FFHS
Fernfachhochschule Schweiz
Mitglied der SUPSI

Module und Packages

Modul PyFr

Urs-Martin Künzi

9. März 2024

Inhalt

Module

Packages

Konkrete Module

- math
- Datum und Zeitdauern

Module

- Ein Modul ist eine Datei mit Python Code.
- Ein Modul enthält Klassen, Funktionen und/oder Variable.
- Der Dateiname eines Moduls enthält die Erweiterung .py.
- Es gibt
  - eingebaute Module
  - Module von Drittherstellern
  - eigene Module

Einige eingebaute Module

- `sys`
- `platform`
- `math`
- `datetime`

Import von Modulen

- Um ein Modul zu verwenden, muss es importiert werden:
  `import` *modulname*.
  Beim Modulnamen muss .py weggelassen werden.
- Um nach dem Import auf ein Element xyz zuzugreifen, wird folgende Syntax verwendet: *modulname.xyz*.
- Einem Modul kann beim Import ein neuer Name zugeordnet werden:
  `import` *modulname* as *modAl*
  Zugriff auf Modulelemente: *modAl.xyz*.
- `dir`(*modul*) liefert alle Elemente eines Moduls.

Import von Modulelementen

- Statt ein ganzes Modul zu importieren,
  können einzelne Elemente importiert werden:
  `from` *modul* `import` *Element*.
  Es können auch mehrere Elemente importiert werden:
  `from` *modul* `import` *Element1, Element2,...*.
- Auf die importierten Elemente kann direkt zugegriffen werden:
  Element, nicht *modul.Element*.
- Es können alle Elemente importiert werden mit
  `from` *modul* `import` *.
- Beim Import aller Elemente ist etwas Vorsicht geboten:
  Bestehende Variablen, Funktionen oder Klassen können
  (unbeabsichtigt) überschrieben werden.

Packages

- Mit Packages können Module gruppiert werden.
- Ein Package ist ein Verzeichnis von Modulen und Subpackages.
- Module in Packages können wie folgt importiert werden:


```
import pkg.mod1              # Zugriff pkg.mod1
import pkg.mod1 as m1        # Zugriff m1
from pkg import mod1         # Zugriff mod1
from pkg import mod1 as m1 # Zugriff m1
```

- Wird ein Package importiert, werden die darin enthaltenen Module und Subpackages nicht importiert.
  `import pkg` ist zwar korrekt, aber (hier) sinnlos.

`__init__.py`

- Ein Package kann eine Datei `__init__.py` mit Python-Code enthalten.
- Dieses wird ausgeführt, wenn das Package geladen wird.
- Häufig enthält `__init__.py` Imports für die im Package enthaltenen Module und Subpackages.
  Dann ist eine Anweisung
  `import pkg`
  sinnvoll.

Suchpfad

- Die Module und Packages werden gemäß den Einträgen der Liste `sys.path` gesucht.

```
import sys
print(sys.path)
```

- `sys.path` enthält folgende Verzeichnisse:
  - `sys.path` enthält die Verzeichnisse mit den Systemmodulen.
  - `sys.path` enthält das aktuelle Verzeichnis beim Starten von Python.
    - Damit werden eigene Module, die sich im Verzeichnis befindet, aus dem Python gestartet wird, gefunden.
  - `sys.path` enthält Pfade aus der Umgebungsvariable `PYTHONPATH`
- `sys.path` kann zur Laufzeit geändert werden (durch `sys.path.append(...)`).

Ausführung von Modulen auf der Kommandozeile

- Ein Python-Modul kann über die Kommandozeile aufgerufen werden durch
  `python` *Dateiname*
- Beispiele:

```
python mod1.py
python pkg/mod1.py # oder pkg\mod1.py
```

- Ein Modul kann auch durch die Angabe des (qualifizierten) Modulnamnes aufgerufen werden.
  Dabei muss der Modulname im Suchpfad sein.
- Beispiele:

```
python -m mod1
python -m pkg.mod1
```

Modulname

- Ein importiertes Modul `mod` besitzt einen Namen `__name__`. `__name__` ist der qualifizierte Modulname.
- Wird ein Modul mit dem python Kommando auf der Kommandozeile gestartet, so lautet sein Name `__Name__ == "__main__"`.
- Mit der Abfrage `__name__ == "__main__"` kann somit entschieden werden, ob das Modul importiert wurde oder via Kommandozeile gestartet wurde.

Importiertes Modul vs Hauptmodul

Beispiel

Das Modul `fakultaet.py` sei wie folgt definiert

```
def fak(n):
    if n <= 1: return 1
    return n * fak(n-1)

if __name__ == "__main__":
    import sys
    print(fak(int(sys.argv[1])))
```

- Wird das Modul importiert, dann wird einfach die Funktion `fak` definiert.
- Wird das Modul mit
  `python fakultaet.py 5`
  gestartet, wird `fak(5)`, also 120 ausgedruckt.

Modul math

- Das math-Modul enthält (unter anderen) die folgenden mathematischen Funktionen:
  `cos, sin, tan, acos, asin, atan, atan2,`
  `pow, prod, remainder,`
  `sqrt, exp, log, log10, log2,`
  `ceil, floor, trunc,`
  `isclose, isfinite, isinf, isnan`
  `comb, perm, factorial, gcd.`
- Konstanten:
  `e, inf, nan, pi`
- Die Funktionen `min, max` und `abs` sind direkt verfügbar und nicht über das Modul `math`.

