Skyfield - Elegante Astronomie für Python

Dietmar Thaler - GLT24
https://glt.foehnwall.at/glt24.html

6. April 2024.

Einleitendes

Motivation

Python Bibliothek für astronomische Berechnungen

Zur Anwendung in eigenen Applikationen ..

- Sonnenauf- und -untergangszeiten
- Dämmerungszeiten
- Mondphasen, Mondauf- und -untergangszeiten
- Jahreszeitenwechsel
- Position und Bahnen von astronom. Objekten
- u.a. ..

Python Pakete für Astronomie

Mit unterschiedlichem Fokus:

- **PySolar** - Beschränkung auf die Sonne inkl. Sonnenphysik: https://pysolar.readthedocs.io/en/latest
- **AstroPy** - für professionelle Astronomie: https://www.astropy.org
- **PyEphem** von **Brandon Rhodes** - kein reines Python: https://rhodesmill.org/pyephem
  Kurzübersicht zu PyEphem auch unter https://glt.foehnwall.at/glt16.html
  Wird nicht mehr entwickelt, nur mehr gewartet.

Empfohlene Alternative zu PyEphem: https://rhodesmill.org/skyfield

- Autor **Brandon Rhodes** https://rhodesmill.org/brandon/
- reiner Python-Code für Python**2** $\geq$ 2.6 sowie Python**3** $\geq$ 3.3
- einzige binäre Abhängigkeit über **Numpy** (z.B. die BLAS-Routinen)
- Anspruch auf eine **Genauigkeit von 0.5 Tausendstel(!) Bogensekunden** (verglichen mit dem Astronomical Almanac des United States Naval Observatory)

Installation

Siehe https://rhodesmill.org/skyfield/installation.html

z.B. (derzeit Skyfield Version 1.48 - 7. Feb. 2024):

1. Falls noch nicht erledigt, **pip3** und **Numpy** installieren:
   `sudo apt install python3-pip, python3-numpy`
2. Lokale Installation (Abhängigkeiten werden i.d.R. mit installiert):
   `pip3 install skyfield`

Dokumentation

Hauptseite: https://rhodesmill.org/skyfield/

- Inhaltsverzeichnis: https://rhodesmill.org/skyfield/toc.html
- API: https://rhodesmill.org/skyfield/api.html
- Bug-Reports: https://github.com/skyfielders/python-skyfield/issues

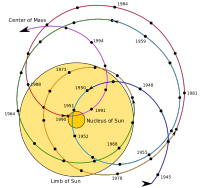
Astronomische Positionen und Koordinaten

International Celestial Reference System (ICRS)

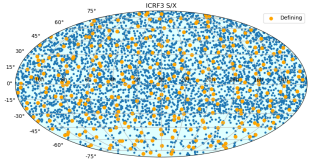
Fixierung eines **Inertialsystems**

International Celestial Reference System (ICRS): weltweites Bezugssystem von kartesischen Koordinaten des Sonnensystems

- **Barycentric Celestial Reference System** BCRS: Ursprung im Schwerpunkt des Sonnensystems
- **International Celestial Reference Frame** ICRF.
  - seit 2019 durch $>$ 4500 extragalaktische Referenzpunkte definiert
  - **Koordinatenachsen** entlang der Rotationsachse der Erde und den Punkten der Tag- und Nachtgleiche für die Epoche J2000.0 (bzw. J2015.0)

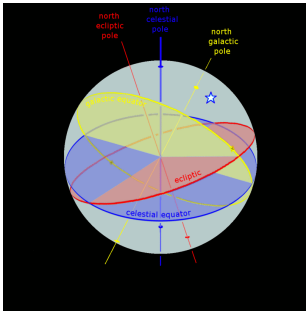


https://commons.wikimedia.org/wiki/File:Solar_system_barycenter.svg



Quelle: SYRTE (Observatoire de Paris), United States Naval Observatory http://hpiers.obspm.fr/icrs-pc/newwww/icrf/icrf3sx.png

Verschiedene Koordinaten-Darstellungen



Quelle: nichtanimiertes Teilbild aus: https://en.wikipedia.org/wiki/Celestial_coordinate_system#/media/File:Ecliptic_equator_galactic_anim.gif

