Modul datetime – mit Datum und Zeit jonglieren

Im Folgenden nutzen wir das Modul `datetime`. Nach dem Import lassen wir uns die Klassen etc. ausgeben über `dir(datetime)`

```
import datetime
print(dir(datetime))
```

Wir erhalten diese Ausgabe:

```
['MAXYEAR', 'MINYEAR', '__builtins__', '__cached__', '__doc__', '__file__',
'__loader__', '__name__', '__package__', '__spec__', 'date', 'datetime',
'datetime_CAPI', 'sys', 'time', 'timedelta', 'timezone', 'tzinfo']
```

Wir verfügen nach diesem kompletten Import über die Klassen:

- date
- datetime
- time
- timedelta
- timezone
- tzinfo

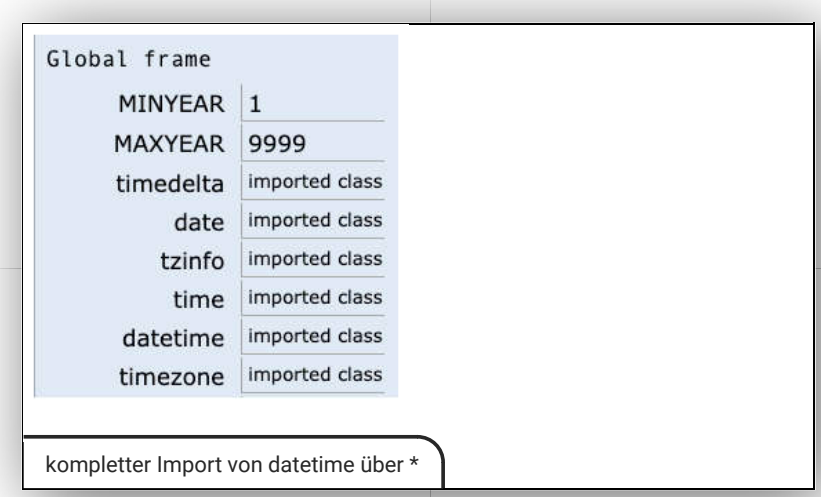
Hier eine kurze Beschreibung. Anhand der folgenden Beispiele werden die Einsatzmöglichkeiten schnell klar.

Klasse	Beschreibung
date	Datumsklasse mit Jahr (year), Monat (month) und Tag (day). Voraussetzungen sind der gregorianische Kalender (und dass dieser gilt)
time	idealisierte Zeit mit den Attributen Stunde (hour), Minute (minute), Sekunde (second) und Microsekunde (microsecond). Voraussetzung ist, dass jeder Tag aus exakt 24 * 60 * 60 Sekunden besteht. Idealisiert deshalb, weil es keine Schaltsekunde gibt.
datetime	Kombination aus date und time. Attribute und Voraussetzung entsprechenden den einzelnen Klassen.

Klasse	Beschreibung
timedelta	Zum Berechnen der Zeitdauer zwischen 2 Zeitpunkten. Rückgabe in Mikrosekunden.
timezone	Zeitzonen und UTC. Die abstrakte Basisklasse tzinfo wird von timezone genutzt

Importieren wir datetime über `from datetime import *` erhalten wir alle verfügbaren Klassennamen:

```
from datetime import *
print(dir(datetime))
```



kompletter Import von datetime über *

```
['__add__', '__class__', '__delattr__', '__dir__', '__doc__', '__eq__',
'__format__', '__ge__', '__getattribute__', '__gt__', '__hash__', '__init__',
'__init_subclass__', '__le__', '__lt__', '__ne__', '__new__', '__radd__',
'__reduce__', '__reduce_ex__', '__repr__', '__rsub__', '__setattr__',
'__sizeof__', '__str__', '__sub__', '__subclasshook__', 'astimezone',
'combine', 'ctime', 'date', 'day', 'dst', 'fold', 'fromordinal', 'fromtimestamp',
'hour', 'isocalendar', 'isoformat', 'isoweekday', 'max', 'microsecond',
'min', 'minute', 'month', 'now', 'replace', 'resolution', 'second', 'strftime',
'strptime', 'time', 'timestamp', 'timetuple', 'timetz', 'today', 'toordinal',
'tzinfo', 'tzname', 'utcfromtimestamp', 'utcnow', 'utcoffset',
'utctimetuple', 'weekday', 'year']
```

