

python

Python Tutorial
- jetzt Python programmieren lernenSie befinden sich: Startseite » ausführbares Python-Programm erstellen » Standardbibliothek/Module » Modul datetime

suchen nach ...

suchen

Startseite

Python installieren

Python IDLE -
Lernumgebung

Python Befehle ausführen

Programm ausführen +
Command Line zum
debuggen

Hilfefunktionen

Online-Editor zum
schnellen Lernen

Text Editor Atom

Python Grundlagen

Ausgabebefehl print()

Variablen/Strings in
Python

Operatoren für Strings

Rechnen mit Zahlen

Listen – viele Inhalte
speichern

mit Listen arbeiten: Methoden

Variablen sind Listen?

Datentyp Dictionary

Tupel (Werte konstant
speichern)Mengen managen set u.
frozensetinput – Nutzerangaben
anfordern

Kommentare nutzen

ausführbares
Python-Programm
erstellen

if-Bedingung

Zufallszahlen über random

Übung:
Schmeichelprogramm

while-Schleife in Python

Übung: Zahlenraten-Spiel

for-Schleife in Python

Übung: Chatbot

range() – Listen erstellen

Übung: Lotto Simulator

Schleifenablauf
beeinflussen: break &
continueModul 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
<code>date</code>	Datumsklasse mit Jahr (year), Monat (month) und Tag (day). Voraussetzungen sind der gregorianische Kalender (und dass dieser gilt)
<code>time</code>	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.
<code>datetime</code>	Kombination aus <code>date</code> und <code>time</code> . Attribute und Voraussetzung entsprechen den einzelnen Klassen.

Bitte
unterstützen
Sie dieses
Projekt

Sie können dieses Projekt in verschiedenen Formen unterstützen - wir würden uns freuen und es würde uns für weitere Inhalte motivieren :).

Empfehlen Sie es weiter - wir freuen uns immer über Links und Facebook-Empfehlungen.

Sie können uns auch [eine Spende](#) über PayPal zukommen lassen.

Bestellen Sie Bücher über folgenden Link bei Amazon:
[Bücher über Python](#)

Vielen Dank für Ihre Unterstützung

Übung: Galgenmännchen

Funktionen in Python

Funktionen mit variabler
ParameteranzahlRückgabewert bei
Funktionen

eingebaute Funktionen

Module nutzen

Standardbibliothek/Module

Modul datetime

Modul time

Modul calendar

tkinter - GUI erstellen

Webbrowser über Python
nutzenURLLIB – Internetseiten
auslesen

Bibliothek requests

Selenium Browser fernsteuern

matplotlib - 2D Diagramme

Natural Language Toolkit
(NLTK)Wordcloud mit Python
erstellenTic-Tac-Toe Spiele-
Tutorial

Spielfeld erstellen

Spielzug durch Spieler

Hauptroutine des Spiels
erstellen

Spielfigur setzen

Spielende durch Gewinnen
oder Unentschieden

Spielzugkontrolle

Dumme KI integrieren

Turtle-Modul von
PythonHauptprogramm
__main__

Dateien auslesen

in Dateien schreiben

CSV Datei einlesen

Objektorientierte
Programmierung
Grundlagen (OOP)

Klassen in Python

Initialisieren der Klasse

Instanz einer Klasse
anlegen

Methoden in Klassen

Vererbung und Python

Attribute und Methoden in
Klassen überschreibenVariablen im Unterschied
zu Eigenschaften in
Klassen

Klasse	Beschreibung
timedelta	Zum Berechnen der Zeitdauer zwischen 2 Zeitpunkten. Rückgabe in Mikrosekunden.
timezone	Zeitzone 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))
```

Global frame	
MINYEAR	1
MAXYEAR	9999
timedelta	imported class
date	imported class
tzinfo	imported class
time	imported class
datetime	imported class
timezone	imported class

kompletter Import von datetime über *

```
['_add_', '__class__', '__delattr__', '__dir__', '__doc__', '__eq__',
'_format__', '__ge__', '__getattr__', '__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', 'strptime',
'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))
```

Eigenschaften vor Zugriff absichern

Klassen auslagern

Python und Datenbanken

SQLite: grundsätzliche Vorgehensweise

Daten speichern: INSERT INTO

Datenbank auslesen

Datensätze ändern: UPDATE

Löschen: DELETE FROM

SQL Grundlagen lernen

SQLite3 Shell

CSV-Datei in SQLite3

Datenbank importieren

Pygame Einführung

Pygame installieren Mac OS x

Formen zeichnen

Spiel Pong: Bewegung des Balls

Steuerung durch Spieler: Tastatur

Kollisionskontrolle – Aktion bei Schlägerberührung

Soundeffekte für unser Spiel und Hintergrundmusik

Koordinatensystem für Computerspiele

Spiel Breakout programmieren

Zeichen der Mauersteine

Spielfeld mit Mauersteinen nach Vorgabe füllen

Breakout-Ball im Spiel zeichnen und bewegen

Kollision zwischen Ball und Mauerstein

Kontrolle: Spielende durch gewinnen

Spielerfigur (Schläger) einbauen und bewegen

Kollisionskontrolle Ball und Schläger

Kontrolle: Spielende durch verlieren

Grundgerüst für Pygame

Bilder und Grafiken anzeigen

Grafiken rotieren

Grafiken skalieren

Spielerfigur animieren

Space Invaders war gestern

eigene Spielerfigur einbauen

Spielerfigur steuern

Bewegung der Spielerfigur begrenzen

Gegner einbauen

Gegner automatisch bewegen

Gegner abschießen

Berechnung, wann Geschoss trifft

Global frame

date imported class

date vom Modul datetime importiert

```
[ '_add_', '_class_', '_delattr_', '_dir_', '_doc_', '_eq_',
  '_format_', '_ge_', '_getattr_', '_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', 'strptime', '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“

Print output (drag lower right corner to resize)

2020-01-22

Frames

Objects

Global frame

date

imported class

aktuellesDatum

date instance

2020-01-22

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

Datenschutzerklärung

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 Monats 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	Jahreszahl 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])
```

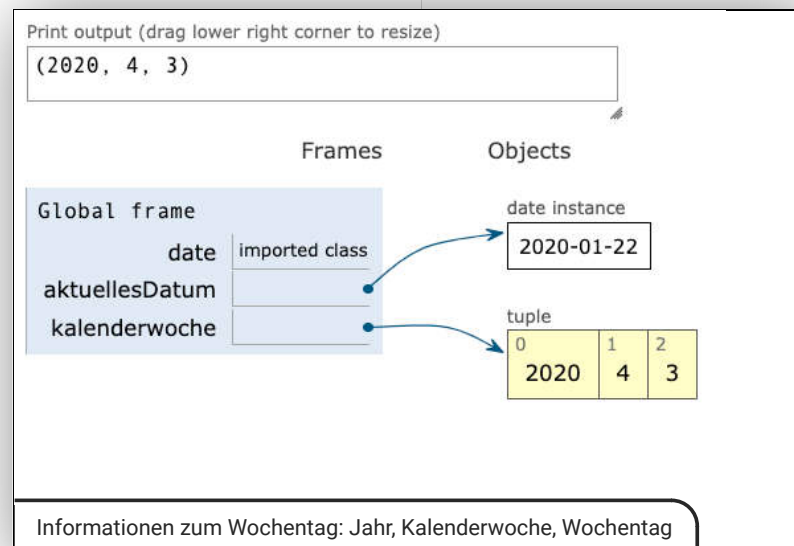
Kalenderwoche, Jahr und Wochentag als Nummer

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 Jahrs 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