
TLE-tools

Release 0.2.0

Federico Stra

May 24, 2020

CONTENTS:

1	Purpose	3
2	Installation	5
3	Links	7
4	Indices and tables	9
4.1	API Documentation	9
	Python Module Index	15
	Index	17

TLE-tools is a small library to work with [two-line element set](#) files.

PURPOSE

The purpose of the library is to parse TLE sets into convenient *TLE* objects, load entire TLE set files into `pandas.DataFrame`'s, convert *TLE* objects into `poliastro.twobody.orbit.Orbit`'s, and more.

From [Wikipedia](#):

A two-line element set (TLE) is a data format encoding a list of orbital elements of an Earth-orbiting object for a given point in time, the epoch. The TLE data representation is specific to the [simplified perturbations models](#) (SGP, SGP4, SDP4, SGP8 and SDP8), so any algorithm using a TLE as a data source must implement one of the SGP models to correctly compute the state at a time of interest. TLEs can describe the trajectories only of Earth-orbiting objects.

Here is an example TLE:

```
ISS (ZARYA)
1 25544U 98067A   19249.04864348   .00001909   00000-0   40858-4 0   9990
2 25544   51.6464 320.1755 0007999   10.9066   53.2893 15.50437522187805
```

Here is a minimal example on how to load the previous TLE:

```
from tletools import TLE

tle_string = """
ISS (ZARYA)
1 25544U 98067A   19249.04864348   .00001909   00000-0   40858-4 0   9990
2 25544   51.6464 320.1755 0007999   10.9066   53.2893 15.50437522187805
"""

tle_lines = tle_string.strip().splitlines()

t = TLE.from_lines(*tle_lines)
```

Then `t` is:

```
TLE(name='ISS (ZARYA)', norad='25544', classification='U', int_desig='98067A',
epoch_year=2019, epoch_day=249.04864348, dn_o2=1.909e-05, ddn_o6=0.0, bstar=4.0858e-
→05,
set_num=999, inc=51.6464, raan=320.1755, ecc=0.0007999, argp=10.9066, M=53.2893,
n=15.50437522, rev_num=18780)
```

and you can then access its attributes like `t.argp`, `t.epoch`...

INSTALLATION

Install and update using `pip`:

```
pip install -U TLE-tools
```


LINKS

- Website: <https://federicostra.github.io/tletools>
- Documentation: <https://tletools.readthedocs.io>
- Releases: <https://pypi.org/project/TLE-tools>
- Code: <https://github.com/FedericoStra/tletools>
- Issue tracker: <https://github.com/FedericoStra/tletools/issues>

INDICES AND TABLES

- [genindex](#)
- [modindex](#)
- [search](#)

4.1 API Documentation

If you are looking for information on a specific function, class, or method, this part of the documentation is for you.

4.1.1 API Documentation

This part of the documentation covers all the interfaces of `tletools`. For guides on how to use them, please consult the tutorials.

TLE Classes

The module `tletools.tle` defines the classes `TLE` and `TLEu`.

The library offers two classes to represent a single TLE. There is the unitless version `TLE`, whose attributes are expressed in the same units that are used in the TLE format, and there is the unitful version `TLEu`, whose attributes are quantities (`astropy.units.Quantity`), a type able to represent a value with an associated unit taken from `astropy.units`. Here is a short example of how you can use them:

```
>>> tle_string = """
... ISS (ZARYA)
... 1 25544U 98067A   19249.04864348   .00001909   00000-0   40858-4 0   9990
... 2 25544   51.6464 320.1755 0007999   10.9066   53.2893 15.50437522187805
... """
>>> tle_lines = tle_string.strip().splitlines()
>>> TLE.from_lines(*tle_lines)
TLE(name='ISS (ZARYA)', norad='25544', ..., n=15.50437522, rev_num=18780)
```

class `tletools.tle.TLE`(*name, norad, classification, int_desig, epoch_year, epoch_day, dn_o2, ddh_o6, bstar, set_num, inc, raan, ecc, argp, M, n, rev_num*)
Data class representing a single TLE.