Man sieht auch sehr schön, wenn wir nur einen Teil (vorzugsweise diesen Teil, den wir auch benötigen) importieren. Somit spart man Speicherplatz und bekommt ein schnelleres Pythonprogramm. Der Import nur von `date`.

```
from datetime import date
print(dir(date))
```

date vom Modul datetime importiert

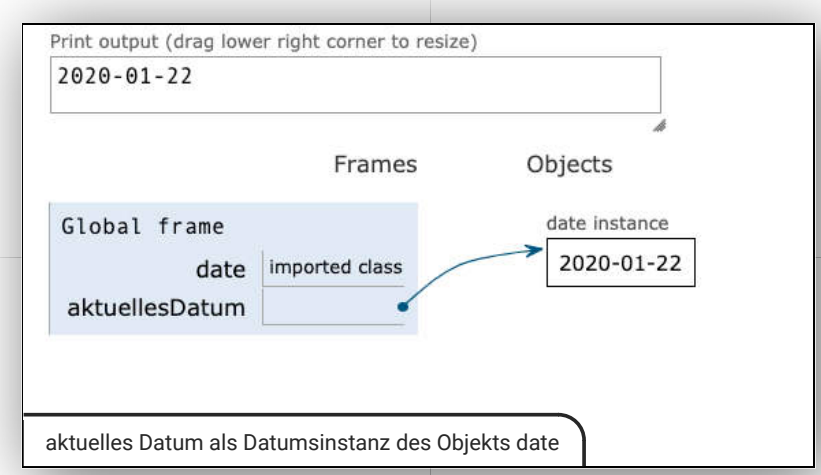
```
['__add__', '__class__', '__delattr__', '__dir__', '__doc__', '__eq__',
'__format__', '__ge__', '__getattribute__', '__gt__', '__hash__', '__init__',
'__init_subclass__', '__le__', '__lt__', '__ne__', '__new__', '__radd__',
'__reduce__', '__reduce_ex__', '__repr__', '__rsub__', '__setattr__',
'__sizeof__', '__str__', '__sub__', '__subclasshook__', 'ctime', 'day',
'fromordinal', 'fromtimestamp', 'isocalendar', 'isoformat', 'isoweekday',
'max', 'min', 'month', 'replace', 'resolution', 'strftime', 'timetuple', 'today',
'toordinal', 'weekday', 'year']
```

Beispiele für unsere Objekt datetime

Ausgabe des aktuellen Datums:

```
from datetime import date
aktuellesDatum = date.today()
print(aktuellesDatum)
```

Als Ergebnis erhält man das aktuelle Datum – als Beispiel erhält die Variable „2020-01-14“



aktuelles Datum als Datumsinstanz des Objekts date

Über `date` kann auch ein Datum „zugewiesen“ werden:

```
from datetime import date
weihnachten2020 = date(2020, 12, 24)
print(weihnachten2020)
```

Als Ausgabe erhalten wir dann:

viele Gegner fürs Spiel

Raspberry Pi – Python Interpreter
Impressum
Datenschutzerkärung
Sitemap

```
2020-12-24
```

Soll es anders ausgegeben werden, können wir dies nach Belieben einstellen:

Formatierte Ausgabe eines Datums

Soll das Datum entsprechend ausgegeben werden, kann über `strftime()` die gewünschten Einstellungen vorgenommen werden. Die Anweisung `strftime` steht für die Abkürzung „STRingFromTIME“ – wir konvertieren damit das datetime-Objekt als String für die Ausgabe. Dabei können wir noch angeben, welche Daten vom Datum (sprich Tag, Jahreszahl etc.) wir ausgeben lassen wollen. Im folgenden Beispiel wollen wir uns das Datum in typisch deutscher Schreibweise mit 4-stelliger Jahreszahl ausgeben lassen:

```
from datetime import date
weihnachten2020 = date(2020, 12, 24)
print(weihnachten2020.strftime("%d.%m.%Y"))
```

