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.

Digital Career Institute

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

- **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

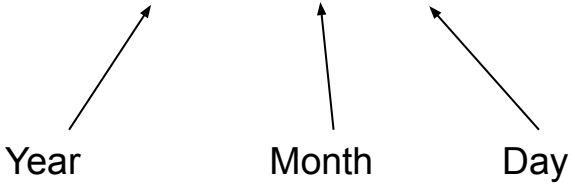
method	Description
`datetime.datetime.today()`	This method is used to get the current local date and time of the day.
`datetime.date.today()`	This method is used to get the current local date (without the time)
`datetime.date.fromisoformat()`	Creates a datetime object using date represented as an ISO 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.

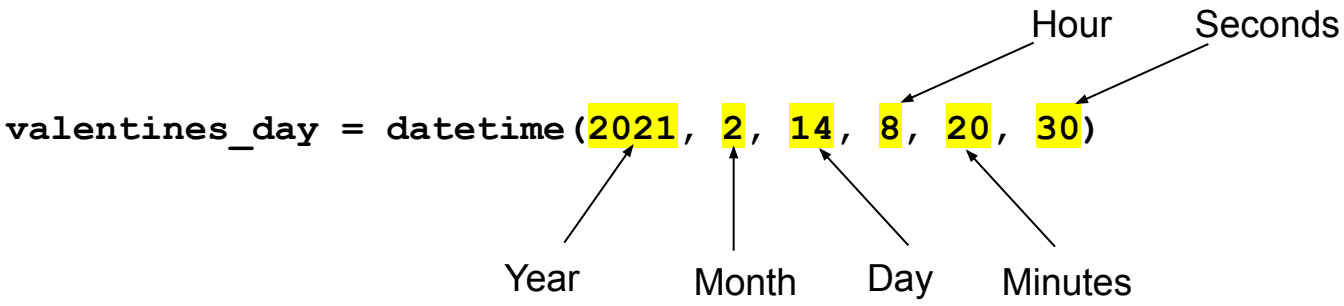
```
from datetime import datetime

datetime(2021, 2, 14)
```



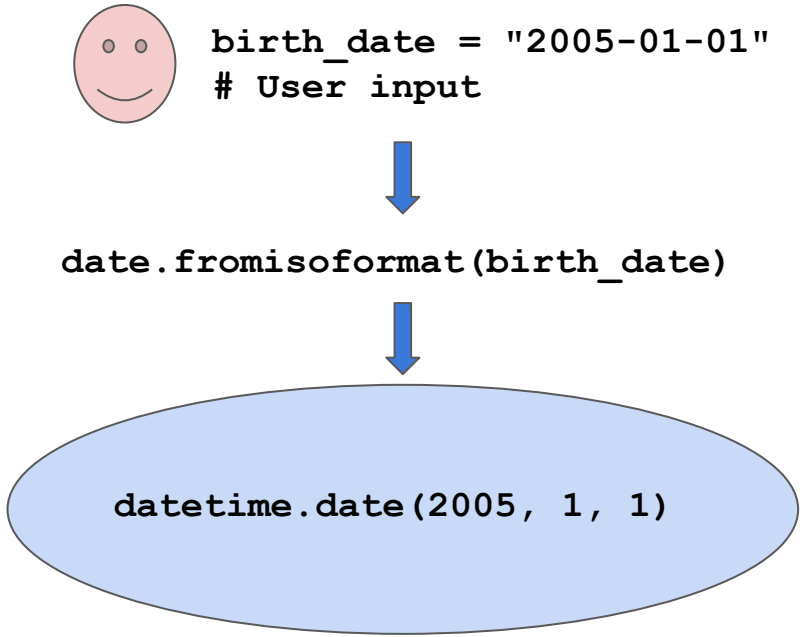
Creating an instance of datetime (Datetime)

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



Creating an 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.



Creating an 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

Digital Career Institute

- 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 an datetime from a string

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

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

Creating an 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()

