launcher `py`, der den Umgang mit mehreren parallel installierten Python-Versionen erleichtert.

So startet der Befehl `py` stets die aktuellste installierte Python-Version, mit `py -3.10 meinskript.py` können Sie Ihr Skript gezielt mit einer bestimmten Version ausführen (in diesem Beispiel 3.10). Damit der Befehl `python3` und die anderen in der Eingabeaufforderung und im Windows-Terminal funktionieren, kann es nötig sein, sich von Windows ab- und wieder anzumelden, damit die geänderte Umgebungsvariable PATH neu geladen wird.

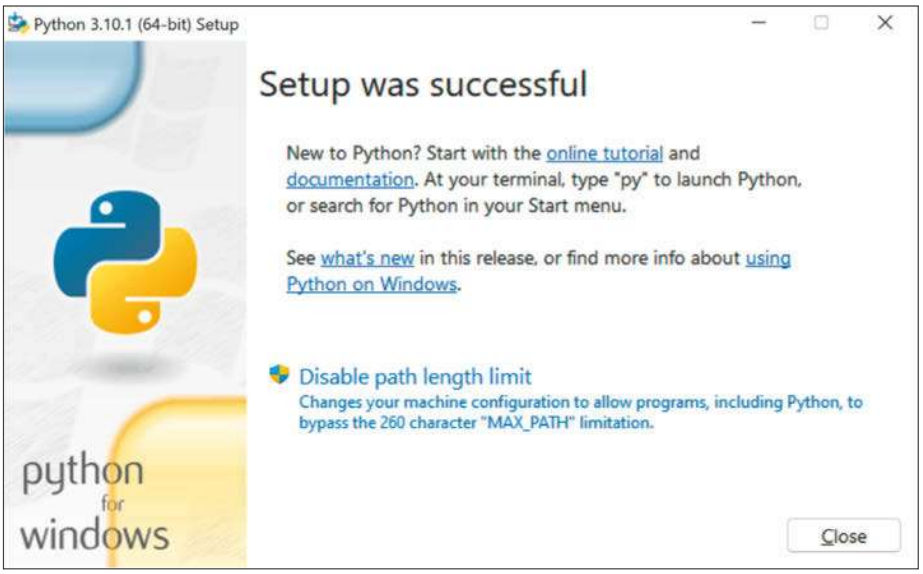
Leichtes Spiel haben Sie auch mit dem Windows-Paketmanager WinGet, den Sie vielleicht schon auf dem System haben. Tippen Sie einfach `winget install Python.Python.3` in Eingabeaufforderung oder Terminal ein, um die aktuelle Python-Version direkt von python.org herunterzuladen und anschließend installieren zu lassen. Sie müssen nur noch den Dialog der Benutzerkontensteuerung abnicken, damit der Installer die nötigen Rechte bekommt.

c't kompakt

- Die Python-Installation unter Windows, Linux und macOS geht leicht von der Hand.
- Um Chaos zu vermeiden, sollten Sie von Beginn an mit virtuellen Umgebungen (venv) arbeiten.
- Das Rundum-Sorglos-Paket Anaconda erleichtert den Umgang mit Modulen und venvs.



Die frischsten Pythons gibts auf python.org. Es gibt aber schnellere Installationswege.



Der Python-Installer bietet unter Windows an, die Längenbegrenzung für Dateipfade aufzuheben. Das erspart einen manuellen Eingriff in die Registry.

Der Python-Installer informiert Sie über den Fortschritt der laufenden Installation, einstellen können Sie jedoch nichts. Hängen Sie dem Befehl die Option `--interactive` (oder kurz `-i`) an, startet das grafische Setup-Programm mit allen Einstellungsmöglichkeiten (wie die Aufhebung der erlaubten Pfadlänge). Mit `--silent` (oder `-h`) hingegen läuft die Installation unsichtbar im Hintergrund ab. Über WinGet wird das Installationsverzeichnis standardmäßig dem PATH hinzugefügt.

