LU14a - Python und die Zeit

Das speichern von Zeit und Datumswerten ist in vielen Applikationen notwendig. Python bietet uns mit den Modul `datetime` ein eifach zu benutzendes Modul dafür. Es hat mehrere Klassen, aber die am häufigsten verwendeten sind:

`datetime`: Diese Klasse stellt einen einzelnen Zeitpunkt dar. Sie hat Attribute für Jahr, Monat, Tag, Stunde, Minute, Sekunde und Mikrosekunde.

`timedelta`: Diese Klasse stellt eine Dauer dar, die zur Durchführung von Arithmetik mit `datetime`-Objekten verwendet werden kann.

Hier ist ein einfaches Beispiel für die Verwendung des `datetime`-Moduls, um das aktuelle Datum und die Uhrzeit zu ermitteln:

```python
from datetime import datetime

# Get the current date and time
now = datetime.now()
print(now)
```

Die Ausgabe sieht dann so aus:

```
2020-08-20 12:34:56.789012
```

Erstellen von datetime-Objekten

Mit dem Konstruktor `datetime()` können Sie ein `datetime`-Objekt für ein bestimmtes Datum und eine bestimmte Uhrzeit erstellen. Der `datetime()`-Konstruktor nimmt die folgenden Argumente entgegen:

- `year`: Das Jahr (vierstellig)
- `month`: Der Monat (1-12)
- `day`: Der Tag des Monats (1-31)
- `hour`: Die Stunde (0-23)
- `minute`: Die Minute (0-59)
- `second`: Die Sekunde (0-59)
- `microsecond`: Die Mikrosekunde (0-999999)

Hier ein Beispiel für die Erstellung eines datetime-Objekts für den `1. Januar 2020 um 12:00 Uhr`:

```
from datetime import datetime

dt = datetime(2020, 1, 1, 12, 0, 0)
print(dt)
```

Die Ausgabe sieht dann so aus:

```
2020-01-01 12:00:00
```

Sie können auch ein `datetime`-Objekt aus einer Zeichenkette mit der Funktion `datetime.strptime()` erstellen. Die Funktion `datetime.strptime()` nimmt zwei Argumente entgegen:

- eine Zeichenfolge, die das Datum und die Uhrzeit darstellt
- eine Formatzeichenfolge, die angibt, wie die Zeichenfolge formatiert werden soll. \\Die einzelnen Formatzeichen finden Sie unter https://docs.python.org/3/library/datetime.html#strftime-and-strptime-behavior beschrieben.

Das folgenden Beispiel zeigen, wie man ein `datetime`-Objekt aus einer Zeichenkette erstellt:

```
from datetime import datetime

# Parse a string into a datetime object
s = '2020-01-01 00:00:00'
dt = datetime.strptime(s, '%Y-%m-%d
%H:%M:%S')
print(dt)
```

Die Ausgabe sieht dann so aus:

```
2020-01-01 00:00:00
```

```
from datetime import datetime

# Parse a string into a datetime object
user_input = '12.03.2023 16:15'
date_time = datetime.strptime(user_input,
'%d.%m.%Y %H:%M')
print(date_time)

print(datetime.strftime(date_time, '%d.%m.%Y
%H:%M:%S'))
```

Die Ausgabe sieht dann so aus:

```
2023-03-12 16:15:00
12.03.2023 16:15:00
```

Mit der Funktion `strftime` können wir die Darstellung des Datums und der Uhrzeit bestimmen. Die Formatcodes sind die gleichen wie bei `strptime`.

Extrahieren von Datums- und Zeitinformationen

Sie können das Jahr, den Monat, den Tag, die Stunde, die Minute und die Sekunde aus einem `datetime`-Objekt mithilfe seiner Attribute extrahieren. Zum Beispiel:

```
from datetime import datetime

dt = datetime(2020, 1, 1, 12, 0, 0)

year = dt.year
month = dt.month
day = dt.day
hour = dt.hour
minute = dt.minute
second = dt.second
```

Datetime in formatierten String umwandeln

Verwenden von strftime

Sie können die Funktion `strftime` verwenden, um das Format anzugeben, in dem Sie das Datum und die Uhrzeit umwandeln möchten. Um zum Beispiel das Datum und die Uhrzeit in folgendem Format zu drucken: `TT.MM.JJJJ HH:MM:SS`, können Sie den folgenden Code verwenden:

```
dt = datetime(2020, 1, 1, 13, 23, 30)

formatted_time = dt.strftime('%d.%m.%Y
%H:%M:%S')
print(formatted_time)

01.01.2020 13:23:30
```

Hier ist die Liste einiger gängiger Formatcodes, die Sie mit `strftime` verwenden können:

- %Y: 4-Stellen Jahr (z.B. 2021)
- %m: 2-Stellen Monat (z.B. 01 for January)
- %d: 2-Stellen Tag des Monats (z.B. 09)
- %H: Stunde, im 24-Stunden Format (z.B. 13 for 1:00 PM)
- %M: Minute (z.B. 59)
- %S: Sekunde(z.B. 59)

Eine vollständige Liste der Formatcodes, die Sie mit `strftime` verwenden können, finden Sie in der Dokumentation: https://docs.python.org/3/library/datetime.html#strftime-strptime-behavior

Verwenden des f-String

f-Strings können auch verwendet werden, um Datums- und Zeitangaben zu formatieren. Hierbei wird das Datumsobjekt direkt in den String eingebettet, gefolgt von einem Doppelpunkt und dem gewünschten Formatierungscode.

```
from datetime import datetime

dt = datetime(2020, 1, 1, 13, 23, 30)
formatted_time = f'{dt:%d.%m.%Y %H:%M:%S}'
print(formatted_time)
# ODER
print(f'{dt:%d.%m.%Y %H:%M:%S}')

01.01.2020 13:23:30
```

Formatierung mit einem f-String

```
formatted_time = f"{dt:%d.%m.%Y %H:%M:%S}"
```

In dieser Zeile wird ein f-String verwendet, um das `datetime`-Objekt `dt` in einen formatierten String zu konvertieren. Innerhalb der geschweiften Klammern `{}` wird das `datetime`-Objekt `dt` platziert, gefolgt von einem Formatierungsmuster, das durch einen **Doppelpunkt** eingeleitet wird. `%d.%m.%Y %H:%M:%S` legt das Format des Datums und der Uhrzeit fest.

Erstellen von timedelta-Objekten

Die Klasse `timedelta` aus dem `datetime`-Modul stellt eine Dauer dar, die zur Durchführung von Berechnungen mit `datetime`-Objekten verwendet werden kann. Sie können `timedelta`-Objekte verwenden, um eine Dauer zu einem `datetime`-Objekt zu addieren oder von diesem zu subtrahieren, um ein neues datetime-Objekt zu erhalten.

Hier ein Beispiel für die Verwendung eines `timedelta`-Objekts, um einen Tag zu einem `datetime`-Objekt hinzuzufügen:

```
from datetime import datetime, timedelta

# Create a datetime object for January 1,
2020
dt = datetime(2020, 1, 1)

# Create a timedelta object representing one
day
one_day = timedelta(days=1)

# Add the timedelta to the datetime object
dt_plus_one_day = dt + one_day
print(dt_plus_one_day)
```

Die Ausgabe sieht dann so aus:

```
2020-01-02 00:00:00
```

Sie können ein `Timedelta`-Objekt erstellen, indem Sie den Konstruktor `timedelta()` aufrufen und ihm Argumente für die Dauer, die Sie darstellen möchten, übergeben. Der `timedelta()`-Konstruktor hat die folgenden Argumente:

- `days`: Die Anzahl der Tage in der Dauer.
- `hours`: Die Anzahl der Stunden in der Dauer.
- `minutes`: Die Anzahl der Minuten in der Dauer.
- `seconds`: Die Anzahl der Sekunden in der Dauer.
- `microseconds`: Die Anzahl der Mikrosekunden in der Zeitdauer.

Das folgende Beispiel zeigt, wie ein `Timedelta`-Objekt erstellt wird, das einen Tag, zwei Stunden und 30 Minuten darstellt:

```
from datetime import timedelta

duration = timedelta(days=1, hours=2,
minutes=30)
```

```
print(duration)
```

Die Ausgabe sieht dann so aus:

```
1 day, 2:30:00
```

Extrahieren von Timedelta-Informationen

Sie können Tage, Sekunden und Mikrosekunden aus einem `timedelta`-Objekt mithilfe seiner Attribute extrahieren. Zum Beispiel:

```
from datetime import timedelta

duration = timedelta(days=1, hours=2,
minutes=30)

day = duration.days
second = duration.seconds
microsenconds = duration.microseconds

# Minuten und Stunden müssen folglich
berechnet werden:
minutes = durations.seconds / 60
hours = durations.seconds / 3600
```

Rechnen mit timedelta-Objekten

Sie können mit `timedelta`-Objekten arithmetische Berechnungen durchführen, um Zeitdauern zu addieren oder zu subtrahieren. Zum Beispiel:

```
from datetime import timedelta

# Create two timedelta objects
duration1 = timedelta(days=1, hours=2,
minutes=30)
duration2 = timedelta(days=2, hours=1,
minutes=15)

# Add the durations
total_duration = duration1 + duration2
print(total_duration)
```

```
# Subtract the durations
difference = duration1 - duration2
print(difference)
```

Die Ausgabe sieht dann so aus:

```
3 days, 3:45:00
-1 day, 1:15:00
```

Sie können ein Timedelta-Objekt auch mit einem Skalar (einer Ganzzahl oder einem Float) multiplizieren oder dividieren. Zum Beispiel:

```
from datetime import timedelta

# Create two timedelta objects
td1 = timedelta(days=1, hours=2, minutes=30)
td2 = timedelta(days=2, hours=1, minutes=45)

# Multiply a timedelta by a scalar
td_scaled = td1 * 2
print(td_scaled)

# Divide a timedelta by a scalar
td_divided = td1 / 2
print(td_divided)
```

Die Ausgabe sieht dann so aus:

```
2 days, 5:00:00
12:45:00
```

Rechnen mit timedelta und datetime

Hinzufügen und Subtrahieren von Timedelta-Objekten zu Datetime-Objekten

Sie können ein `Timedelta`-Objekt zu einem `Datetime`-Objekt addieren oder davon subtrahieren, indem Sie die Operatoren + und - verwenden. Zum Beispiel:

```
from datetime import datetime, timedelta

# Create a datetime object for the current
date and time
```

```
now = datetime.now()

# Create a timedelta object representing a
duration of 1 day, 2 hours, and 30 minutes
td = timedelta(days=1, hours=2, minutes=30)

# Add the timedelta to the datetime object
later = now + td
print(later)

# Subtract the timedelta from the datetime
object
earlier = now - td
print(earlier)
```

Die Ausgabe sieht dann so aus:

```
2020-08-21 15:34:56.789012
2020-08-19 09:34:56.789012
```

Messung der Dauer zwischen zwei datetime-Objekten

Sie können ein `timedelta`-Objekt verwenden, um die Dauer zwischen zwei `datetime`-Objekten zu messen, indem Sie ein `datetime`-Objekt vom anderen subtrahieren. Zum Beispiel:

```
from datetime import datetime

# Create two datetime objects
dt1 = datetime(2020, 1, 1, 12, 0, 0)
dt2 = datetime(2020, 1, 2, 12, 0, 0)

# Subtract the two datetime objects to get a
timedelta object
duration = dt2 - dt1
print(duration)
```

Die Ausgabe sieht dann so aus:

```
1 day, 0:00:00
```

```
start = datetime(2020, 1, 1, 12, 15, 17)
end = datetime(2020, 1, 2, 12, 8, 20)
```

```python
# Subtract the two datetime objects to get a timedelta object
duration = end - start
duration_seconds = duration.total_seconds()
print(f'In Sekunden: {duration_seconds}')
print(f'In Minuten: {duration_seconds / 60}')
print(f'In Stunden: {duration_seconds / 60 / 60}')
print(f'In Tagen: {duration_seconds / 60 / 60 / 24}')
```

Die Ausgabe sieht dann so aus:

```
In Sekunden: 85983.0
In Minuten: 1433.05
In Stunden: 23.884166666666665
In Tagen: 0.995173611111111
```