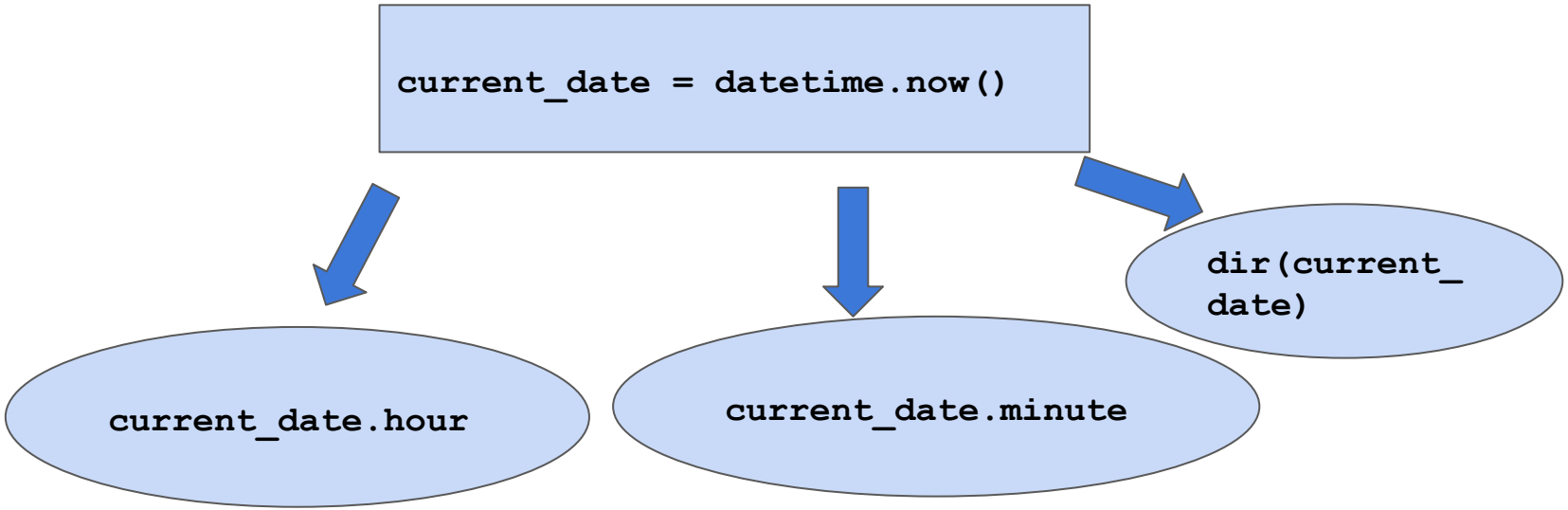
Digital Career Institute

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/

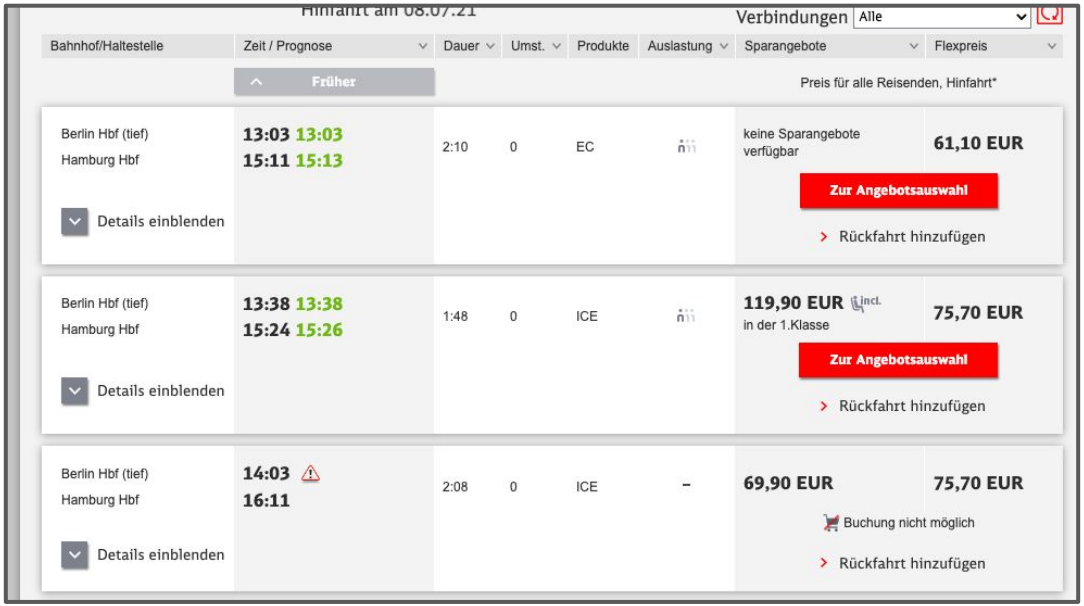
datetime instance properties/methods

- 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

Digital Career Institute

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()

Calendar

Digital Career Institute

calendar module

Digital Career Institute

- This inbuilt module handles operations that are related to the calendar. The calendars are usually defined based on the current Gregorian calendar.

- The calendars have Monday as the first day of the week and Sunday as the last day of the week.

calendar module - usage

```
print(calendar.month(2021, 8))
```

```
    August 2021
Mo Tu We Th Fr Sa Su
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
```

Is current year a leap one?