A two-line element set (TLE) is a data format encoding a list of orbital elements of an Earth-orbiting object for a given point in time, the epoch.

All the attributes parsed from the TLE are expressed in the same units that are used in the TLE format.

Variables

- **name** (*str*) – Name of the satellite.
- **norad** (*str*) – NORAD catalog number (https://en.wikipedia.org/wiki/Satellite_Catalog_Number).
- **classification** (*str*) – ‘U’, ‘C’, ‘S’ for unclassified, classified, secret.
- **int_desig** (*str*) – International designator (https://en.wikipedia.org/wiki/International_Designator).
- **epoch_year** (*int*) – Year of the epoch.
- **epoch_day** (*float*) – Day of the year plus fraction of the day.
- **dn_o2** (*float*) – First time derivative of the mean motion divided by 2.
- **ddn_o6** (*float*) – Second time derivative of the mean motion divided by 6.
- **bstar** (*float*) – BSTAR coefficient (<https://en.wikipedia.org/wiki/BSTAR>).
- **set_num** (*int*) – Element set number.
- **inc** (*float*) – Inclination.
- **raan** (*float*) – Right ascension of the ascending node.
- **ecc** (*float*) – Eccentricity.
- **argp** (*float*) – Argument of perigee.
- **M** (*float*) – Mean anomaly.
- **n** (*float*) – Mean motion.
- **rev_num** (*int*) – Revolution number.

asdict (*computed=False, epoch=False*)
Return a dict of the attributes.

astuple ()
Return a tuple of the attributes.

classmethod from_lines (*name, line1, line2*)
Parse a TLE from its constituent lines.

All the attributes parsed from the TLE are expressed in the same units that are used in the TLE format.

classmethod load (*filename*)
Load multiple TLEs from a file.

classmethod loads (*string*)
Load multiple TLEs from a string.

to_orbit (*attractor=Earth* ())
Convert to a `poliastro.twobody.orbit.Orbit` around the attractor.

```
>>> tle_string = """ISS (ZARYA)
... 1 25544U 98067A   19249.04864348   .00001909   00000-0   40858-4 0   9990
... 2 25544   51.6464 320.1755 0007999   10.9066  53.2893 15.50437522187805"""
>>> tle = TLE.from_lines(*tle_string.splitlines())
>>> tle.to_orbit()
6788 x 6799 km x 51.6 deg (GCRS) orbit around Earth () at epoch 2019-09-
↪ 06T01:10:02.796672000 (UTC)
```

property a
Semi-major axis.

property epoch
Epoch of the TLE.

property nu
True anomaly.

class `tletools.tle.TLEu`(*name, norad, classification, int_desig, epoch_year, epoch_day, dn_o2, ddn_o6, bstar, set_num, inc, raan, ecc, argp, M, n, rev_num*)
Unitful data class representing a single TLE.

This is a subclass of `TLE`, so refer to that class for a description of the attributes, properties and methods.

The only difference here is that all the attributes are quantities (`astropy.units.Quantity`), a type able to represent a value with an associated unit taken from `astropy.units`.

Interoperability

Pandas

The module `tletools.pandas` provides convenience functions to load two-line element set files into `pandas.DataFrame`'s.

Given a file `oneweb.txt` with the following contents:

```
ONEWEB-0012
1 44057U 19010A 19290.71624163 .00000233 00000-0 58803-3 0 9997
2 44057 87.9055 22.9851 0002022 94.9226 265.2135 13.15296315 30734
ONEWEB-0010
1 44058U 19010B 19290.71785289 .00000190 00000-0 47250-3 0 9991
2 44058 87.9054 22.9846 0002035 97.1333 263.0028 13.15294565 30783
ONEWEB-0008
1 44059U 19010C 19290.86676214 -.00000034 00000-0 -12353-3 0 9990
2 44059 87.9055 22.9563 0001967 95.9628 264.1726 13.15300216 30897
ONEWEB-0007
1 44060U 19010D 19290.87154896 .00000182 00000-0 45173-3 0 9998
2 44060 87.9067 22.9618 0001714 97.9802 262.1523 13.15299021 30927
ONEWEB-0006
1 44061U 19010E 19290.72095254 .00000179 00000-0 44426-3 0 9991
2 44061 87.9066 22.9905 0001931 95.0539 265.0811 13.15294588 30940
ONEWEB-0011
1 44062U 19010F 19291.17894923 .00000202 00000-0 50450-3 0 9993
2 44062 87.9056 22.8943 0002147 94.8298 265.3077 13.15300820 31002
```

you can load the TLEs into a `pandas.DataFrame` by using

```
>>> load_dataframe("oneweb.txt")
      name norad classification int_desig epoch_year epoch_day dn_o2
→ ddn_o6   bstar set_num      inc   raan      ecc   argp      M      n
→ rev_num                                     epoch
0  ONEWEB-0012 44057          U   19010A      2019  290.716242  2.330000e-06
→   0.0  0.000588      999  87.9055  22.9851  0.000202  94.9226  265.2135  13.152963
→   3073 2019-10-17 17:11:23.276832
1  ONEWEB-0010 44058          U   19010B      2019  290.717853  1.900000e-06
→   0.0  0.000472      999  87.9054  22.9846  0.000204  97.1333  263.0028  13.152946
→   3078 2019-10-17 17:13:42.489696
```

(continues on next page)

(continued from previous page)

```

2  ONEWEB-0008  44059          U    19010C          2019  290.866762 -3.400000e-07  ␣
→  0.0 -0.000124          999  87.9055  22.9563  0.000197  95.9628  264.1726  13.153002  ␣
→  3089 2019-10-17 20:48:08.248896
3  ONEWEB-0007  44060          U    19010D          2019  290.871549  1.820000e-06  ␣
→  0.0  0.000452          999  87.9067  22.9618  0.000171  97.9802  262.1523  13.152990  ␣
→  3092 2019-10-17 20:55:01.830144
4  ONEWEB-0006  44061          U    19010E          2019  290.720953  1.790000e-06  ␣
→  0.0  0.000444          999  87.9066  22.9905  0.000193  95.0539  265.0811  13.152946  ␣
→  3094 2019-10-17 17:18:10.299456
5  ONEWEB-0011  44062          U    19010F          2019  291.178949  2.020000e-06  ␣
→  0.0  0.000504          999  87.9056  22.8943  0.000215  94.8298  265.3077  13.153008  ␣
→  3100 2019-10-18 04:17:41.213472

```

You can also load multiple files into a single `pandas.DataFrame` with

```

>>> from glob import glob
>>> load_dataframe(glob("*.txt"))

```

`tletools.pandas.add_epoch(df)`

Add a column 'epoch' to a dataframe.

`df` must have columns 'epoch_year' and 'epoch_day', from which the column 'epoch' is computed.

Parameters `df` (`pandas.DataFrame`) – `pandas.DataFrame` instance to modify.

Example

```

>>> from pandas import DataFrame
>>> df = DataFrame([[2018, 31.2931], [2019, 279.3781]],
...               columns=['epoch_year', 'epoch_day'])
>>> add_epoch(df)
>>> df
   epoch_year  epoch_day          epoch
0         2018    31.2931 2018-01-31 07:02:03.840
1         2019   279.3781 2019-10-06 09:04:27.840

```

`tletools.pandas.load_dataframe(filename, *, computed=False, epoch=True)`

Load multiple TLEs from one or more files and return a `pandas.DataFrame`.

Parameters `filename` (`str` or `iterable`) – A single filename (`str`) or an iterable producing filenames.

Returns A `pandas.DataFrame` with all the loaded TLEs.

Examples

```

>>> load_dataframe("oneweb.txt")

```

```

>>> load_dataframe(["oneweb.txt", "starlink.txt"])

```

```

>>> from glob import glob
>>> load_dataframe(glob("*.txt"))

```


Poliastro

Use the `TLE.to_orbit()` method like this:

```
>>> tle_string = """ISS (ZARYA)
... 1 25544U 98067A   19249.04864348   .00001909   00000-0   40858-4   0   9990
... 2 25544   51.6464 320.1755 0007999   10.9066   53.2893 15.50437522187805"""
>>> tle = TLE.from_lines(*tle_string.splitlines())
>>> tle.to_orbit()
6788 x 6799 km x 51.6 deg (GCRS) orbit around Earth () at epoch 2019-09-06T01:10:02.
↪ 796672000 (UTC)
```

Utils

`tletools.utils.M_to_nu(M, ecc)`

True anomaly from mean anomaly.

New in version 0.2.3.

Parameters

- **M** (*float*) – Mean anomaly in radians.
- **ecc** (*float*) – Eccentricity.

Returns *nu*, the true anomaly, between - and radians.

Warning

The mean anomaly must be between - and radians. The eccentricity must be less than 1.

Examples

```
>>> M_to_nu(0.25, 0.)
0.25
```

```
>>> M_to_nu(0.25, 0.5)
0.804298286591367
```

```
>>> M_to_nu(5., 0.)
Traceback (most recent call last):
...
AssertionError
```

`tletools.utils.partition(iterable, n, rest=False)`

Partition an iterable into tuples.

The iterable *iterable* is progressively consumed *n* items at a time in order to produce tuples of length *n*.

Parameters

- **iterable** (*iterable*) – The iterable to partition.
- **n** (*int*) – Length of the desired tuples.
- **rest** (*bool*) – Whether to return a possibly incomplete tuple at the end.

Returns A generator which yields subsequent *n*-uples from the original iterable.

Examples

By default, any remaining items which are not sufficient to form a new tuple of length *n* are discarded.

```
>>> list(partition(range(8), 3))
[(0, 1, 2), (3, 4, 5)]
```

You can ask to return the remaining items at the end by setting the flag *rest* to `True`.

```
>>> list(partition(range(8), 3, rest=True))
[(0, 1, 2), (3, 4, 5), (6, 7)]
```

```
tletools.utils.dt_dt64_Y = dtype('<M8[Y]')
    numpy.dtype for a date expressed as a year.
```

```
tletools.utils.dt_td64_us = dtype('<m8[us]')
    numpy.dtype for a timedelta expressed in microseconds.
```

```
tletools.utils.rev = Unit("rev")
    astropy.units.Unit of angular measure: a full turn or rotation. It is equivalent to astropy.units.
    cycle.
```

PYTHON MODULE INDEX

t

`tletools.pandas`, [11](#)
`tletools.tle`, [9](#)
`tletools.utils`, [13](#)

INDEX

A

`a()` (*tletools.tle.TLE property*), 10
`add_epoch()` (*in module tletools.pandas*), 12
`asdict()` (*tletools.tle.TLE method*), 10
`astuple()` (*tletools.tle.TLE method*), 10

D

`dt_dt64_Y` (*in module tletools.utils*), 14
`dt_td64_us` (*in module tletools.utils*), 14

E

`epoch()` (*tletools.tle.TLE property*), 11

F

`from_lines()` (*tletools.tle.TLE class method*), 10

L

`load()` (*tletools.tle.TLE class method*), 10
`load_dataframe()` (*in module tletools.pandas*), 12
`loads()` (*tletools.tle.TLE class method*), 10

M

`M_to_nu()` (*in module tletools.utils*), 13
module
 tletools.pandas, 11
 tletools.tle, 9
 tletools.utils, 13

N

`nu()` (*tletools.tle.TLE property*), 11

P

`partition()` (*in module tletools.utils*), 13

R

`rev` (*in module tletools.utils*), 14

T

`TLE` (*class in tletools.tle*), 9
tletools.pandas
 module, 11

tletools.tle
 module, 9
tletools.utils
 module, 13
`TLEu` (*class in tletools.tle*), 11
`to_orbit()` (*tletools.tle.TLE method*), 10