- **Äquatorialsystem** (Rektaszension α, Deklination δ)
- **Ekliptikalsystem** (ekliptikale Länge λ, ekliptikale Breite β)

Koordinatenursprung

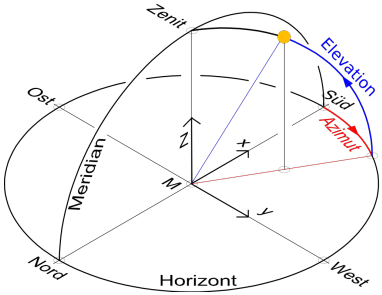
- **Baryzentrisch** bezügl. des Sonnensystems (ICRS)
- **Heliozentrisch** um den Sonnenmittelpunkt
- **Geozentrisch** um den Erdmittelpunkt

- **fest** oder **mitrotierend**
- alle Transformationen unter Berücksichtigung der **allgemeinen Relativitätstheorie**

Topozentrisches Bezugssystem - Horizontalsystem

Z.B. an der Erdoberfläche:

- Elevation bzw. Höhe (0 . . . 90 Grad)
- Azimut (0 . . . 360 Grad)
  - Astronomie beginnend von Süden über W nach E und N
  - Navigation (und in Skyfield) von Norden über E nach S und W
- Winkeleinheiten:
  - Dezimalgrad
  - Grad-Minuten-Sekunden
  - Stundenwinkel (quasi als “Uhrzeit”)



Wikipedia, S.Wetzel (2009):
https://de.wikipedia.org/wiki/Datei:HorSys.svg

Zeitsysteme

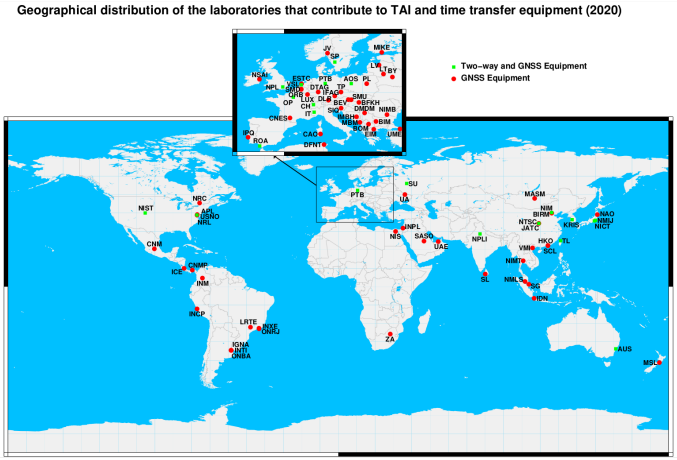
Definition der Sekunde

Die **Atomsekunde** (SI-Sekunde)[a] :

- Eine Sekunde ist das 9 192 631 770-fache der Periodendauer der Strahlung, die dem Übergang zwischen den beiden Hyperfeinstrukturniveaus des Grundzustandes von Atomen des Nuklids Cs-133 entspricht.
- Der **SI-Tag** hat 86400 Atomsekunden (SI-Sekunden)

[a] Historisch die Sonnensekunde: der $\frac{1}{24 \times 60 \times 60} = \frac{1}{86400}$ Teil eines mittleren Sonnentages.

Internationale Atomzeit (TAI) und Terrestrische Zeit (TT)



TAI: Gewichtetes Mittel von derzeit über 600 über die Erde verteilten Atomuhren.

Quelle: Buro International des Poid et Mesures (BIMP)

Internationale Atomzeit (TAI) und Terrestrische Zeit (TT)