Digital Career Institute

- A leap year occurs once every 4 years
- Let’s experiment with some of the calendar methods

```
calendar.isleap(2021)
```

Some calendar methods – TextCalendar class

method	Description
`calendar.setfirstweekday()`	This method is used to set the start number of week
`calendar.firstweekday()`	This method is returns the first day of the week. Default value is 0 or Monday.
`calendar.isleap()`	This method returns a boolean (true or false) depending on whether the year supplied to it as an argument is a leap year or not.
`References`	More references can be found in here: https://docs.python.org/3/library/calendar.html

At the core of the lesson

Lessons Learned:

- Working with the calendar module
- You can print out a calendar

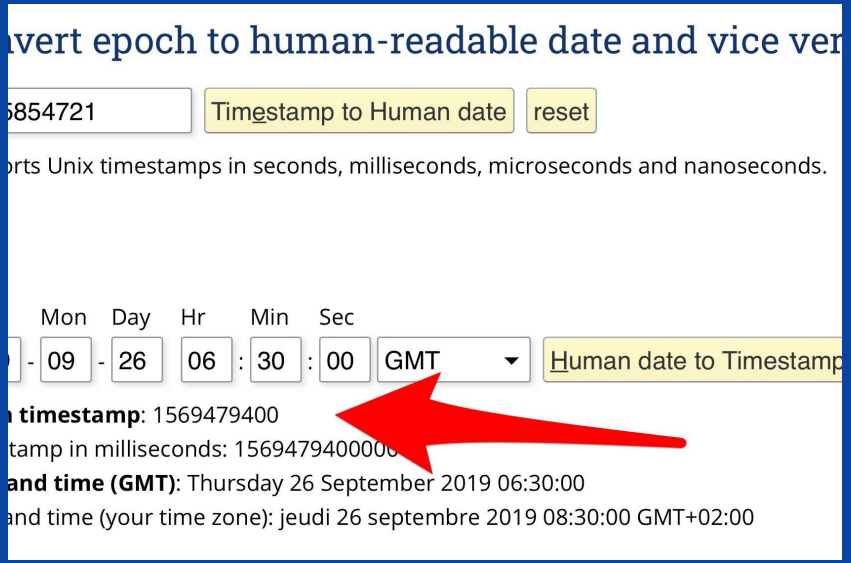
Digital Career Institute

Self Study

How to format dates in a human readable way

Digital Career Institute

Timestamps



Digital Career Institute

Timestamps

- A unix timestamp is a way to track time in seconds
- Counting of time starts from January 1st, 1970 at UTC
- The Timestamp is there for the time between January 1970 and the current date you look at

Timestamps

- 1 Hour = 3600 seconds
- 1 Day = 86400 Seconds
- 1 Week = 604800 Seconds
- 1 Month (30.44 days) = 2629743 Seconds
- 1 Year (365.24) = 31556926 Seconds

Bugs and Time - Y2K

- Did you know that on January 19, 2038, the UNIX timestamp will stop working (32 bit overflow) on older systems

- Another trivial time situation happened with Y2K - computers had problems dealing with dates after December 31, 1999

- https://www.nationalgeographic.org/encyclopedia/Y2K-bug

At the core of the lesson

Lessons Learned:

- Dealing with Timestamps
- Numbers representing Timestamps can be stored easily by machines making it easier to manage time by representing all timezones at the same time

Summer/Winter and Daylight savings

Digital Career Institute

Winter Time and Summer Time

- It begins at 1.00 am GMT on the last Sunday in October every year
- Clocks are reversed by 1 hour
- Winter time ends at 1.00am on the last Sunday in March the following year - this signals Summer has started

Time Zones

Digital Career Institute

Working with timezones

Digital Career Institute

- By default datetime uses your computer's timezone. For your programs to work globally, a universal point of reference should be used when dealing with time. For example GMT, UTC or your local time.

- These references can then be used to help users from other countries see time from their perspective.

- You can manipulate time on your own but it can be harder to keep track of different time zones and rules around the world – including daylight savings.

Some Common time zones

TIME ZONE	Region
GMT	Greenwich Mean Time (UK area as focal point)
CEST	Central European Summer Time (1hr ahead of GMT)
UTC	Coordinated Universal Time - a successor to the GMT time zone.
EST	Eastern Standard Time (New York area for example)
PST	Pacific Standard Time (California or western America regions)

For a detailed list: https://www.timeanddate.com/time/zones/

Working with Time zones in Real World

Digital Career Institute

- Many software applications are designed with location in mind
- For the most part, you should not manage timezones by yourself.
- Stick to a reference such as GMT or UTC when saving your data.
- Derive the actual time by calling special datetime methods

At the core of the lesson

Digital Career Institute

Lessons Learned:

- You have learned how to work with dates and timezones
- You can manipulate dates while maintaining a different timezone

Documentation

Documentation

1. https://docs.python.org/3/library/calendar.html
2. https://dateutil.readthedocs.io/en/stable/index.html

Contact Details
DCI Digital Career Institute gGmbH

Digital Career Institute