Falls der WinGet-Befehl nicht funktioniert, müssen Sie zunächst das mit Windows vorinstallierte App-Paket „App-Installer“ über den Store auf den aktuellen Stand bringen. Das lohnt sich durchaus, denn mit dem Paketmanager können Sie nicht nur Python installieren und aktualisieren, sondern auch viele andere Programme. Eine ausführliche Einführung in WinGet haben wir bereits veröffentlicht [1].

Installation unter Linux

Bei vielen Linux-Distributionen ist Python bereits vorinstalliert. Wenn nicht (oder nicht in der gewünschten Version), werden Sie möglicherweise in den Repositories Ihrer Distribution fündig. Unter Ubuntu etwa bringen Sie die Paketlisten der Repos zunächst mit `sudo apt update` auf den aktuellen Stand. Mit `apt search python3.10` können Sie anschließend nach Installationspaketen zu einer bestimmten Python-Version suchen (in diesem Fall 3.10) und diese mit `apt install python3.10` installieren. Um Python-Module leicht nachrüsten zu können, sollten Sie auch den Paketmanager pip installieren: `sudo apt install python3-pip`.

Es dauert mitunter eine Weile, bis aktuelle Python-Versionen in den offiziellen Repositories der Distributionen aufschlagen, daher werden Sie dort eventuell noch nicht fündig. Wenn Sie eine taufrische Python-Version möchten, können Sie sich in alternativen Paketquellen umsehen. Ubuntu-Nutzer finden aktuelle Python-Pakete zum Beispiel im Deadsnakes-Repository, das Sie wie folgt hinzufügen:

```
sudo add-apt-repository ppa:deadsnakes/ppa
```

Anschließend bringen Sie die Paketliste der Repositories wieder mit `sudo apt update` auf den aktuellen Stand, sollte dies nicht bereits automatisch passiert sein. Jetzt können Sie mit `apt search` erneut nach passenden Python-Versionen suchen und diese wie oben beschrieben installieren. Falls bereits eine ältere Minorversion installiert war, kann es sein, dass der Alias `python3` weiterhin darauf zeigt. In diesem Fall erreichen Sie den Neuzugang am einfachsten, indem Sie ihn direkt ansprechen: `python3.10`

Falls Sie kein fertiges Installationspaket in der gewünschten Version für Ihre Distribution finden – oder den angebotenen Paketen nicht vertrauen –, können Sie Python auch selbst kompilieren. Das Vorgehen unterscheidet sich nicht von anderen Software-Projekten. Stellen Sie zunächst sicher, dass alle nötigen Werkzeuge und Abhängigkeiten vorhanden sind:

```
sudo apt update
sudo apt install -y build-essential ↵
 ↳checkinstall
sudo apt install libreadline-gplv2↵
 ↳-dev libncursesw5-dev libssl-dev ↵
 ↳libsqlite3-dev tk-dev libgdbm-dev ↵
 ↳libc6-dev libbz2-dev
```

Laden Sie anschließend den aktuellen Quellcode von python.org, zum Beispiel Version 3.10.2 mit `wget https://www.python.org/ftp/python/3.10.2/Python-3.10.2.tar.xz`. Entpacken Sie das Archiv (etwa `tar -xvf`

Windows schickt seine Nutzer nach der Eingabe des Befehls „python“ automatisch in den Microsoft Store, allerdings nicht immer zur aktuellen Version.

Python-3.10.2.tar.xz) und wechseln Sie in das extrahierte Verzeichnis. Führen Sie anschließend der Reihe nach die folgenden Befehle aus:

```
./configure
make
sudo make altinstall
```

Anschließend können Sie Python ausführen, in diesem Fall also `python3.10`. Mit diesen Befehlen konnten wir Python unter Ubuntu 20.04.3 LTS erfolgreich kompilieren. Falls Sie damit keinen Erfolg haben, schauen Sie am besten im Netz nach einer Schritt-für-Schritt-Anleitung, die individuell zu Ihrer Linux-Distribution passt.

