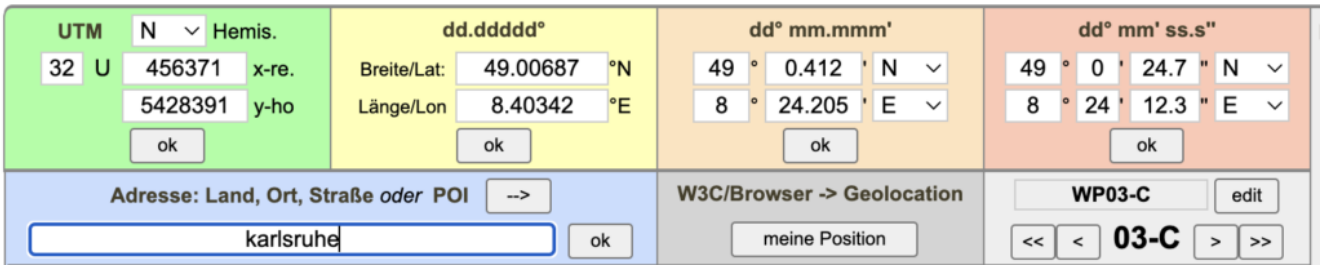
Berechne den nächsten Überflug der ISS

In diesem Tutorial erfährst du, wie den nächsten Überflug der ISS mit Python berechnest. Auf Pollux Labs gibt es bereits zwei Projekte, die sich mit der International Space Station auseinandersetzen: Ein Raspberry Pi Projekt, mit dem du die Flugbahn der ISS auf einer Karte darstellst und ein eine Erweiterung der LEGO ISS, die bei der Passage der Raumstation aufleuchtet.

Letzteres Projekt basierte auf einer API, die die Überflugsdaten für einen beliebigen Standort bereitstellen konnte. Leider wurde der Betrieb dieser API eingestellt, **weswegen du diese Berechnung am besten selbst vornimmst – wie, lernst du hier.**

Die Koordinaten für deinen Standort ermitteln

Um zu berechnen, wann die ISS über deinem Kopf fliegt (oder zumindest theoretisch sichtbar ist), benötigst du zunächst die Koordinaten für deinen Standort. Hierfür eignet sich z.B. die Website geoplaner.de – trage hier deinen Standort ein und du erhältst umgehend die zugehörigen Koordinaten (zum Herauskopieren eignet sich der gelbe Kasten):



Im obigen Fall wären das für Karlsruhe circa 49° nördliche Breite und 8,4° östliche Länge. Diese Zahlen benötigst du später im Python-Script für die Berechnung des Überflugs bzw. der Sichtbarkeit der ISS.

Die benötigte Python-Bibliothek

Um den Überflug der ISS zu ermitteln, benötigst du die Bibliothek **ephem**, die im Terminal wie folgt installierst:

```
pip install ephem
```

Diese Bibliothek bietet umfangreiche Funktionen für astronomische Beobachtungen – du kannst damit neben verschiedenen Himmelskörpern wie Planeten, Asteroiden und Kometen auch die Flugbahnen von Satelliten berechnen, dazu gehört im weitesten Sinne auch die ISS. Mehr über dieses großartige Projekt erfährst du auf der offiziellen Webseite.

Die aktuellen TLE-Daten ermitteln

Damit die Bibliothek **ephem** den Überflug der ISS berechnen kann, benötigt sie neben deinem Standort auch aktuelle Daten der Raumstation. Hierfür kommt ein sogenanntes **Two-Line Element zum Einsatz, kurz TLE.** Hierbei handelt es sich um ein standardisiertes Format für eine Reihe von Daten eines Flugkörpers in der Umlaufbahn der Erde. Mehr über TLEs erfährst du auf Wikipedia (Englisch).

Um das aktuelle TLE der ISS zu erhalten, eignen sich die beiden Webseiten von ARISS und Celestrak. Auf beiden Seiten findest du die aktuellen Daten für das ISS-Modul Sarja (englisch Zarya). Hier ein Beispiel eines TLE:

```
ISS (ZARYA)
1 25544U 98067A   24238.20528372  .00032948  00000-0  58135-3
0  9999
2  25544   51.6406  333.3912  0006392  271.4643  173.4301
15.50058100469255
```

Die beiden Zeilen kannst du von der Webseite kopieren und gleich in dein Script eintragen. **Ein wichtiger Hinweis:** Das TLE für die Raumstation wird regelmäßig aktualisiert und an die tatsächlichen Begebenheiten der ISS angepasst. Damit deine Berechnung zuverlässige Daten liefert, kopiere am besten die aktuellen Werte in dein Script.

