Empfohlene Unterrichtsinhalte als Vorbereitung für dieses Thema: 01-03 (Variablen), 02-01 (Strings), 03-04 (Zufallszahlen), 05-01 (Funktionen)

Einführung in die computerlinguistische Programmierung mit Python

05-03: Module 🧩

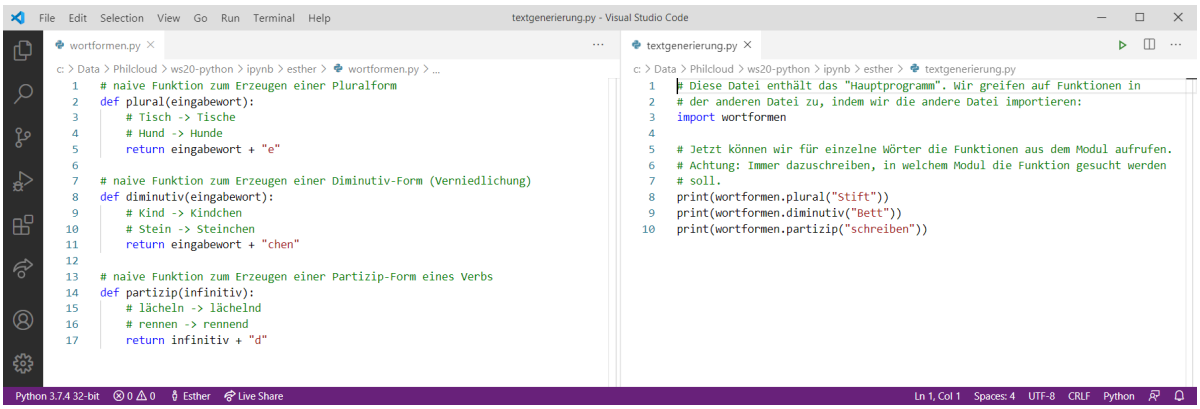
Wir haben vor kurzem begonnen, Sinnabschnitte unserer Programme in Funktionen zu verpacken. Funktionen haben einige Vorteile, zum Beispiel, dass sie komplexen Code übersichtlicher und lesbarer machen. Wir können verschiedene Sinnabschnitte klar voneinander trennen und die Interaktion zwischen den einzelnen Bestandteilen in unserem Programm deutlich sichtbar machen.

Wir können noch einen Schritt weitergehen und unsere Funktionsdefinitionen in **separate Dateien** (wir sagen: **Module**) auslagern. Das kann nützlich sein, falls wir zum Beispiel sehr viele Funktionen für unser Projekt brauchen und einen klaren Einstiegspunkt für das Programm definieren wollen. Dann kann beispielsweise der Einstiegspunkt eine Pythondatei sein, die nacheinander auf alle Funktionen in einer zweiten Pythondatei zugreift.

Damit wir innerhalb einer Pythondatei auf Funktionen aus einer zweiten Pythondatei zugreifen können, müssen wir die zweite Datei zu Beginn unseres Codes **importieren**. Wir kennen das schon vom Modul `random`, das wir verwendet haben, um Zufallszahlen zu erzeugen.

Beim Importieren von Modulen können wir keine komplexen Pfade angeben. Daher achten wir darauf, dass Pythondateien, zwischen denen etwas importiert wird, **am gleichen Ort gespeichert** sind. Beide Dateien müssen eine `.py`-Dateiendung haben.

Im folgenden Beispiel sammeln wir einige Funktionen zum Erzeugen neuer Wortformen in einem Modul namens `wortformen(.py)`. Diese Datei enthält **nur Funktionsdefinitionen**, aber keine eigenen Funktionsaufrufe. Das Modul kann jetzt von einem anderen Programm aus importiert werden, damit wir von dort auf die Funktionsdefinitionen zugreifen können. Das sieht dann etwa so aus:



Ihr findet die beiden Dateien auf der Kurswebseite und könnt sie im gleichen Verzeichnis abspeichern, um euch zu überzeugen, dass wir fehlerfrei auf die Funktionen der importierten Datei zugreifen können.

