Digital Career Institute

Python Course – Dates in Python

Digital Career Institute

Goal of the Submodule

The goal of this submodule is to help the learners work with Dates and time in Python. By the end of this submodule, the learners should be able to understand

- Different ways of storing time in a variable
- How to work with timezones
- Working with time based values such as adding days, subtracting days, and much more.
- Alternative libraries that help computation with time.

Topics

- Introduction to the **`datetime`** module in Python
- Working with dates and strings
  - datetime.strftime()
  - datetime.strptime()
- Working with time
  - Current time where you are located using `pytz.timezone()`
- Working with timezones
  - How to easily work with timezones
  - Using the python-dateutil module (IANA tz database)

Digital Career Institute

Glossary

Term	Definition
IANA	Internet Assigned Numbers Authority (a global coordination of the Internet protocols, website domain related issues and associated numbers)
ISO 8601	International Organization for Standardization – 8601 (specifies how dates are written in order of most to least significant data.

Excurs to the complexity of date and time

Let’s watch this:

Introduction to Python datetime module

Digital Career Institute

Introduction to datetime

Digital Career Institute

- **datetime** is a fast implementation of the datetime type. You have so far seen data types like strings, integers and others. This is yet another data type that we use to handle time – past, present and future as well as associated time computations.

- To use this module, we first have to import it, and then invoke some special methods we shall look at over the next few sessions.

datetime's inner methods and classes

Digital Career Institute

method	Description
`datetime.datetime.today()`	This method is used to get the current local date and time of the day.
`datetime.date.today()`	This method is used to get the current local date (without the time)
`datetime.date.fromisoformat()`	Creates a datetime object using date represented as an ISO 8601 String.
`datetime.date()`	Create a datetime instance by providing keyword arguments such as year, month, day, hour minute and second.
`dir(datetime)`	See the list of methods you have access to 😉

Creating an instance of datetime (Date)

- Datetime has another module named datetime from which we can create an instance of time by providing year, month and date values as integers.

```
from datetime import datetime

datetime(2021, 2, 14)
```

Year Month Day

Creating an instance of datetime (Datetime)

- Previously we created only date, but we can provide time (hour, minute and seconds)

```
valentines_day = datetime(2021, 2, 14, 8, 20, 30)
```

Year Month Day Hour Minutes Seconds

Creating a datetime from a string

As you program, some input you receive from users comes in a string format. We should be able to convert that string to a datetime object for further management.

```
birth_date = "2005-01-01"
# User input
```

```
date.fromisoformat(birth_date)
```

```
datetime.date(2005, 1, 1)
```

Creating a datetime using .date()

- If you know the year, month and day, you can create a basic datetime instance as follows:


```
date(year=2001, month=10, day=10)
```

Creating a datetime from a string

- Sometimes users may have a different style of handling dates,
- In the US, dates are usually written "month, day and Year" - 01-02-2005 – means January 2nd, 2005.
- In Germany, the dates are in following style: "Day, Month and Year", so the date would be 1st February, 2005.

Creating a datetime from a string

```
usa_meeting = "January 1, 2005"
```

```
datetime.strptime(usa_meeting, "%B %d, %Y")
```

Creating a datetime from a string

```
german_meeting = "1 January, 2005"
```

```
datetime.strptime(german_meeting, "%d %B, %Y")
```

Converting datetime instance to a string

- A Python program can be used to process time that was previously stored as a datetime object which can be harder to read for a human, so we can make a lot more friendly by using the **datetime.strftime()** method.
- We can call this formating time.


```
datetime.now().strftime("%H:%M:%S")
```