Dies ergibt als Ausgabe dann

```
24.12.2020
```

Um auch neben Datum auch die Uhrzeit nutzen zu können, importieren wir alles:

```
import datetime
weihnachten2020 = datetime.datetime(2020, 12, 4, 15, 30)
print(weihnachten2020.strftime("%H:%M:%S %d.%m.%Y"))
```

Hier stehen folgende Kürzel zur Verfügung:

Anweisung	Bedeutung	Beispiel
%a	Wochentag in Kurzschreibweise	Sun, Mon, …, Sat (en_US); So, Mo, …, Sa (de_DE)
%A	Wochentag ausgeschrieben	Sunday, Monday, …, Saturday (en_US); Sonntag, Montag, …, Samstag (de_DE)
%w	Wochentag als Nummer, dabei steht 0 für den Sonntag und 6 für Samstag	0, 1, …, 6
%d	Tag des Monates mit führender Null	01, 02, …, 31
%b	Monatsname abgekürzt.	Jan, Feb, …, Dec (en_US); Jan, Feb, …, Dez (de_DE)
%B	Monatsname ausgeschrieben	January, February, …, December (en_US); Januar, Februar, …, Dezember (de_DE)
%m	Monat als Zahl mit führender Null.	01, 02, …, 12
%y	Jahrszahl zweistellig	00, 01, …, 99

Anweisung	Bedeutung	Beispiel
%Y	Jahreszahl vierstellig	0001, 0002, …, 2013, 2014, …, 9998, 9999
%H	Stunde als 24 Stunden mit führender Null	00, 01, …, 23
%I	Stunde mit 12 Stunden mit führender Null	01, 02, …, 12
%p	Anzeige ob AM oder PM (at morning/past morning)	AM, PM (en_US); am, pm (de_DE)
%M	Minuten mit führender Null	00, 01, …, 59
%S	Sekunden mit führender Null	00, 01, …, 59
%f	Mikrosekunden mit führenden Nullen	000000, 000001, …, 999999
%z	UTC Offset	(empty), 0000, -0400, 1030, 063415, -030712.345216
%Z	Name der Zeitzone	(empty), UTC, EST, CST
%j	Tag des Jahres	001, 002, …, 366
%U	Wochennummer (wenn Sonntag der erste Tag in der Woche ist)	00, 01, …, 53
%W	Wochennummer (wenn Montag der erste Tag in der Woche ist)	00, 01, …, 53
%c	Komplette Ausgabe von Datum und Uhrzeit	Tue Aug 16 21:30:00 1988 (en_US); Di 16 Aug 21:30:00 1988 (de_DE)
%x	Komplettes Datum	08/16/88 (None); 08/16/1988 (en_US); 16.08.1988 (de_DE)
%X	Komplette Uhrzeit	21:30:00 (en_US); 21:30:00 (de_DE)
%%	Falls man doch mal das Prozentzeichen ausgeben möchte, einfach doppelt schreiben	%

Wochentag als Nummer

Die Nummer des Wochentags von einem gegebenen Datum wird ausgegeben über `. weekday()`. Wobei für Montag die 0 steht, Dienstag die 1 usw.

```
from datetime import date
aktuellesDatum = date.today()
print(aktuellesDatum.weekday())
```

möchte man lieber, dass der Wochentag mit 1 für Montag startet, hilft `.timetuple()`

```
from datetime import date
```

```
aktuellesDatum = date.today()
print(aktuellesDatum.isoweekday())
```