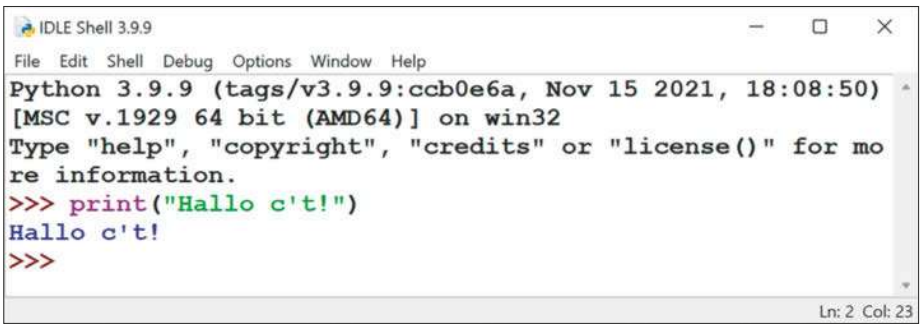
Python auf dem Mac

Geht es darum, vermeintlich veraltete Anschlüsse aus den Geräten rauszuwerfen, ist Apple gern vorne dabei: So haben iPhones keine Klinkenbuchse für Kopfhörer mehr und MacBooks hatten jahrelang keine HDMI-Buchsen. Bei Software ist Apple lange nicht so konsequent – auf einem aktuellen macOS 12.1 ist Python in der veralteten Version 2.7.18 installiert. Ende Januar 2022 wurde bekannt, dass Python künftig gar nicht mehr installiert sein wird. Was installiert ist, können Sie mit dem Befehl `python --version` auf der Kommandozeile herausfinden. Arbeiten oder gar ins Python-Programmieren einsteigen, wollen Sie mit Python 2 nicht mehr – Python 3 muss her.

Auch auf dem Mac können Sie Python über python.org als Installer herunterladen und installieren. Das geht schnell, stellt sich aber spätestens dann als unschöner Weg heraus, wenn Sie zwischen Versionen wechseln wollen (oder müssen). Empfehlenswert ist die Installation über den Mac-Paketmanager Brew [2]. Den installieren Sie vorab über den Kommandozeilen-Einzeiler:

```
/bin/bash -c "$(curl -fsSL ↵
 ↳https://raw.githubusercontent.com/↵
 ↳Homebrew/install/HEAD/install.sh)"
```

Zum Kopieren finden Sie den Befehl auch auf der Website des Projekts unter der Adresse brew.sh. Wenn Sie ohnehin gerade ins Programmieren einsteigen, werden Sie Brew schnell lieben lernen. Die Entwicklungsumgebung Visual Studio Code zum Beispiel installieren Sie ebenfalls einfach mit `brew install visual-studio-code`. Der charmante Vorteil eines Paketmanagers: Mit `brew update` und `brew upgrade` bringen Sie alle installierten Programme auf den neuesten Stand.



Über die häufig mitinstallierte IDLE Shell können Sie erste Python-Befehle direkt ausprobieren, Skripte aus Dateien laden und mehr.

Anschließend das Terminal komplett beenden und wieder öffnen. Jetzt könnten Sie Python mit `brew install python` installieren. Es gibt jedoch eine bessere Alternative, die den Umstand berücksichtigt, dass Sie zwischen Projekten die Python-Version wechseln möchten. Für diesen Zweck wurde Pyenv erfunden, das Sie per `brew install pyenv` installieren. Der Trick von Pyenv: Sie registrieren in der Konfiguration Ihrer Shell eine sogenannte Shim – der Befehl `python` wird von einem Skript abgefangen, das entscheidet, welche Python-Binärdatei es aufrufen soll. Damit das klappt, müssen Sie Pyenv in die Konfiguration der ZSH-Shell eintragen. Das ist die Shell, die Apple ab 2019 zum Standard gemacht hat. Wenn Sie eine andere Shell unter macOS nutzen, finden Sie die Anleitung in der Pyenv-Doku (siehe ct.de/yknd).

