# TLE-tools

***Release 0.2.0***

**Federico Stra**

**Oct 18, 2019**

# CONTENTS:

**TLE-tools** is a small library to work with two-line element set files.

CHAPTER

# ONE

# 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`'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`...

CHAPTER

TWO

# INSTALLATION

Install and update using pip:

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

CHAPTER

# THREE

# 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

CHAPTER

# FOUR

# 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, pleas 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*, *ddn_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 an orbit around the attractor.

**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.'

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`.

### Poliastro

*coming soon*

### Utils

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

# INDEX