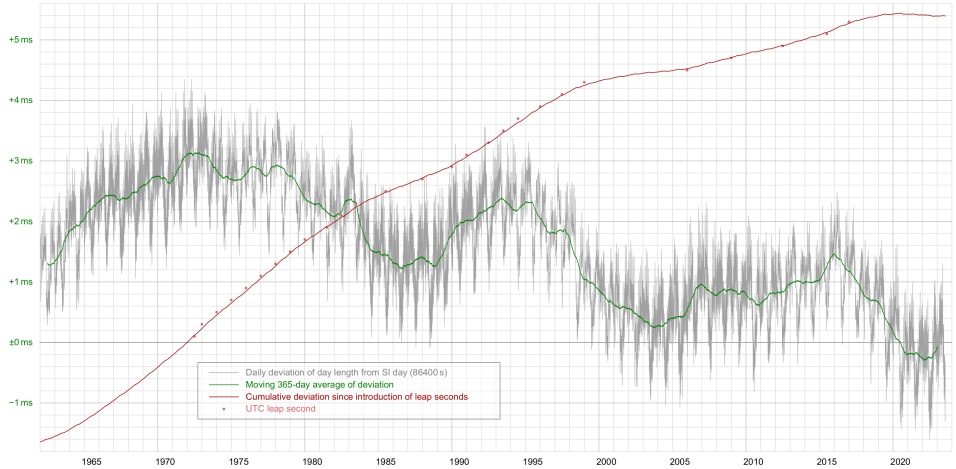
- **TAI**: Gewichtetes Mittel von derzeit über 600 über die Erde verteilten Atomuhren.
- **TT = TAI+32.184 s**
- Die gemessene Atmomzeit TAI ist (bis auf die konstante Differenz [1]) eine Realisierung der Terrestrischen Zeit TT.
- **TT** ist der Zeitparameter in den Bewegungsgleichungen.

[1] Die Differenz hat historische Gründe

Universal Time UT

- **UT** (genauer **UT1**) orientiert sich an der Rotation der Erde
- UT-Sekunde verwandt mit der alten Definition der Sekunde als Bruchteil des mittleren Tages.
- **Rotationsperiode** der Erde **variabel**
- Die UT-Sekunde ist **unregelmäßig** und nicht ident mit der SI-Sekunde
- UT ist ein Maß für den **Phasenwinkel der Erdrotation**
- In Greenwich ist im Mittel die **Sonne zu Mittag im Süden**

Abweichung der Tageslänge vom SI-Tag (TT/TAI-Tag)



https://en.wikipedia.org/wiki/File:Deviation_of_day_length_from_SI_day.svg

Weltzeit UTC - Schaltsekunden

- **TT/TAI** und **UT** laufen immer weiter auseinander
- Lösung: **UTC** (Universal Time Coordinated)
  - Einheit "Atomsekunden" wie bei TT/TAI
  - Wenn **UTC** der **UT** davon läuft ($|UT - UTC| \geq 0.9s$) –> **Schaltsekunde**
    z.B.:
    31.12.2016 23:59:59
    31.12.2016 23:59:60
    01.01.2017 00:00:00

UTC ist eine Zeitmaß, welches die Vorteile einer mittleren "Sonnenuhr" mit den Vorteilen einer gleichmäßigen Zeiteinheit verbindet.

Julianisches Datum - Julian Date (JD)

Eine Zeitrechnung für die Terrstrische Zeit **TT** (oder **UT**)

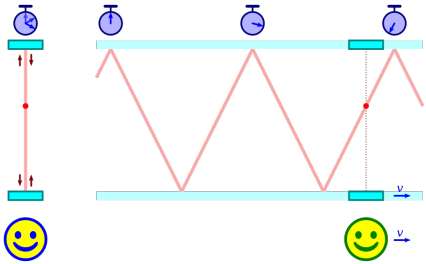
- **JD 0.0**: **Mittag am 1. Jan. 4713 v.Chr.**[2] gemäß dem **proleptischen Julianischen Kalender** (1 Jahr = 365.25 Tage)[3]
- Gezählt wird in ganzen Tagen zu 86400 Sekunden
- Weil **TT** und **UT** verschieden sind, muss man angeben, auf welche Zeitskale es sich bezieht. Heute in der Astronomie üblich: **TT**
- z.B.: 2020-02-29, 11:58:50.816 UTC = 2 458 909.0000000000 JD(TT)

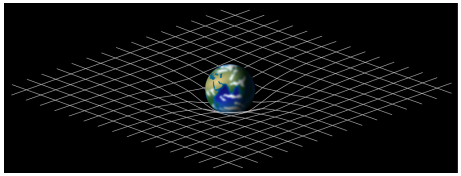
[2] geht zurück auf *Joseph Scaliger*, franz.-niederländ. Astronom, 1540-1609

