ReadFile

Wednesday, November 26, 2025 7:55 PM

Python module containing a data-loader function (using the packages numpy and astropy) to parse snapshot files and extract time, particle count, and structured arrays for further analysis.

```
import numpy as np
import astropy.units as u

def Read(filename):
    with open(filename, 'r') as file:
```

The character 'r' means open for reading (default). Other features like 'w' (writing), 'x' (exclusive creation), 'a' (open for writing, appending to the end of the file if it exists), 'b' (binary mode), 't' (text mode: default), '+' (open for updating).

```
        line1 = file.readline()
```

→ this reads the first line of the tar file snapshot.

```
        _, value = line1.split()
```

line1 is a string containing the first line of the snapshot. Given how the data is organized in the snapshots, i.e. Time 14.28570,

line1.split() splits the string into a list of words/values separated by whitespace.

∴ line1.split() # -> ['Time', '14.28570']

_, value = ... : this is tuple unpacking

'_' functions as an arbitrary variable, thus capturing the list element 'Time': → throwaway

'value' captures the numeric value for time.

```
time = float(value) * Myr
```

: crucial, convert time to a usable value and show that the snapshots are spaced apart with time elapsing on the order of 10^6 years.

```
line2 = file.readline()
```

→ reads the second line of the file.

for the .tar files, it will read something like 'Total 14300'

```
value = line2.split()
```

→ ['Total', '14300']

```
count = float(value)
```

→ total particle count

```
data = np.genfromtxt(filename, dtype=None, names=True, skip_header=3)
```

↙ function for reading tabular data from a txt file.

(like np.loadtxt, but more flexible → can handle missing values, named columns and a variety of data types)

(like np.loadtxt, but more flexible → can handle missing values, named columns and a variety of data types)

filename : the file being read.

dtype = None : Numpy automatically determines the data type of each column.

names = True : The first line of the remaining data (after skipping header) is used as column names.

so it takes : # type, m, x, y, z, vx, vy, vz

Each column is accessible by its name :-

```
data['x'] # array of x positions
data['vx'] # array of x velocities
```

names = True expects the first line after skip_header to be column names.

so, skip_header = 3 ensures you skip the first 2 metadata lines, and the third line contains the column names.

if skip=0, the first line of the file would be treated as the column names.

```
# Now, data is a structured Numpy array.
```

Each column of the array can be accessed by :-

```
data['type'] → particle type
data['m']    → mass
data['vx']   → x velocities
...
return time, count, data
```

```
# We get 1.) time 2.) count 3.) data :-
```

1.) Simulation time of the snapshot.

2.) This returns the total number of particles in the snapshot.

3.) This is a structured NumPy array containing the tabular data of all particles.