Convert TLE records to h5 using pytables

In [ ]:
```
import calendar
from datetime import datetime, timedelta
import itertools
import os
import time
import zipfile

from tables import *

import skyfield
from skyfield.sgp4lib import EarthSatellite
```

In [ ]:
```
class TLE(IsDescription):
    epoch     = Float64Col(pos=0)
    norad_id  = Int64Col(pos=1)
    line_one  = StringCol(80, pos=2)
    line_two  = StringCol(80, pos=3)

export_path = "data/satellite data/TLE/reexport.h5"
def create_new_h5_with_tle_table(path):
    h5file = open_file(path, mode="w", title="TLE Indexable Data")
    return h5file.create_table(h5file.root, 'tle', TLE, "Main TLE Listing")
```

In [ ]:
```
tle_folder = "data/satellite data/TLE"
zips = [os.path.join(tle_folder, f) for f in os.listdir(tle_folder)]
```

In [ ]:
```
def fmt_epoch(epoch_val):
    return time.strftime('%Y-%m-%d %H:%M:%S', time.localtime(epoch_val))
```

In [ ]:

```
def read_tles_from_flo(flo):
    """
    Produce an iterator of subsequent

    """
    for n in itertools.count():
        tle1 = f.readline().decode().strip()
        if not tle1: break
        tle2 = f.readline().decode().strip()
        sat = EarthSatellite(tle1, tle2)

        epoch = sat.epoch.utc_datetime().timestamp()
        norad_id = sat.model.satnum

        yield epoch, norad_id, tle1, tle2

def read_tles_from_zip(path: str):
    """
    Iterate through the rows in the TLE file inside of a zip file.
    """
    with zipfile.ZipFile(path) as z:
        # Assumes only one file contained inside the zip, ignores OSX detritus
        name = list(filter(lambda fn: "_MACOSX" not in fn, z.namelist()))[0]
        with z.open(name) as f:
            return read_tles_from_flo(f)

def read_tles_from_csv(path: str):
    """
    Iterate through the rows in a csvfile
    """
    with open(path, newline='') as csvfile:
        spamreader = csv.DictReader(csvfile)
        for row in spamreader:
            epoch = datetime(row["epoch_year"], 1, 1) + timedelta(days=row["epoch_day"] - 1)
            tle1 = row["tle"][0:80]
            tle2 = row["tle"][80:]
            yield epoch, row["norad_id"], tle1, tle2

def read_to_table(table_node, src_iter, limit=float("inf")):
    """
    Given an iterable (likely consturcted from one of the above) and a table.
    Populate the table with values from the interable.
    """

    entry = table_node.row

    for i, (entry, norad_id, tle1, tle2) in enumerate(src_iter):
        entry["epoch"] = entry
        entry["norad_id"] = norad_id
        entry["line_one"] = tle1
        entry["line_two"] = tle2

        entry.append()

        if i % 1000 == 0:
            table.flush()

        if i > limit:
            break

    table.flush()
```

In [ ]:
```
def read_directory_into_table(d, table_node):
    assert, os.path.isdir(d)

    paths = list(os.path.join(directory, f) for f in os.listdir(d))
    for path in paths:
        iterable = None

        if path.endswith("csv"):
            iterable = read_tles_from_csv(path)

        if path.endswith("zip"):
            iterable = read_tles_from_zip(path)

        if iterable is None:
            print("Ignoring file: " + path)

        before_rows = table_node.nrows
        read_to_table(table_node, iterable)
        added_rows = table_node.nrows - before_rows
        print("Added %i entries from %s. Total size: %i" % (added_rows, path, table_node.nrows))

def build_indices(table):
    print("Building indices")
    table.cols.epoch.create_index()
    table.cols.norad_id.create_index()


print("Done..")
h5file.close()
```

```
!cd /mnt/disk100/persist_home/meawoppl/
!ls /mnt/disk100/persist_home/meawoppl/
```

```
!du -sch *
```