ZSH-Nutzer (also alle, die ein aktuelles macOS nutzen und sich nicht bewusst für eine andere Shell entschieden haben) führen folgende Befehle aus, um die pyenv-Shim zu registrieren:

```
echo 'eval "$(pyenv init --path)"'↵
  ↳ >> ~/.zprofile
echo 'eval "$(pyenv init -)"'↵
  ↳ >> ~/.zshrc
```

```
❯ pyenv
pyenv 2.2.3
Usage: pyenv <command> [<args>]

Some useful pyenv commands are:
   --version   Display the version of pyenv
   commands    List all available pyenv commands
   exec        Run an executable with the selected Python version
   global      Set or show the global Python version(s)
   help        Display help for a command
   hooks       List hook scripts for a given pyenv command
   init        Configure the shell environment for pyenv
   install     Install a Python version using python-build
   local       Set or show the local application-specific Python version(s)
   prefix      Display prefix for a Python version
   rehash      Rehash pyenv shims (run this after installing executables)
   root        Display the root directory where versions and shims are kept
   shell       Set or show the shell-specific Python version
   shims       List existing pyenv shims
   uninstall   Uninstall a specific Python version
   version     Show the current Python version(s) and its origin
   version-file   Detect the file that sets the current pyenv version
   version-name   Show the current Python version
   version-origin   Explain how the current Python version is set
   versions    List all Python versions available to pyenv
   whence      List all Python versions that contain the given executable
   which       Display the full path to an executable

See `pyenv help <command>' for information on a specific command.
For full documentation, see: https://github.com/pyenv/pyenv#readme
```

Pyenv macht den Wechsel zwischen Python-Versionen auf einem System komfortabler. Es wird in der Shell als sogenannte Shim registriert und lenkt Aufrufe des Befehls „python“ auf die gewünschte Python-Binärdatei um.

Danach müssen Sie das Terminal schließen (Cmd+Q). Anschließend können Sie mit Pyenv loslegen; die Funktion in aller Kürze: Der Befehl `pyenv version` verrät, auf welche Version der Befehl `python` aktuell zeigt. Mit `pyenv install 3.10.2` installieren Sie die aktuelle Version 3.10.2. Der Aufruf von `pyenv global 3.10.2` macht diese Version für den aktuellen Nutzer zum Standard. Mit `pyenv local 3.10.1` verlangen Sie im aktuellen Kontext eine andere Version, ohne die globale Einstellung zu ändern. `pyenv versions` zeigt Ihnen, welche Versionen gerade installiert sind. Mit einer Entwicklungsumgebung (siehe S. 26) und einer passenden Python-Erweiterung kann der Wechsel sogar noch bequemer funktionieren. In Visual Studio Code zum Beispiel finden Sie mit installiertem Pyenv und der auf Seite 28 vorgestellten Python-Erweiterung in der unteren Leiste ein Auswahlmenü für installierte Versionen.

Versionen und Module ohne Chaos

Python-Skripte nutzen meist Module, die Funktionen bereitstellen, um ein Rad nicht neu erfinden zu müssen. Im offiziellen Python Package Index (pypi.org) finden Sie inzwischen Hunderttausende davon für jeden Zweck, die Sie leicht installieren können. Der Teufel steckt jedoch im Detail, denn es ist durchaus möglich, dass zwei Skripte auf Ihrem System nicht nur unterschiedliche Python-Versionen, sondern auch unterschiedliche Modulversionen voraussetzen. Und wenn man immer wieder die vom gerade ausprobierten Skript geforderten Module nachrüstet, verliert man leicht den Überblick.

Damit das nicht im Chaos endet, arbeiten Sie am besten von Anfang an mit virtuellen Umgebungen (Virtual Environments, kurz venv). Es bietet sich an, für jedes neue Projekt ein solches venv zu erstellen. So wird jedes Skript mit den exakt passenden Zutaten versorgt, die es braucht. Das klingt komplizierter, als es ist: Wenn Sie ein venv erstellen, erzeugen Sie damit im Wesentlichen einen neuen Ordner. Aktivieren Sie das venv, arbeitet Python anschließend mit diesem Ordner und speichert neu installierte Module ausschließlich darin. Diese sind nur innerhalb der jeweiligen virtuellen Umgebung erreichbar und kommen so der globalen Python-Installation und anderen Skripten nicht in die Quere. Neben den Modulen ist die Python-Version durch das venv vorgegeben, damit auch alles zusammen passt.