[3] 24. November 4714 v.Chr. gemäß dem proleptischen Gregorianischen Kalender (1 Jahr = 365.2425 Tage)

Relativistisches Raum-Zeitgefüge

commons.wikipedia.org .. Zeitdilitation-Lichtuhr





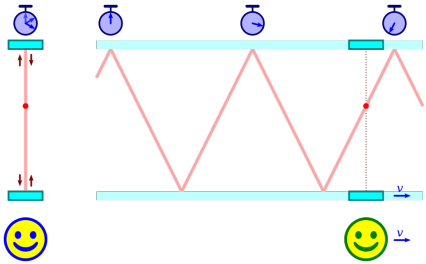
commons.wikipedia.org .. Spacetime lattice

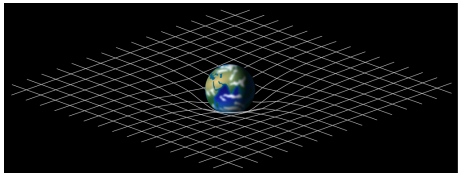
Astronom. Berechnungen müssen berücksichtigen:

- **bewegte Uhren** gehen langsamer
- **schwere Uhren** gehen langsamer
- Licht eines **entfernten Objektes** benötigt Zeit bis zum Beobachter
- **Deflektion**: Ablenkung des Lichts durch schwere Objekte
- **Aberration**: "Neigung" der Lichtstrahlen, welche von bewegten Objekten auf bewegte Objekte treffen

Relativistisches Raum-Zeitgefüge

commons.wikipedia.org .. Zeitdilitation-Lichtuhr





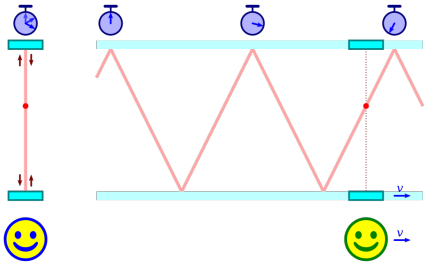
commons.wikipedia.org .. Spacetime lattice

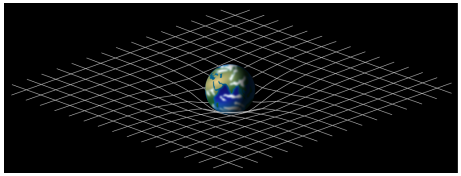
Astronom. Berechnungen müssen berücksichtigen:

- **bewegte Uhren** gehen langsamer
- **schwere Uhren** gehen langsamer
- Licht eines **entfernten Objektes** benötigt Zeit bis zum Beobachter
- **Deflektion**: Ablenkung des Lichts durch schwere Objekte
- **Aberration**: "Neigung" der Lichtstrahlen, welche von bewegten Objekten auf bewegte Objekte treffen

Relativistisches Raum-Zeitgefüge

commons.wikipedia.org .. Zeitdilitation-Lichtuhr





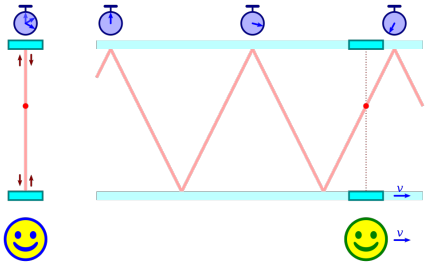
commons.wikipedia.org .. Spacetime lattice

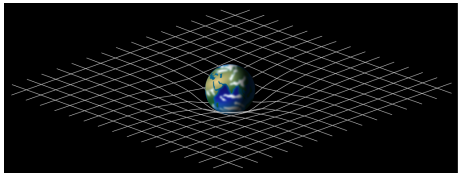
Astronom. Berechnungen müssen berücksichtigen:

- **bewegte Uhren** gehen langsamer
- **schwere Uhren** gehen langsamer
- Licht eines **entfernten Objektes** benötigt Zeit bis zum Beobachter
- **Deflektion**: Ablenkung des Lichts durch schwere Objekte
- **Aberration**: "Neigung" der Lichtstrahlen, welche von bewegten Objekten auf bewegte Objekte treffen