Die Zeile, die mit `import` beginnt, ist dafür zuständig, den Interpreter zu informieren, dass wir **vordefinierte Funktionen aus einer anderen Pythondatei laden** wollen. Wir schreiben nur den Namen der Datei und lassen beim Importieren die Dateiendung weg.

Man kann übrigens **beliebig viele Module** importieren. Dazu schreibt man mehrere `import`-Zeilen untereinander:

```
import utils
import user_model
import other_project
```

Im weiteren Verlauf einer Datei mit diesen Import-Statements stehen alle Funktionen aus allen importierten Modulen zur Verfügung. Wir verwenden importierte Funktionen immer nach diesem Muster:

```
<modul>.<funktion>()
```

Warum die Schreibweise mit dem Modulnamen wichtig ist, wird in Thema 05-04 besprochen.

Was passiert beim Importieren?

Wenn eine `import`-Zeile ausgeführt wird, liest der Interpreter die **gesamte importierte Datei** ein und führt alle enthaltenen Befehle aus. Wenn das importierte Modul beispielsweise eine `print()`-Anweisung enthält, sehen wir die entsprechende Ausgabe sofort, wenn wir das Modul importieren. Das ist allerdings schlechter Stil und sollte vermieden werden.

Es ist außerdem wichtig, dass `import`-Statements am **Anfang des Codes** platziert werden! Beim Importieren verändern wir den Zustand des Interpreters. Damit das nicht mittendrin passiert und zu Problemen führt, passieren Imports vor allem anderen. Mehr zu den möglichen Problemen besprechen wir in Thema 05-04: Namenskonflikte.

Warum importieren?

Wir könnten die drei Funktionsdefinitionen für die Wortformen auch alle in unsere Hauptprogrammdatei schreiben. Die Datei wäre dann aber **sehr lang** und es wäre immer schwieriger, sich zu orientieren. In echten Anwendungskontexten kommt es manchmal vor, dass einzelne Pythondateien mehrere hundert oder sogar mehrere tausend Zeilen haben! Das Aufteilen von Aufgaben in Module, die **Sinneinheiten** bilden, hilft uns, die Strukturierung der Aufgabe von der Vorbereitung der notwendigen Werkzeuge zu trennen.

Im Beispiel mit den Wortformen ist es leicht vorstellbar, dass wir die Funktionen auf eine **andere Sprache** übertragen wollen. Dann wissen wir, dass wir nur das Modul `wortformen` anpassen müssen und andere Regeln für die Erzeugung der jeweiligen Formen brauchen.

Schließlich hilft uns die Zerlegung von Projekten in einzelne Module auch dabei, im **Team** zu arbeiten und an separaten Aspekten einer Aufgabe gleichzeitig zu arbeiten. Für die Person, die an der Datei `textgenerierung` arbeitet, ist es nicht so wichtig, *wie* die Wortformen gebildet werden, sondern nur, dass die Funktionen eine passende Rückgabe liefern.

Zusammenfassung

- Pythondateien können auf Funktionsdefinitionen zugreifen, die in anderen Pythondateien stehen, indem wir die anderen Dateien als Module importieren.
- Module, die importiert werden, müssen im gleichen Verzeichnis liegen wie die aktive Pythondatei.
- Wir können eingebaute Module importieren, wie `random`, oder unsere eigenen Module schreiben und importieren.
- Wir können mehrere Module gleichzeitig importieren.
- Import-Statements gehören an den Anfang des Codes.
- Beim Importieren wird der Code in der externen Datei einmal vom Interpreter durchgearbeitet. Wir müssen gut aufpassen, dass beim Import nur das passiert, was wir brauchen!

Weitere Themen dieser Woche

- 05-01: Funktionen
- 05-02: None
- 05-04: Namenskonflikte