Module, Packete und PIP

Eric Niklas Wolf, Moritz Pflügner

Python-Kurs

23. November 2021

iFSR
Fachschaftsrat
Informatik

Gliederung

Module

- Ein Modul ist die python-interne Repräsentation einer `.py` Datei
- Der Dateiname setzt sich daher aus Modulname + `.py` zusammen
- Module beinhalten eine beliebige Anzahl an Definitionen (Klassen, Funktionen, Variablen/Konstanten)
- Das komplette Modul wird mit `import` hinzugefügt
- Einzelne Inhalte mit `from` Modul `import` Name
- Mit Hilfe von `as` kann ein Alias für den importierten Namen angelegt werden

Ein Beispiel

Die Datei **incdec.py** enthält eine Reihe von Funktionen:

```
1 def increment(a):
2     return a+1
3
4
5 def decrement(a):
6     return a-1
```

Ein Beispiel

In einem anderen Python-Script kann ich diese Funktionen nutzen:

```
1  # Importieren des kompletten Moduls incdec
2  import incdec
3
4  # Die Funktionen koennen wie folgt aufgerufen werden:
5  incdec.increment(3)
6  # => 4
7  incdec.decrement(3)
8  # => 2
9
10 # Importieren einzelner Funktionen
11 from incdec import increment
12
13 # Diese kann jetzt sofort aufgerufen werden
14 increment(3)
15 # => 4
```

Ein Beispiel

In einem anderen Python-Script kann ich diese Funktionen nutzen:

```
1 # Alias fuer Module verwenden
2 import incdec as plusoneminusone
3
4 plusoneminusone.inrement(3)
5 # => 4
```

Der Sinn der Boilerplate

- Verhindert, dass Code beim Importieren eines Scriptes ausgeführt wird
- Beim Aufruf des Moduls über `import` ist der Name des Moduls nicht `__main__`, sondern der Name des Scriptes (im vorherigen Beispiel wäre das `incdec`)

Suchpfad für Module

- Python sucht Module beim Import an allen im **PYTHONPATH** aufgelisteten Ordnern
- dieser findet sich in Python unter `sys.path`, in der Kommandozeile in der Umgebungsvariable PYTHONPATH
- `sys.path` lässt sich zur Laufzeit im Interpreter ändern (z.B. um nachträglich neue eigene Module hinzuzufügen)

Suchpfad für Module

Standardmäßig enthält der Suchpfad folgende Module:

- die *Standardbibliothek* der derzeitig verwendeten Python-Version
- das *aktuelle Verzeichnis*, in dem der Interpreter aufgerufen wurde
- eine *Verzeichnis mit plattfomspezifischen Modulen*, z.B. /usr/local/Cellar/python3/3.5.1/Frameworks/Python.framework/Versions/3.5/lib/python3.5/plat-darwin" für Mac
- *Benutzerspezifische Module für die jeweilige Python Version*, z.B. /usr/local/lib/python3.5/site-packages" für Python 3.5

Suchpfad modifizieren

Zusätzliche Verzeichnisse für den PYTHONPATH kann man beim Aufruf übergeben:

```
$ PYTHONPATH=/my/directory python3 script.py
```

Oder im Programm:

```
1 import sys
2
3 sys.path.append('my/directory')
```

Änderungen am Pfad sind erst **nach** der entsprechenden Codezeile verfügbar.

Standardmodule

Python liefert viele nützliche Module bereits in der Standardbibliothek mit, z.B.: `sys`, `os`, `http`, `re`, `fuctools`, `itertools`, `collections`, `hashlib`, `urllib` und viele mehr.

Eine Liste aller Module findet man in der offiziellen Dokumentation. Daher bezeichnet man Python oft auch als **Batteries included**.

Packages

- Packages sind Ordner, die mindestens ein `__init__.py` Modul enthalten.
- der Inhalt dieses Moduls ist prinzipiell egal
- Packages können genau wie Module importiert werden
- Wird das Package selbst importiert, sind alle Definitionen in `__init__.py` über das importierte Package erreichbar.
- wird benutzt, um Module in sinnvolle Gruppen zusammenzufassen

Installation von PIP

Nicht jedes Modul ist in der Standardbibliothek vorhanden, muss aber nicht zwingen manuell als Datei hinzugefügt werden. Man kann ganz einfach PIP zur Hilfe nehmen.

Installation:

1. Downloaden der `get-pip.py` von pypa (hier)
2. Installieren via `python3 get-pip.py`

Hinweis für Mac-Nutzer:
Bei der Pythoninstallation via *Homebrew* wird pip gleich mit installiert.

Verwendung

Die Installation von Modulen ist einfach und es gibt mehrere Möglichkeiten:

- die aktuellste Version eines Moduls:
  ```
  pip install Modulname
  ```
- eine bestimmte Version:
  ```
  pip install Modulname == 2.1
  ```
- alle Versionen ab einem Minimum:
  ```
  pip install Modulname >= 2.1
  ```
- alle Versionen bis zu einem Maximum:
  ```
  pip install Modulname <= 2.1
  ```

Die requirements.txt

Wenn jemand Scripte ausführen möchte, in denen Module verwendet werden, die mit Hilfe von PIP installiert wurden, ist es hilfreich eine `requirements.txt` mitzuliefern.
In dieser sind alle Module aufgelistet, die benötigt werden.

```
pathlib>=1.0
pyyaml>=3.08
markdown
```

Via `pip install -r requirements.txt` können alle Module von der Liste installiert werden