Datum und Zeit

- Zur Behandlung von Datum und Zeit existiert das Modul `datetime`.
- Dokumentation: https://docs.python.org/3/library/datetime.html.
- Wichtigste Klassen in diesem Modul:
  - `datetime`
  - `date`
  - `time`
  - `timedelta`

Klasse time

- `import datetime as dt`
- Konstruktor: `dt.time(`*hour*, *minute*, *second*`)`
  (second ist optional)
- Aktuelle Zeit: `dt.datetime.now().time()`
- Aus ISO 8601-String: `dt.time.fromisoformat(`*iso_string*`)`
- In ISO 8601-String transformieren: *zeit*`.isoformat()`
- Zugriff auf Attribute eines Zeitobjekts `zeit`:
  - *zeit*`.hour`
  - *zeit*`.minute`
  - *zeit*`.second`

Klasse date

- `import datetime as dt`
- Konstruktor: `dt.date(year, month, day)`
- Aktuelles Datum: `dt.datetime.now().date()`
- Zugriff auf Attribute eines Datumobjekts *datum*:
  - *datum*`.year`
  - *datum*`.month`
  - *datum*`.day`
- Methoden:
  - `dt.date.fromisoformat(`*iso_string*`)`
  - *datum*`.isoformat()` (Stringdarstellung yyyy-mm-dd)
  - *datum*`.isoweekday()` (Zahl, Mo=1, So=7)
  - *datum*`.weekday()` (Zahl, Mo=0, So=6)

Klasse datetime

- `import datetime as dt`
- Aktuelles Datum: `dt.datetime.now()`
- Konstruktor:
  - `dt.datetime(`*year*, *month*, *day*, *hour*, *minutes*, *seconds*`)`
    (Parameter ab `hour` sind optional)
- `datetime` ist Subklasse von `date`, alle Attribute und Methoden von `date` existieren auch für `datetime`.
- Zugriff auf weitere Attribute eines datetime Objekts `datum`:
  - *datum*`.hour`
  - *datum*`.minute`
  - *datum*`.second`

datetime Methoden

- `dt.datetime.combine(datum, zeit)`
- `datum_zeit.date()`
- `datum_zeit.time()`
- `datum_zeit.isoformat()`:
  Stringdarstellung yyyy-mm-ddTHH:MM:SS
  (T ist Trennzeichen zwischen Datum und Zeit;
  kann mit optionalem Argument `sep` geändert werden)
- `dt.datetime.fromisoformat(`*iso_string*`)`
  Umkehrung von isoformat()
- `datum_zeit.timestamp()`:
  Sekunden seit 1.1.1970 UTC
- `dt.datetime.fromtimestamp()`:
  Umkehrung von `timestamp()`:

Formatierung von Datum und Zeit

- Mit der `str`-Funktion oder `isoformat`-Methode werden Daten im Iso-Format dargestellt bzw. mit der `fromisoformat`-Methode werden Datums-Objekte aus Strings erzeugt.
- Für sprachspezifische Darstellungen gibt es die beiden Methoden
  - `strftime(`*format_string*`)`:
    Erzeugt eine Stringdarstellung gemäß angegebenem Format-String
  - `strptime(`*datums_str*, *format_string*`)`:
    Erzeugt aus eine String-Repräsentation eines Datums gemäß Format-String ein datetime-Objekt

Format-Strings

Code	Erklärung	Beispiel
%y	Year, short version, without century	18
%Y	Year, full version	2018
%b	Month name, short version	Dec
%B	Month name, full version	December
%m	Month as a number 01-12	02
%d	Day of month	31
%H	Hour 00-23	17
%I	Hour 00-11	05
%p	AM/PM	PM
%M	Minute	41
%S	Second	08

Format-Strings, Fortsetzung

Code	Erklärung	Beispiel
%a	Weekday, short version	Wed
%A	Weekday, full version	Wednesday
%w	Weekday as a number 0-6, Sunday is	0
%u	ISO weekday (1-7), Sunday is	7
%f	Microsecond	548513
%j	Day number of year 001-366	365
%U	Week number of year, Sunday as the first day of week, 00-53	52
%W	Week number of year, Monday as the first day of week, 00-53	52
%C	Century	20
%%	% Symbol	%

Sprachanpassungen

- Um für Wochentage und Monatsnamen eine bestimmte Sprache zu verwenden, kann man das Modul `locale` verwenden.
- `import locale`
  `locale.setlocale(locale.LC_ALL, '')`
  Damit wird die auf dem Betriebssystem definierte Sprache verwendet
- `locale.setlocale(locale.LC_ALL, 'de_CH')`
  Damit wird Deutsch (CH) verwendet
- Eine Alternative zu `locale` ist das `babel`-Package

Klasse timedelta

- Der Unterschied zwischen zwei datetimes ist ein timedelta
- Konstruktor: `timedelta(days=tage, hours=stunden, minutes=minuten, seconds=sekunden)`
- Attribute entsprechen den Konstruktor-Parametern
- datetime - datetime = timedelta
- datetime $\pm$ timedelta = datetime
- timedelta $\pm$ timedelta = timedelta
- zahl * timedelta = timedelta