Und nun den Wochentag anhand einer Liste als Text ausgeben:

```
from datetime import date
aktuellesDatum = date.today()
wochentag_nr = aktuellesDatum.isoweekday()
print(wochentag_nr)

wochentage_kuerzel = ["So", "Mo", "Di", "Mi", "Do", "Fr", "Sa"]
print("aktueller Wochentag: ", wochentage_kuerzel[wochentag_nr])
```

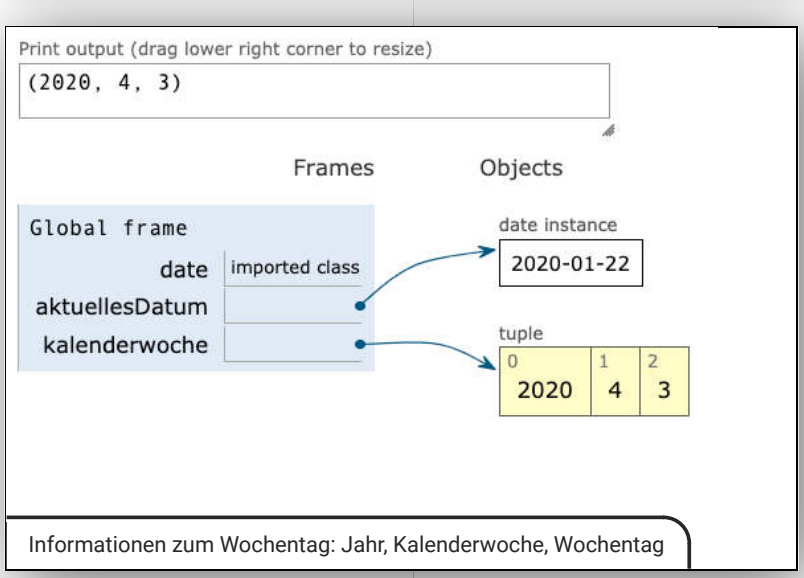
Kalenderwoche, Jahr und Wochentag als Nummer

Benötigt man die Kalenderwoche, erhält man über `.isocalendar()` ein Tuple mit dem Inhalt in der Reihenfolge (Jahr, Kalenderwoche, Wochentag im ISO-Format):

```
from datetime import date
aktuellesDatum = date.today()
print(aktuellesDatum.isocalendar())
kalenderwoche = aktuellesDatum.isocalendar()
```

Als Rückgabe erhalten wir:

```
(2020, 1, 22)
```



Im Datum etwas austauschen: replace

Möchte man etwas in einem bestehenden Datum austauschen, kann das sehr einfach über `replace()` geschehen. Dabei ist der Aufbau:

```
replace (year=self.year, month=self.month, day=self.day)¶
```

Nehmen wir an, wir möchten von Weihnachten (wahlweise geht auch der eigene Geburtstag) dieses Jahrs und nächsten Jahres den Wochentag und die Kalenderwoche erfahren. Also setzen wir das Datum von diesem Jahr und tauschen für die zweite Ausgabe das Jahr aus.

```
from datetime import date
weihnachten2020 = date(2020, 12, 24)
print(weihnachten2020.isocalendar())
weihnachten2021 = weihnachten2020.replace(year=2021)
print(weihnachten2021.isocalendar())
```

Wir erhalten als Ergebnis:

```
(2020, 52, 4)

(2021, 51, 5)
```

Sprich 2020 ist Weihnachten in der Kalenderwoche 52 an einem Donnerstag und 2021 an einem Freitag.

mit Tagen rechnen

Wir können sehr einfach über das Datum rechnen. Wir errechnen im folgenden Beispiel die vergangenen Tage seit der Geburt:

```
from datetime import date
heute = date.today()
print(heute)

heute_umf = heute.strftime("%m-%d-%Y. %d.%b.%Y ist ein %A am %d. Tag des %B.")
print(heute_umf)

# mit dem Datum lässt sich rechnen
geburtstag = date(1969,10,5)
heute = date.today()
alter = heute - geburtstag
print(alter.days, "Tage seit Geburt vergangen")
```

Standardbibliothek/Module Modul time