Relativistisches Raum-Zeitgefüge

commons.wikipedia.org .. Zeitdilitation-Lichtuhr





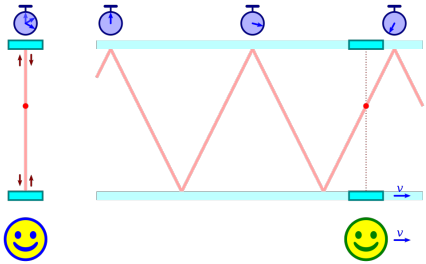
commons.wikipedia.org .. Spacetime lattice

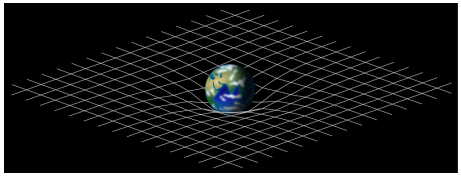
Astronom. Berechnungen müssen berücksichtigen:

- **bewegte Uhren** gehen langsamer
- **schwere Uhren** gehen langsamer
- Licht eines **entfernten Objektes** benötigt Zeit bis zum Beobachter
- **Deflektion**: Ablenkung des Lichts durch schwere Objekte
- **Aberration**: "Neigung" der Lichtstrahlen, welche von bewegten Objekten auf bewegte Objekte treffen

Relativistisches Raum-Zeitgefüge

commons.wikipedia.org .. Zeitdilitation-Lichtuhr





commons.wikipedia.org .. Spacetime lattice

Astronom. Berechnungen müssen berücksichtigen:

- **bewegte Uhren** gehen langsamer
- **schwere Uhren** gehen langsamer
- Licht eines **entfernten Objektes** benötigt Zeit bis zum Beobachter
- **Deflektion**: Ablenkung des Lichts durch schwere Objekte
- **Aberration**: “Neigung” der Lichtstrahlen, welche von bewegten Objekten auf bewegte Objekte treffen

Skyfield in Beispielen

Bezugssysteme in Skyfield

- **Barycentric** .. Basiskoordinatensystem des ICRS
- **Astrometric** .. Koordinaten eines Objektes unter Berücksichtigung der Zeitverzögerung
- **Apparent** .. Koordinaten eines Objektes unter Berücksichtigung von Deflektion und Aberration. Entspricht der Darstellung in einem astronom. Almanach.
- **Altitude-Azimut** .. Koordinaten bezogen auf einen Ort an der Erdoberfläche

Erstes Beispiel

In [1]:
```
import skyfield.api as api
ts = api.load.timescale()
ephem = api.load('de421.bsp')
```

```
[################################] 100% de421.bsp
```

- Wird beim ersten Mal heruntergeladen und fallweise erneuert:
  de421.bsp .. Ephemeriden für Sonne, Mond und Planeten (1900-2050)

Ephemeriden-Dateien

JPL (Jet Propulsion Laboratory, USA):
de* .. Planeten, Sonne, Erd-Mond

Issued	Short	Medium	Long
1997		`de405.bsp` 1600 to 2200 63 MB	`de406.bsp` -3000 to 3000 287 MB
2008	`de421.bsp` 1900 to 2050 17 MB		`de422.bsp` -3000 to 3000 623 MB
2013	`de430_1850-2150.bsp` 1850 to 2150 31 MB	`de430t.bsp` 1550 to 2650 128 MB	`de431t.bsp` -13200 to 17191 3.5 GB
2020	`de440s.bsp` 1849 to 2150 32 MB	`de440.bsp` 1550 to 2650 114 MB	`de441.bsp` -13200 to 17191 3.1 GB