Den nächsten ISS Überflug berechnen

Nun zum Kern des Vorhabens: mit dem folgenden Python-Script berechnest du den nächsten Überflug der ISS:

```
import ephem

# Koordinaten deines Standorts
latitude = '49'
longitude = '8.4'

# Aktuelles TLE
tle1 = "1 25544U 98067A   24238.20528372  .00032948  00000-0
58135-3 0  9999"
tle2 = "2 25544  51.6406 333.3912 0006392 271.4643 173.4301
15.50058100469255"


# Instanz für deinen Standort erzeugen
observer = ephem.Observer()
observer.lat = latitude
```

```
observer.lon = longitude

# Instanz für die ISS erzeugen
iss = ephem.readtle("ISS (ZARYA)", tle1, tle2)

# Höhe deines Standorts (optional) und lokale Zeit einstellen
observer.elevation = 0  # Elevation in meters (optional)
observer.date = ephem.localtime(ephem.now())  # Use current
date and time

print("Aktuelle Zeit (lokal):", observer.date)

# Nächsten Überflug berechnen
next_pass = observer.next_pass(iss)

# Daten ausgeben
print("Next pass of the ISS:")
print(f"Date     and     Time     (lokale     Zeit):
{ephem.localtime(next_pass[0])}")
print(f"Rise (lokale Zeit): {ephem.localtime(next_pass[1])}")
print(f"Maximum     Elevation     (lokale     Zeit):
{ephem.localtime(next_pass[2])}")
print(f"Set (lokale Zeit): {ephem.localtime(next_pass[3])}")
print(f"Max Elevation (degrees): {next_pass[4]}")
```

So funktioniert das Script

Nachdem du die Bibliothek **ephem** eingebunden hast, hinterlegst du die Koordinaten deines Standorts. Anschließend folgen die Werte des aktuellen TLE. Als nächstes erzeugst du eine Instanz **observer**, die die Koordinaten deines Standorts enthält sowie eine Instanz für die ISS mit den Werten des TLE.

Zusätzlich zu den Koordinaten kannst du auch die Höhe deines Standorts eintragen. Liegt dieser z.B. 150 Meter über dem Meeresspiegel, trage hier einfach eine 150 ein. In der Zeile darunter stellst du die aktuelle Uhrzeit deiner Zeitzone ein.

Mit all diesen Daten rufst du die Funktion

observer.next_pass(iss) auf und speicherst das Ergebnis in der Variablen **next_pass**. Die Berechnung berücksichtigt also deine Standortdaten in der Instanz **observer** sowie die TLE-Werte in der Insanz **iss**.

Zuletzt musst du nur noch die Ergebnisse ausgeben. Hier gibt es allerdings ein kleines Problem: Eigentlich erhältst du nicht nur einen Zeitpunkt, an dem die ISS am Himmel sichtbar ist, sondern auch den Zeitpunkt, an dem sie über den Horizont steigt und wieder dahinter verschwindet. Außerdem berechnet **ephem** auch den maximalen Winkel der ISS über dem Horizont. Leider gibt es – zumindest bei meinen Tests – nur eine sinnvollen Wert: den Zeitpunkt ihrer maximalen Höhe:

Next pass of the ISS:
Date and Time (lokale Zeit): 2024-08-26 02:50:05.657860
Rise (lokale Zeit): 1900-01-03 17:40:39.115906
Maximum Elevation (lokale Zeit): 2024-08-26 02:54:23.887056
Set (lokale Zeit): 1899-12-31 16:40:30.704498
Max Elevation (degrees): 2024/8/26 00:58:43

Wie du in der obigen Ausgabe siehst, sind die Zeitangaben für **Rise** und **Set** (also Aufgang und Untergang) unsinnig. Ebenso der Wert für **Max Elevation**. Einen guten (und für dein Projekt möglicherweise ausreichenden) Wert findest du jedoch hinter **Maximum Elevation (lokale Zeit)** – diese Zeitangabe sagt dir, wann der nächste Überflug der ISS an deinem Standort zu sehen ist.

Falls du einen Anhaltspunkt oder eine Lösung für diese Fehler hast, schreib mir gerne an info@polluxlabs.net – ich bin für jeden Hinweis dankbar.

Überprüfung der Ergebnisse

Falls du der Berechnung noch nicht so recht Glauben schenkst, kannst du z.B. eine App auf deinem Smartphone hinzuziehen und dort nachschauen, wo sich die Raumstation gerade befindet und

wann du mit dem nächsten Überflug der ISS rechnen kannst. Ich selbst verwende hierfür die kostenlose **App SkyView Lite**.

Was kannst du damit bauen?

Falls du „nur“ den Zeitpunkt für den nächsten Überflug der ISS benötigst, damit du sie am Abend- oder Morgenhimmel beobachten kannst, reicht dir dieses Script vermutlich schon. Falls du jedoch etwas bauen möchtest, könnte die oben verlinkte LEGO ISS interessant für dich sein. Dort habe ich zwar einen ESP8266 verwendet, darauf läuft jedoch kein Python. **Stattdessen könntest du einen Raspberry Pi Zero verwenden,** der ähnlich klein ist und sich in der aufgebauten Raumstation gut unterbringen lässt.

So könnte das beleuchtete Modell aussehen: