

Public Data Access Overview (/data/) / Example Scripts

We provide example scripts for reading the data files of the Illustris[TNG] simulations. They are available in a few languages, with identical functionality in each. This includes:

- (i) reading a given particle type and/or data field from the snapshot files,
- (ii) reading only the particle subset from the snapshot corresponding to a halo or subhalo,
- (iii) extracting the full subtree or main progenitor branch from either SubLink or LHaloTree for a given subhalo,
- (iv) walking a tree to count the number of mergers,
- (v) reading the entire group catalog at one snapshot,
- (vi) reading specific fields from the group catalog, or the entries for a single halo or subhalo.

We expect they will provide a useful starting point for writing any analysis task, and intend them as a 'minimal working examples' which are short and simple enough that they can be quickly understood and extended.

Currently available are: **Python** (3.6+ recommended), **IDL** (8.0+ required), **Matlab** (R2013a+ required), and **Julia** (1.6+ recommended). Select one to show all the content on this page specifically for that language.

[Python](#)

[IDL](#)

[Matlab](#)

[Julia](#)

In all cases, these scripts assume that you have downloaded local copies of the relevant files. Paths to files are defined with respect to a `basePath` which is passed in to all read functions. The locations of group catalog files, snapshot files, and merger trees files are then specified in a handful of functions, e.g. `gcPath()`, `snapPath()`, and `treePath()`. These can be modified as necessary to point to your local files, but it is best to keep a directory structure organized as e.g.:

- TNG100-1/
- TNG100-1/output/
 - group catalogs: TNG100-1/output/groups_099/fof_subhalo_tab_099.*.hdf5
 - snapshots: TNG100-1/output/snapdir_099/snap_099.*.hdf5
- TNG100-1/postprocessing/
 - offsets: TNG100-1/postprocessing/offsets/offsets_*.hdf5
 - SubLink mergertree: TNG100-1/postprocessing/trees/SubLink/tree_extended.*.hdf5
 - other catalogs: TNG100-1/postprocessing/catalog_name/files*.hdf5

where you would set `basePath = 'TNG100-1/output/'` in the following.

Getting Started Guide

We walk through the process of downloading and exploring data in the Subfind group catalogs, raw snapshot files, and the SubLink merger tree. In these examples we will work with **Illustris-3** since the file sizes are smaller to download and easier to work with, although you can replace each occurrence with e.g. 'Illustris-1', 'TNG100-1', or 'TNG300-1' for the flagship simulations.

If you haven't already, let's download the example scripts from Github (<https://www.github.com/illustristng/>) and make sure they installed (Python and Julia) or on the appropriate path for loading (Matlab and IDL). Specifically:

```
$ cd ~
$ git clone https://github.com/illustristng/illustris_python.git
$ pip install illustris_python/
```

Group Catalogs

First, make a base directory for the run and a subdirectory for the $z=0$ group catalogs (snapshot 135), then download the catalog (~100 MB).

```
$ mkdir -p ~/Illustris-3/output/groups_135
$ cd ~/Illustris-3/output/groups_135/
$ wget -nd -nc -nv -e robots=off -l 1 -r -A hdf5 --content-disposition --header="API-Key: 544f44ca2820e2600e6fd85e1b553564" "http://www.tng-project.org/api/Illustris-3/files/groupcat-135/?format=api"
```

Note: Separate offset files for TNG

For all TNG simulations the "offset" files are separate and must be downloaded in addition to the group catalog files, in order to load (i) single halos or subhalos, or (ii) load the particles of particular halos or subhalos. For example, if you were working with TNG100-3 at $z=0$, then:

```
$ mkdir -p ~/TNG100-3/postprocessing/offsets/
$ cd ~/TNG100-3/postprocessing/offsets/
$ wget --content-disposition --header="API-Key: 544f44ca2820e2600e6fd85e1b553564" "http://www.tng-project.org/api/TNG100-3/files/offsets.99.hdf5"
```

Start up your interface of choice and import the example scripts:

```
$ python
>>> import illustris_python as il
>>>
```

Define the base path for the data (modify as needed), and load the total masses (SubhaloMass) and star formation rate within twice the stellar half mass radius (SubhaloSFRinRad) of all the Subfind subhalos.

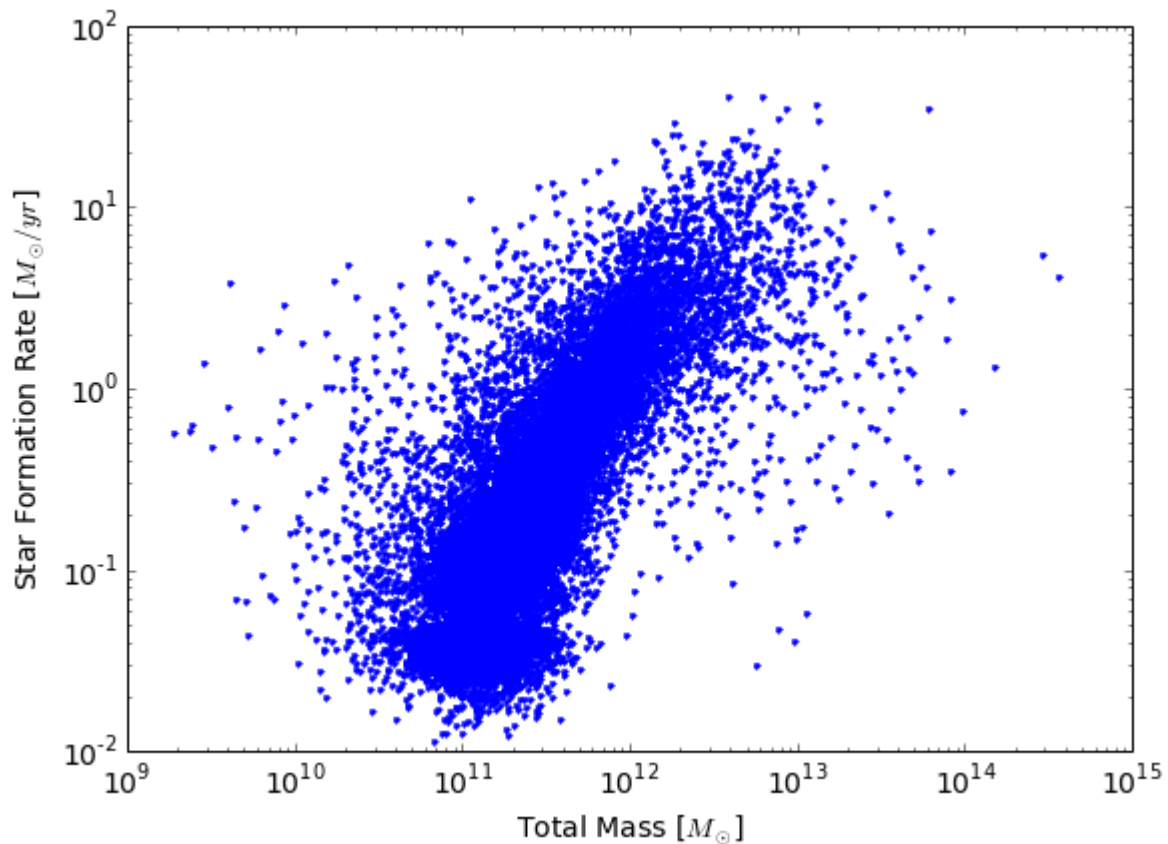
```
>>> basePath = './Illustris-3/output/'
>>> fields = ['SubhaloMass', 'SubhaloSFRinRad']
>>> subhalos = il.groupcat.loadSubhalos(basePath, 135, fields=fields)
```

Inspecting the return, we see it is a dict/hash/struct with a key `count` indicating that there are **121209** total subhalos. Each requested field is returned as a numeric array with a key name equal to its field name in the group catalog.

```
>>> subhalos.keys()
['count', 'SubhaloSFRinRad', 'SubhaloMass']
>>> subhalos['SubhaloMass'].shape
(121209,)
```

Make a simple scatterplot of the relation (note the units).

```
>>> import matplotlib.pyplot as plt
>>> mass_msun = subhalos['SubhaloMass'] * 1e10 / 0.704
>>> plt.plot(mass_msun, subhalos['SubhaloSFRinRad'], '.')
>>> plt.xscale('log')
>>> plt.yscale('log')
>>> plt.xlabel('Total Mass [ $M_{\odot}$ ]\n')
>>> plt.ylabel('Star Formation Rate [ $M_{\odot} / \text{yr}$ ]\n')
```



Let us get a list of primary subhalo IDs by loading the `GroupFirstSub` field from the FoF groups.

```
>>> GroupFirstSub = il.groupcat.loadHalos(basePath, 135, fields=['GroupFirstSub'])
>>> GroupFirstSub.dtype
dtype('uint32')
>>> GroupFirstSub.shape
(131727,)
```

Note: Return type

When a single field is requested, the default return is a raw numeric array and not a container type object.

For the 5 most massive central subhalos, let's load all their fields from the group catalog and print a gas fraction (gas mass over total baryonic mass) in the stellar half mass radius.

```
>>> ptNumGas = il.snapshot.partTypeNum('gas') # 0
>>> ptNumStars = il.snapshot.partTypeNum('stars') # 4
>>> for i in range(5):
>>>     all_fields = il.groupcat.loadSingle(basePath,135,subhaloID=GroupFirstSub[i])
>>>     gas_mass    = all_fields['SubhaloMassInHalfRadType'][ptNumGas]
>>>     stars_mass  = all_fields['SubhaloMassInHalfRadType'][ptNumStars]
>>>     frac = gas_mass / (gas_mass + stars_mass)
>>>     print GroupFirstSub[i], frac

0 0.0688846
608 0.0236937
1030 0.0638515
1396 0.00357705
1801 0.1222
```

Merger Trees

Let's download the full SubLink merger tree to play with (~8 GB).

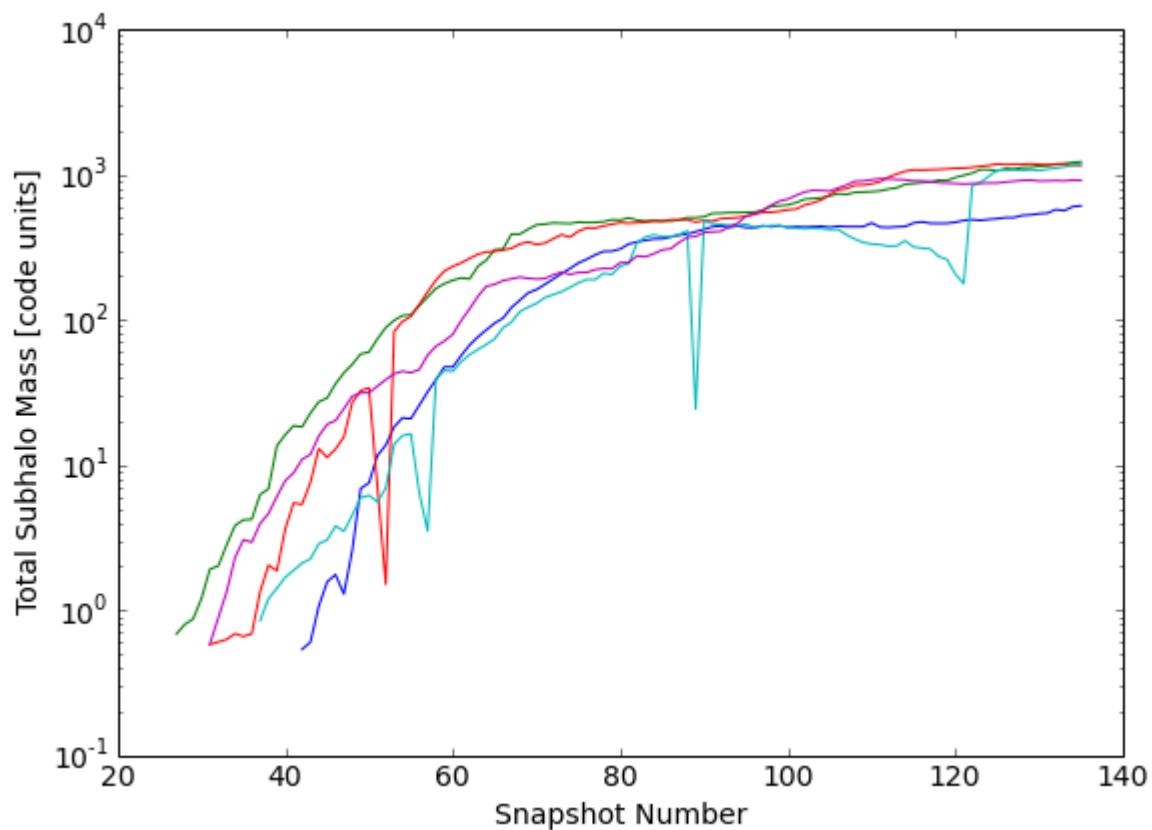
```
$ mkdir -p ~/Illustris-3/postprocessing/trees/SubLink
$ cd ~/Illustris-3/postprocessing/trees/SubLink/
$ wget -nd -nc -nv -e robots=off -l 1 -r -A hdf5 --content-disposition --header="API-Key:
544f44ca2820e2600e6fd85e1b553564" "http://www.tng-project.org/api/Illustris-3/files/subli
nk/?format=api"
```

Reading individual sub-trees from the full merger trees with 'offsets'

The example scripts make use of pre-computed offsets to accelerate the reading of subsets of the full merger tree files. For example, in order to load the main progenitor branch of the SubLink tree for a single subhalo. These offsets are stored in the group catalog files. Therefore, the `loadTree()` function requires that you have already downloaded the group catalogs corresponding to the snapshot of the subhalo (in addition to the tree files themselves).

For the 101st through 105th (indices 100 through 104 if 0-based) most massive primaries, extract the total mass, Subfind ID, and snapshot along the main progenitor branch. Plot the mass histories.

```
>>> fields = ['SubhaloMass', 'SubfindID', 'SnapNum']
>>> start = 100
>>> for i in range(start, start+5):
>>>     tree = il.sublink.loadTree(basePath, 135, GroupFirstSub[i], fields=fields, onlyMPB=True)
>>>     plt.plot(tree['SnapNum'], tree['SubhaloMass'], '-')
>>> plt.yscale('log')
>>> plt.xlabel('Snapshot Number')
>>> plt.ylabel('Total Subhalo Mass [code units]')
```



Note that the single-snapshot dips seen in the cyan and red curves can sometimes occur due to the 'subhalo switching problem'. The downward trend in mass followed by the sudden increase in the cyan is a signature of a merger. For details on both aspects of the trees, see the SubLink paper (<http://arxiv.org/abs/1502.01339>).

We include a semi-complex example of walking through the tree to determine the number of past mergers of a given subhalo, above some mass ratio threshold. Here, the mass ratio is defined as the ratio of the maximum past stellar mass of the two progenitors. For the same halos as above, count the number of major mergers (mass ratio $> 1/5$).

```
>>> ratio = 1.0/5.0
>>> # the following fields are required for the walk and the mass ratio analysis
>>> fields = ['SubhaloID', 'NextProgenitorID', 'MainLeafProgenitorID', 'FirstProgenitorID', 'SubhaloMassType']
>>> for i in range(start, start+5):
>>>     tree = il.sublink.loadTree(basePath, 135, GroupFirstSub[i], fields=fields)
>>>     numMergers = il.sublink.numMergers(tree, minMassRatio=ratio)
>>>     print GroupFirstSub[i], numMergers
```

```
9106 4
9137 2
9151 3
9170 5
9191 2
```

Snapshot Data

Let's download the full $z=0$ snapshot to play with (~20 GB).

```
$ mkdir ~/Illustris-3/output/snapdir_135
$ cd ~/Illustris-3/output/snapdir_135/
$ wget -nd -nc -nv -e robots=off -l 1 -r -A hdf5 --content-disposition --header="API-Key:
544f44ca2820e2600e6fd85e1b553564" "http://www.tng-project.org/api/Illustris-3/files/snaps
hot-135/?format=api"
```

Reading individual halos and subhalos from the snapshots with 'offsets'

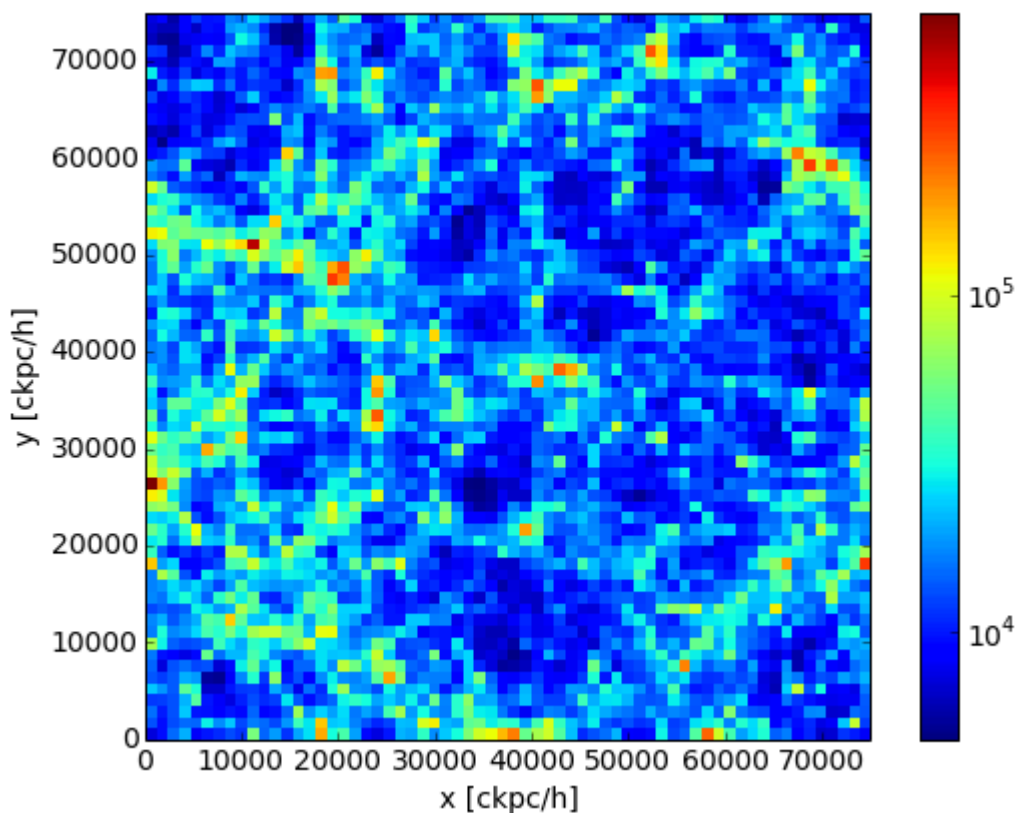
As with the merger trees, reading subsets of the snapshots makes use of pre-computed offsets. For example, in order to load only the particles belonging to a single FoF halo. For Illustris, these offsets are stored in the group catalog files. For TNG, these offsets are stored in separate "offset" files, as described above. As a result, the `loadSubhalo()` and `loadHalo()` functions require that you have already downloaded the corresponding offset files (in addition to the snapshot files themselves).

First, load the `Masses` of all the gas cells in the entire box, and calculate their mean, converting to log solar masses.

```
>>> import numpy as np
>>> fields = ['Masses']
>>> gas_mass = il.snapshot.loadSubset(basePath,135,'gas',fields=fields)
>>> print np.log10( np.mean(gas_mass,dtype='double')*1e10/0.704 )
7.92160161941
```

Next, load the spatial `Coordinates` of all the dark matter particles in the box, and make a quick image with a 2D histogram, projecting out the z-axis.

```
>>> import matplotlib as mpl
>>> dm_pos = il.snapshot.loadSubset(basePath,135,'dm',['Coordinates']);
>>> plt.hist2d(dm_pos[:,0], dm_pos[:,1], norm=mpl.colors.LogNorm(), bins=64);
>>> plt.xlim([0,75000])
>>> plt.ylim([0,75000])
>>> plt.xlabel('x [ckpc/h]')
>>> plt.ylabel('y [ckpc/h]')
```



Finally, load the star particles belonging to FoF halo ID **100** (all fields). Print the minimum and maximum of all positions for each axis to check we have loaded only stars in a localized region.

```
>>> stars = il.snapshot.loadHalo(basePath,135,100,'stars')
>>> stars.keys()
['count', u'GFM_Metals', u'SubfindVelDisp', u'GFM_InitialMass', u'Masses', u'Velocities',
u'Coordinates', u'Potential', u'SubfindHsm1', u'SubfindDensity', u'NumTracers', u'ParticleIDs', u'GFM_StellarFormationTime', u'GFM_StellarPhotometrics', u'GFM_Metallicity']

>>> for i in range(3):
>>>     print np.min(stars['Coordinates'][:,i]), np.max(stars['Coordinates'][:,i])
17993.7 19585.6
58373.0 59606.2
67596.5 68610.6
```

That's it! This should give you a good starting point, together with the reference below, to any aspect of the simulation data.

scida - scalable analysis for scientific big data

Interested in working with TNG data using a higher-level package, that automatically handles many complex tasks such as data loading, units, parallel calculations, and more?

Check out scida (<https://github.com/cbyrohl/scida>), which has built-in support for Illustris, TNG, EAGLE, SIMBA, and many other cosmological hydrodynamical simulations.

I/O Scripts Reference

Loading from the FoF and Subfind group catalogs:

The optional `fields` argument always accepts a string list/array of field names, which must agree (case-sensitive) to the available datasets in the group catalog. If it is not specified, all fields will be read and returned, which could be significantly slower.

```
illustris_python.groupcat.  
  
def loadSubhalos(basePath, snapNum, fields=None):  
    """ Load all subhalo information from the entire group catalog for one snapshot  
        (optionally restrict to a subset given by fields). """  
  
def loadHalos(basePath, snapNum, fields=None):  
    """ Load all halo information from the entire group catalog for one snapshot  
        (optionally restrict to a subset given by fields). """  
  
def loadHeader(basePath, snapNum):  
    """ Load the group catalog header. """  
  
def load(basePath, snapNum):  
    """ Load complete group catalog all at once. """  
  
def loadSingle(basePath, snapNum, haloID=-1, subhaloID=-1):  
    """ Return complete group catalog information for one halo or subhalo. """
```

Loading particle-level data from the snapshots:

The optional `fields` argument always accepts a string list/array of field names, which must agree (case-sensitive) to the available datasets *for that particle type* in the snapshot. If it is not specified, all fields will be read and returned, which could be significantly slower.

The `partType` argument may either be the particle type number, or one of the recognized string names, e.g. 'gas', 'stars', 'bhs', or 'dm'.

```
illustris_python.snapshot.  
  
def loadSubset(basePath, snapNum, partType, fields=None):  
    """ Load a subset of fields for all particles/cells of a given partType. """  
  
def loadSubhalo(basePath, snapNum, id, partType, fields=None):  
    """ Load all particles/cells of one type for a specific subhalo  
        (optionally restricted to a subset fields). """  
  
def loadHalo(basePath, snapNum, id, partType, fields=None):  
    """ Load all particles/cells of one type for a specific halo  
        (optionally restricted to a subset fields). """
```

Loading from the SubLink merger trees:

The optional `fields` argument always accepts a string list/array of field names, which must agree (case-sensitive) to the available datasets in the SubLink trees. If it is not specified, all fields will be read and returned, which could be significantly slower.

If `onlyMPB = True`, then only the main progenitor branch will be loaded (that is, only FirstProgenitor links will be followed).


```
illustris_python.sublink.
```

```
def loadTree(basePath, snapNum, id, fields=None, onlyMPB=False, onlyMDB=False):  
    """ Load portion of Sublink tree, for a given subhalo, in its existing flat format.  
        (optionally restricted to a subset fields). If onlyMPB, then only the 'main progenitor branch'  
        is loaded. If onlyMDB, then only the 'main descendant branch' is loaded, which is  
        a single  
        tree branch if and only if this subhalo lies on the MPB of its z=0 descendant, while if not,  
        then the return is the full descendant merger tree which contains as its first entry the z=0  
        descendant of this subhalo. """  
  
def numMergers(tree, minMassRatio=1e-10, massPartType='stars', index=0):  
    """ Calculate the number of mergers, along the main-progenitor branch, in this subtree (optionally above some mass ratio threshold). """
```

Loading from the LHaloTree merger trees:

The optional `fields` argument always accepts a string list/array of field names, which must agree (case-sensitive) to the available datasets in the LHaloTree trees. If it is not specified, all fields will be read and returned, which could be significantly slower.

If `onlyMPB = True`, then only the main progenitor branch will be loaded (that is, only FirstProgenitor links will be followed).



```
illustris_python.lhalotree.
```

```
def loadTree(basePath, snapNum, id, fields=None, onlyMPB=False):  
    """ Load portion of LHaloTree, for a given subhalo, re-arranging into a flat format.  
    """
```

General utility functions:

```
illustris_python.util.
```

```
def partTypeNum(partType):  
    """ Mapping between common names and numeric particle types. """
```

We gratefully acknowledge support from  (<https://www.mpcdf.mpg.de/>) and  (<http://www.rc.fas.harvard.edu/>)

(c) 2023 The TNG Collaboration. (You are logged in: Luuk Westerhoek).