Seit Python 3.3 gehört das venv-Modul zum Lieferumfang; eine neue virtuelle Umgebung namens env (können Sie beliebig ändern) entsteht mit nur einem Befehl:

```
python3 -m venv env
```

Er legt in diesem Beispiel ein Verzeichnis env an, und zwar unterhalb des Verzeichnisses, in dem Sie sich gerade befinden. Führt der Befehl unter Linux zu einem Fehler, müssen Sie noch das Paket python[version]-venv mit `sudo apt install` aus dem Repo installieren. Alternativ können Sie pip nutzen: `pip3 install virtualenv`. Danach können Sie in die virtuelle Umgebung wechseln. Unter Linux und macOS genügt zur Aktivierung des venv der folgende Befehl: `source env/bin/activate`. Unter Windows führen Sie das Skript activate.bat aus, das venv für Sie angelegt hat, im einfachsten Fall mit `env\Scripts\activate.bat`.

Anschließend erscheint `env`, der Name der Umgebung, in Klammern vor der Eingabezeile. Daran erkennen Sie, dass Sie im venv unterwegs sind. Jetzt können Sie ein beliebiges Python-Modul installieren, um die virtuelle Umgebung zu testen – zum Beispiel `requests`, das HTTP(S)-Anfragen absetzen kann. Nutzen Sie dazu den Paketinstaller pip: `pip3 install requests`.

Wenn der Befehl nicht funktioniert, können Sie den Paketmanager über `python3 -m pip` erreichen. Klappt auch das nicht, müssen Sie `pip` zunächst nachinstallieren. Sie finden ihn zum Beispiel als `python3-pip` im Ubuntu-Repository (siehe oben). Falls sich `pip` beschwert, dass es nicht auf dem aktuellen Stand ist, können Sie es mit `pip3 install --upgrade pip` erneuern.

Nach der Installation des `requests`-Moduls innerhalb des venv finden Sie es im Dateisystem unterhalb von env\Lib\site-packages, wo es nur für das venv erreichbar ist. Mit `deactivate` können Sie das venv jederzeit wieder verlassen. Um es zu löschen, können Sie einfach dessen Ordner eliminieren.

Wenn Sie fremde Skripte ausführen möchten, ist die Wahrscheinlichkeit sehr hoch, dass es Module voraussetzt, die Sie noch nicht installiert haben. Diese sind üblicherweise in einer Datei namens requirements.txt aufgelistet, die dem Skript beiliegt. Doch keine Sorge, Sie müssen diese Liste nicht von Hand abklappern, stattdessen können Sie den Paketmanager damit füttern: `pip3 install -r requirements.txt`

Kleine Versionskunde

Hinter Versionsnummern von Python wie **3.10.2** steckt ein System, das sogenannte Semantic Versioning [3]. Nur sehr selten ändert sich der erste Teil, die Major-Version (im Beispiel die **3**). Häufiger gibt es eine sogenannte Minor-Version, also im Beispiel **10**. Jede Minor-Version bringt neue Funktionen mit. Die Python-Software-Foundation gibt üblicherweise im Jahresrhythmus solche Minors heraus, der angepeilte Zeitpunkt für das nächste Update ist der Oktober. Am Ende steht eine Patchversion wie **2**, die in diesem Fall besagt, dass die Foundation schon zweimal mit Bugfixes nachgebessert hat. Solche Bugfixes erscheinen auch für ältere Minor-Versionen (in der Regel alle zwei Monate über einen Zeitraum von insgesamt 18 Monaten), mit Sicherheitsupdates kann man sogar fünf Jahre lang rechnen.

Auch wenn die Python-3-Versionen grundsätzlich abwärtskompatibel sind, ist es eine gute Idee, ein Skript mit der Minor-Version auszuführen, für die es entwickelt wurde. Bei der Installation von Python 3.10 wird eine ältere Version 3.9 daher auch nicht ersetzt, stattdessen wird die neue Minorversion zusätzlich installiert. Das steinalte Python 2 sollten Sie wenn möglich meiden, da es schon über zwei Jahre nicht mehr mit Sicherheitsupdates versorgt wird. Beachten Sie allerdings, dass Sie nicht alle Python-2-Skripte mit Python 3 ausführen können, weil Funktionen geändert oder abgeschafft wurden – teilweise ist Handarbeit im Code nötig. Demnächst läuft auch für Python 3.7 die Zeit ab: Laut aktueller Planung ist am 27. Juni 2023 mit Sicherheitsupdates Schluss.