Abb. aus https://rhodesmill.org/skyfield/planets.html

Ephemeriden-Datei

- Inhalt der Ephemeriden-Datei abfragen

```
In [1]: import skyfield.api as api
        ephem = api.load('de421.bsp')
        print(ephem)

        SPICE kernel file 'de421.bsp' has 15 segments
          JD 2414864.50 - JD 2471184.50  (1899-07-28 through 2053-10-08)
               0 -> 1    SOLAR SYSTEM BARYCENTER -> MERCURY BARYCENTER
               0 -> 2    SOLAR SYSTEM BARYCENTER -> VENUS BARYCENTER
               0 -> 3    SOLAR SYSTEM BARYCENTER -> EARTH BARYCENTER
               0 -> 4    SOLAR SYSTEM BARYCENTER -> MARS BARYCENTER
               0 -> 5    SOLAR SYSTEM BARYCENTER -> JUPITER BARYCENTER
               0 -> 6    SOLAR SYSTEM BARYCENTER -> SATURN BARYCENTER
               0 -> 7    SOLAR SYSTEM BARYCENTER -> URANUS BARYCENTER
               0 -> 8    SOLAR SYSTEM BARYCENTER -> NEPTUNE BARYCENTER
               0 -> 9    SOLAR SYSTEM BARYCENTER -> PLUTO BARYCENTER
               0 -> 10   SOLAR SYSTEM BARYCENTER -> SUN
               3 -> 301  EARTH BARYCENTER -> MOON
               3 -> 399  EARTH BARYCENTER -> EARTH
               1 -> 199  MERCURY BARYCENTER -> MERCURY
               2 -> 299  VENUS BARYCENTER -> VENUS
               4 -> 499  MARS BARYCENTER -> MARS
```

Timescale-Dateien

- Download neuester timescale-Dateien funktioniert oft nicht
- Default: Verwendung eingebauter timescale-Datei

In [1]:
```
import skyfield.api as api
import sys
try:
    ts = api.load.timescale(builtin=False)
except OSError as err:
    print('OSError: ',err, file=sys.stderr)
    print('Using builtin timescales ..', file=sys.stderr)
    ts = api.load.timescale(builtin=True)
# etc ..
```

```
OSError:  cannot download ftp://ftp.iers.org/products/eop/rapid/standar
d/finals2000A.all because <urlopen error [Errno 111] Connection refused>
Using builtin timescales ..
```

Alternative Formulierung (Off-Line)

```
In [1]: import skyfield.api as api
        load = api.Loader('~/.local/share/skyfield', expire=False)
        ts = load.timescale(builtin=True)
        ephem = load('de421.bsp')
```

- Ephemeriden und Timescales in eigenem Verzeichnis
- Sobald Ephemeriden einmal heruntergeladen wurden, erfolgt kein Update mehr
- Per Default kein automatischer Downloadversuch für die Timescales
- Im Notfall **manueller Download** für Timescales möglich, z.B.: https://datacenter.iers.org/data/9/finals2000A.all

Beispiel - Zeitvergleich

```
from skyfield import api
ts = api.load.timescale(builtin=True)
t = ts.now()
```

```
print('UTC ', t.utc)
print('TT  ', t.tt_calendar())
print('TAI ', t.tai_calendar())
print(50 * '-')
print('UT1 JD ', t.ut1)
print('TT  JD ', t.tt)
print('TDB JD ', t.tdb)   # Barycentric Dynamical Time as Julian Date
print('TAI JD ', t.tai)
```

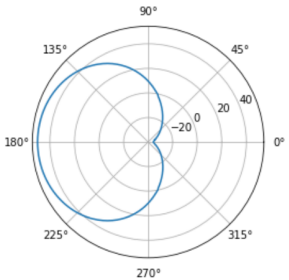
```
UTC  (2020, 3, 5, 12, 21, 27.926951050758362)
TT   (2020, 3, 5, 12, 22, 37.11096525192261)
TAI  (2020, 3, 5, 12, 22, 4.92695152759552)
--------------------------------------------------
UT1 JD  2458914.0148973092
TT  JD  2458914.015707303
TDB JD  2458914.01570732
TAI JD  2458914.0153348027
```

Beispiel - Sonnenbahn heute in Graz

In [1]:
```
import skyfield.api as api                          # API inportieren
import numpy as np                                  # numpy für arrays

load = api.Loader('~/.local/share/skyfield',     # Verzeichnis mit Daten angeben,
                  expire=False)                  # Aktualisierung verhindern (offline?)

ts = load.timescale(builtin=True)                   # eingebauten Zeitscale laden (offline?)
t = ts.utc(2024,4,7,hour=np.linspace(0,24,241))  # Zeitobjekt als array definieren
ephem = load('de421.bsp')                           # "kleine" Ephemeriden laden
sun, earth = ephem['sun'], ephem['earth']           #  Sonnen- und Erd-Ephemeriden

graz = earth + api.wgs84.latlon(47.0781,  # ein wgs84-Geoids-Objekt befüllen
                                15.4491)
astrometric = graz.at(t).observe(sun)               #  mit Licht-Zeit-Verzögerung
apparent = astrometric.apparent()                   #  mit Deflektion und Aberration
alt, az, dist = apparent.altaz(temperature_C = 15.0,  # Berücksichtigung der
                               pressure_mbar=960.)    # Lichtbrechung durch die Atmosphäre
```