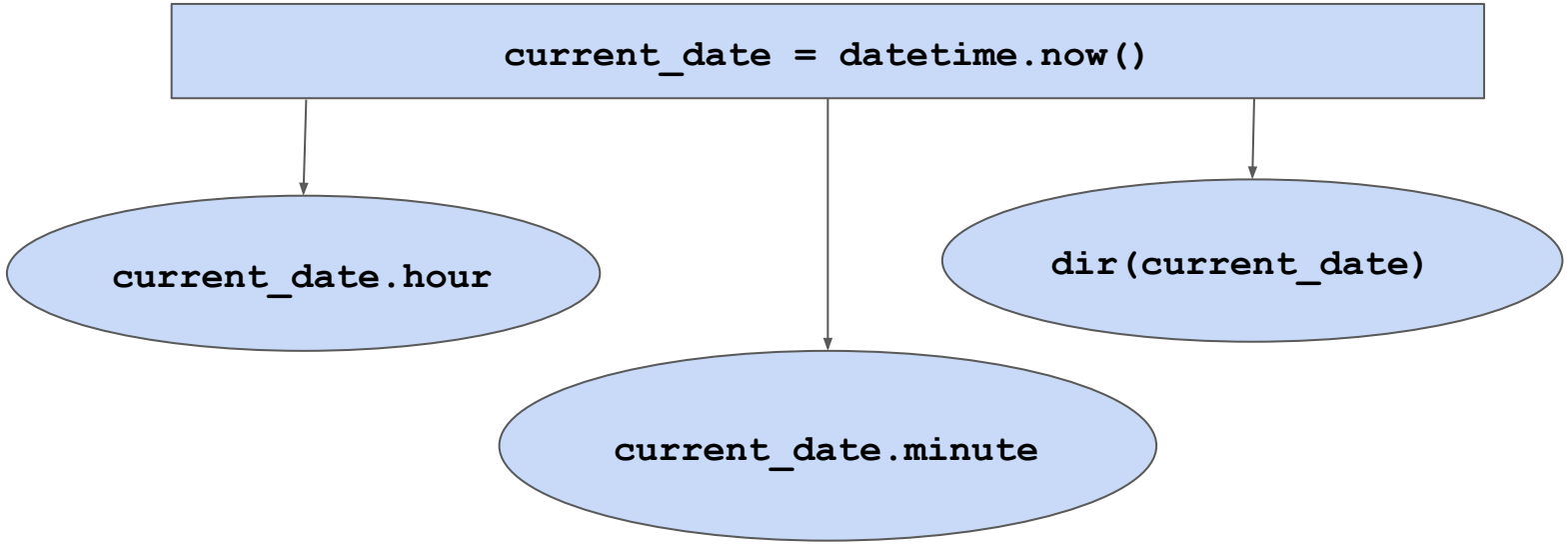
A few codes you can pass to either strftime() or strptime()

Directive	Description
`%a`	Abbreviated week day name such as Sun, Mon, Thur etc.
`%A`	Full weekday name (Sunday, Monday, Thursday, etc.)
`%w`	Weekday as an integer between 0 and 6.

An exhaustive list can be found in this reference:
https://strftime.org/

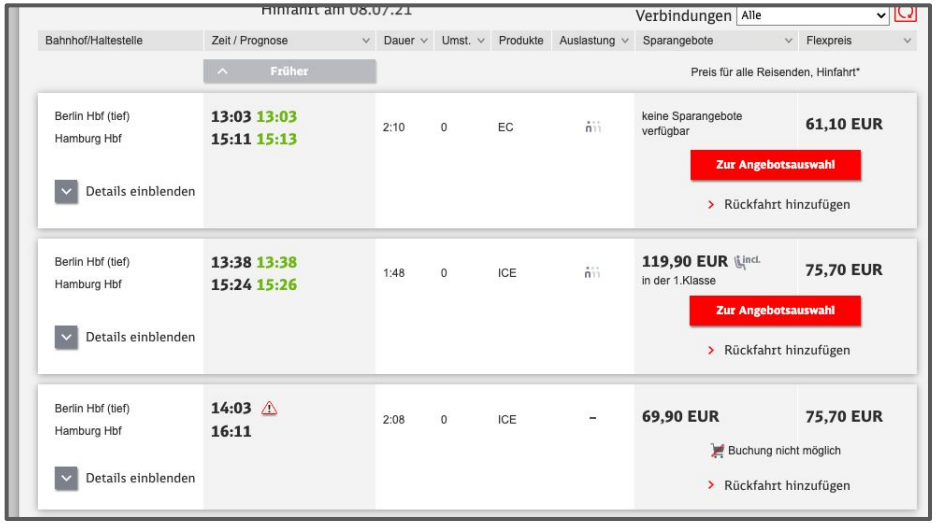
datetime instance properties/methods

- Your datetime has some helpful properties you can access such as year, day and month.
- To see other methods that exist, use the dir() function



Comparing time

- On booking websites such as the Deutsche Bahn, you may have been able to sort ticketing options by when a train departs – this operation involves comparing time and dates. We'll look at some examples using **datetime.timedelta**



Complications with Time management

Digital Career Institute

- Dealing with changes in time can be a complicated process
- A lot like adding numbers together, 1 + 1, we can do something similar with time
- A convenient method to help is **`timedelta`**

Adding time

- You can manipulate datetime objects with timedelta quite easily.
- Lets add a few more days to our vacation.

At the core of the lesson

Lessons Learned:

- We know how to create instances of datetime
- We can manipulate time
- We know how to use date, datetime and timedelta
- We can format time from a string with strftime()
- We can create time from a string provided by a user using strptime()