Einen guten Überblick über die Lebenszyklen der einzelnen Versionen liefert der englischsprachige Wikipedia-Artikel „History of Python“ (siehe ct.de/yknd).

Der Paketmanager installiert anschließend nach und nach die geforderten Python-Module. Außerhalb eines venv sorgt `--user` dafür, dass Module nur für den aktiven User und nicht systemweit installiert werden, wodurch sie auf `sudo` verzichten können.

Weitere nützliche `pip`-Kommandos finden Sie in der Doku (siehe ct.de/yknd). Mit `pip3 freeze > requirements.txt` etwa generieren Sie eine Liste der installierten Module, die Sie mit Ihrem Skript weitergeben können. Mit `pip3 uninstall modulname` werden Sie ein Modul wieder los.

Mehr Komfort mit pipenv

Das Hantieren mit den virtuellen Umgebungen geht schnell in Fleisch und Blut über. Wenn Sie sich das Leben damit vereinfachen möchten, kann sich ein Blick auf pipenv lohnen, das virtuelle Umgebungen und Paketinstallationen auf komfortable Weise kombiniert. Sie installieren es mit `pip3 install pipenv`.

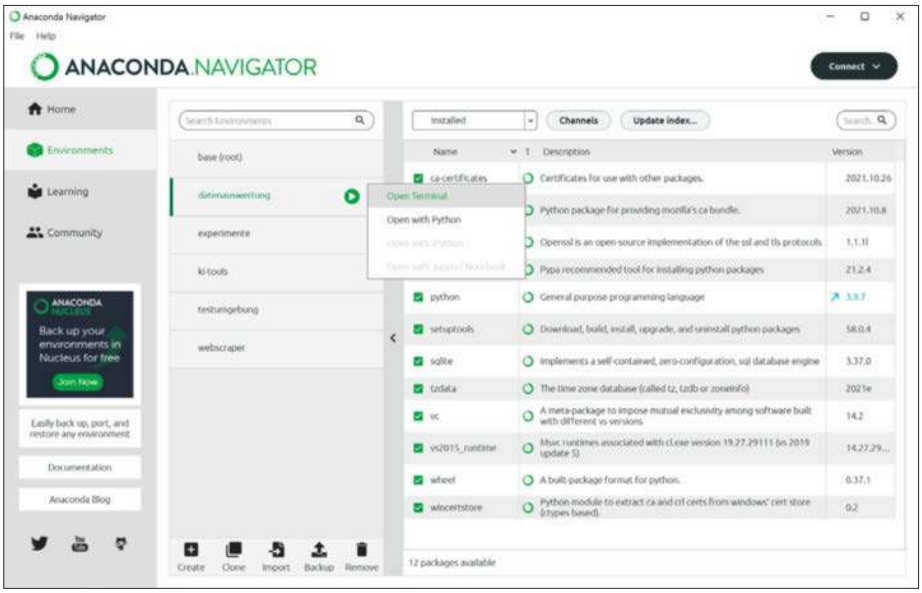
Falls pip dabei eine Warnung anzeigt, laut der das Installationsverzeichnis nicht in der Umgebungsvariable PATH steht, müssen Sie den angegebenen Pfad noch dem PATH hinzufügen und sich neu anmelden, damit die folgenden Befehle funktionieren (siehe ct.de/yknd).

Anschließend erstellen Sie ein Verzeichnis für Ihr Projekt und wechseln mit `cd` hinein. Wenn Sie jetzt mit `pipenv install paketname` ein beliebiges Paket installieren, legt pipenv zunächst automatisch eine neue Umgebung an. Im Anschluss wird das gewünschte Paket heruntergeladen und installiert.