Beispiel - Sonnenbahn heute in Graz

In [2]:
```
# Eine linkische Darstellung
%matplotlib inline
import matplotlib.pyplot as plt
h = plt.polar(np.array(az.radians), np.array(alt.degrees))    # ACHTUNG: Interpretation!
```



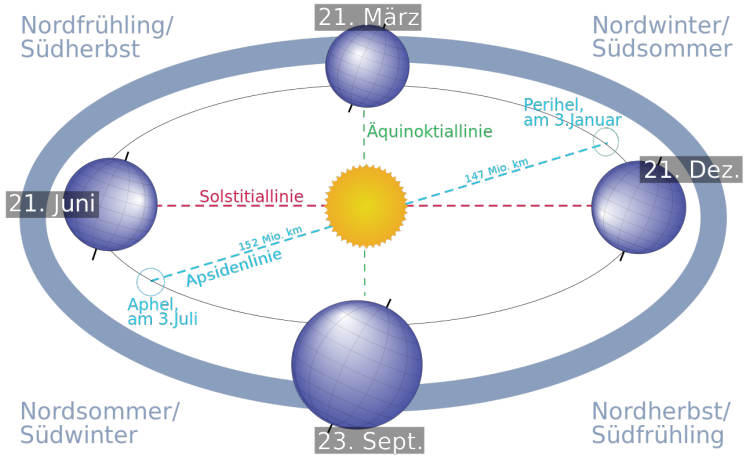
Utility Functions in **skyfield.almanac**

Nachbildung der Ergebisse des **Astronomical Almanac** des **USNO**
(derzeit online nur eingeschränkt erreichbar: https://www.usno.navy.mil/USNO)

- **moon_phases(ephemeris)** .. Mondphasen von der Erde ausgesehen (erstes bis letztes Viertel)
- **dark_twilight_day(ephemeris,wgs84)** .. Dämmerung, Sonnenauf- und -untergang
- **seasons(ephemeris)** .. Beginn der Jahreszeiten
- **risings_and_settings(ephemeris, target, wgs84, ..)** ..
  Auf- und -untergang beliebiger Objekte
- .. und anderes mehr

find_discrete(start_time, end_time, f[..,..]) ..
Funktion, die den Werte-Umschlag der Funktion f zwischen 2 Zeiten findet.

Beispiel - Jahreszeiten



https://upload.wikimedia.org/wikipedia/commons/0/04/Four_season_german_infotext.svg

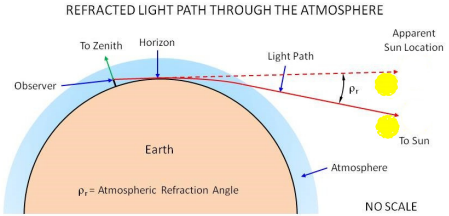
Beispiel - Jahreszeiten (Code)

In [1]:
```
import skyfield.api as api
import skyfield.almanac as almanac
load = api.Loader('~/.local/share/skyfield',
                  expire=False)
ts = load.timescale(builtin=True)
ephem = load('de421.bsp')
t0 = ts.utc(2024,1,1)
t1 = ts.utc(2024,12,31)
t, y = almanac.find_discrete(t0, t1, almanac.seasons(ephem))

for yi, ti in zip(y, t):
    print(yi, almanac.SEASON_EVENTS[yi], ti.utc_iso(' '))
```

```
0 Vernal Equinox 2024-03-20 03:06:24Z
1 Summer Solstice 2024-06-20 20:51:00Z
2 Autumnal Equinox 2024-09-22 12:43:40Z
3 Winter Solstice 2024-12-21 09:20:34Z
```