Mit `pipenv shell` starten Sie eine Shell in der virtuellen Umgebung und mit `pipenv run skript.py` können Sie von außen direkt ein Skript darin ausführen. Das Tool hat diverse weitere Tricks auf Lager und kann Ihr Projekt etwa mit `pipenv check` auf Sicherheitslücken abklopfen. Der folgende Befehl generiert Ihnen eine requirements.txt: `pipenv lock -r > requirements.txt`. Weitere Kniffe finden Sie in der Dokumentation (siehe ct.de/yknd).

Bequeme Anaconda

Sie kennen jetzt die wichtigsten Grundlagen, um Python auf Ihrem System an den Start zu bringen und damit zu arbeiten. Die Einrichtung ist mit etwas Handarbeit verbunden, bietet Ihnen aber auch die Gelegenheit, die grundlegenden Abläufe zu verstehen. Mit Anaconda existiert auch ein umfangreiches Python-Installationspaket für Windows, Linux und macOS, das Ihnen die Arbeit hinter den Kulissen weitestgehend abnimmt, sodass Sie sich ganz auf Ihre Skripte konzentrieren können. Es ist mit über 500 MByte jedoch nicht gerade schlank und Sie bekommen wahrscheinlich mehr, als Sie brauchen: Neben Python bringt Anaconda gleich noch die Entwicklungsumgebung (IDE) Spyder, die Python-Experimentierumgebung Jupyter Notebook und etliche Python-Pakete mit.



Der Python-Installer Anaconda bringt ein GUI-Tool zur Verwaltung von Modulen und virtuellen Umgebungen mit.

Anaconda wurde für Datenauswertungen und KI-Experimente zusammengestellt, insbesondere das GUI-Programm Anaconda Navigator dürfte aber auch für Einsteiger interessant sein: Über „Environments“ kann man damit per Mausklick virtuelle Umgebungen erstellen, die eine wählbare Python-Version nutzen. Ist diese noch nicht auf dem System installiert, erledigt der Manager dies automatisch. Mit einem Klick auf den Play-Knopf öffnet man ein Terminal oder eine Python-Shell in der ausgewählten Umgebung. Die jeweils installierten Modulpakete listet das Tool grafisch auf, zudem kann man die verfügbaren Pakete komfortabel durchsuchen und nachrüsten. Hinter den Kulissen arbeitet ein eigener Paketmanager namens `conda`.

Den Komfortgewinn zahlen Sie mit Abhängigkeit: Anaconda ist kommerzielle Software. Die Firma Anaconda Inc. bietet eine kostenlose „Individual Edition“ an, Unternehmen ab einer Größe von 200 Mitarbeitern müssen jedoch zu einer der kostenpflichtigen Editionen greifen. Es gibt keine Garantie dafür, dass sich die aktuellen Nutzungsbedingungen in der Zukunft nicht zu Ihrem Nachteil ändern.

Fazit

Die Einrichtung von Python ist schnell geschafft, ganz gleich, welche der vielen Installationswege Sie nutzen. Und wenn Sie von Beginn an mit virtuellen Umgebungen arbeiten, verlieren Sie auch nicht den Überblick im Versionsdschungel. Da Python-Skripte einfache Textdateien sind, können Sie diese mit jedem beliebigen Editor erstellen und bearbeiten – besonders komfortabel ist das jedoch nicht.

Im nächsten Schritt sollten Sie sich daher nach einer IDE wie Sublime, PyCharm, Spyder oder Visual Studio Code umsehen, die Ihnen die Arbeit durch Autovervollständigung, Syntaxhervorhebung, Versionierung und vieles mehr deutlich erleichtern kann. Im folgenden Artikel auf Seite 26 stellen wir die wichtigsten IDEs vor, suchen Sie sich einfach die für Sie passende aus. *(rei@ct.de)* c't

Literatur

[1] Hajo Schulz, Überholspur, Windows-Tools für Tastatur-Fans, c't 3/2022, S. 166
[2] Immo Junghärtchen, App Store fürs Terminal, Die Paketverwaltung Homebrew unter macOS einrichten und nutzen, c't 12/2020, S. 122
[3] Jan Mahn, Bedeutung 2.0.0, Warum Versionsnummern nicht willkürlich sind, c't 24/2021, S. 128

Python-Downloads & Doku: ***ct.de/yknd***