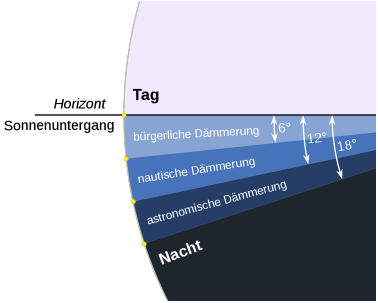
Beispiel - Dämmerung und Sonnenaufgang

- Sonnenauf-/-untergang



Aus http://astronomyinsider.net (leicht adaptiert)

- Dämmerung



Aus Wikipedia - TWCarlson:
https://upload.wikimedia.org/wikipedia/commons/a/a9/Twilight_subcategories_de.svg

Beispiel - Dämmerung, Sonnenauf- und -untergang (Code)

```
import skyfield.api as api
import skyfield.almanac as almanac

load = api.Loader('~/.local/share/skyfield', expire=False) # data directory, no updating
ts = load.timescale(builtin=True)                  # eingebauten Zeitscale laden (offline?)
t0 = ts.utc(2024,4,7, 0)                           # Zeitobjekt als array definieren
t1 = ts.utc(2024,4, 7, 24)
ephem = load('de421.bsp')                          # "kleine" Ephemeriden laden
graz = api.wgs84.latlon(47.0781, 15.4491)          # ein wgs84-Objekt befüllen,
                                                   # ACHTUNG: Erd-Vektor NICHT addieren!


#  "dark_twilight_day" ist die Funktion für Sonnenstände (Dämmerungen, etc...)
f = almanac.dark_twilight_day(ephem, graz)
#  "find_discrete" ist eine Art "Nullstellen"-Löser
t, y = almanac.find_discrete(t0, t1, f)
```

Beispiel - Dämmerung, Sonnenauf- und -untergang (Ergebnis)

```python
for ti, yi in zip(t, y):
    print(yi, ti.utc_iso(), ' Start of', almanac.TWILIGHTS[yi])
```

```
1 2024-04-07T02:36:03Z  Start of Astronomical twilight
2 2024-04-07T03:16:03Z  Start of Nautical twilight
3 2024-04-07T03:53:41Z  Start of Civil twilight
4 2024-04-07T04:24:57Z  Start of Day
3 2024-04-07T17:36:21Z  Start of Civil twilight
2 2024-04-07T18:07:44Z  Start of Nautical twilight
1 2024-04-07T18:45:33Z  Start of Astronomical twilight
0 2024-04-07T19:25:50Z  Start of Night
```

Resumé

Zusammenfassung - Allgemeines Schema (1)

Aus **skyfield.api**

1. **Load**-Objekt (implizit oder explizit)
2. **Timescale** definieren
3. **Zeit**-Objekt definieren
4. **Ephemeriden** laden
5. **“Planeten”** aus den Emphmriden abrufen

usw. ..

z.B. ..

```
load = api.Loader(<skyfield-dir>)
ts = load.timescale(<optionen>)
t = ts.utc(2020,4,18,12)
ephem = load('de421.bsp')
earth =  ephem['earth']
sun = ephem['sun']
...
```

Zusammenfassung - Allgemeines Schema (2)

Danach je nach weitere Absicht z.b. Objekte aus der **skyfield.api**

1. **wgs84**-Objekt definieren
2. **astrometric**-Objekt
3. **apparent**-Objekt
4. **altaz**-Methode

und/oder

- Objekte/Methoden aus **skyfield.almanach**

Auswahl an weiteren Fähigkeiten von **Skyfield**

Lose Auflistung ..

- Sternkataloge laden: `skyfield.data.hypparcos`
- Sternbildern: `skyfield.api.load_constellation_map()`
- Erdsatelliten
- Zeitpunkt astronomischer Ereignisse
- Interface zu **AstroPy**

.. und einiges mehr

Vor- und Nachteile von Skyfield

- + Einfache Installation,
- + kaum Abhängigkeiten
- + Hohe Exaktheit der Rechnung
- + Formale Eleganz der Bibliothek

- - Erfordert erhöhtes Wissen
- - Datenfiles aus dem Netz zu laden

Danke für die Aufmerksamkeit

Fragen?

Download Vortrag und Jupyter-Notebooks unter
https://glt.foehnwall.at/glt24.html