# pkdgrav ssio.py Documentation

Last Update 7/17/19

Created by Ron Ballouz 11/25/15

## 1 OVERVIEW

The ssio.py library contains the Python functions read_SS() and write_SS() that allow the reading and writing of ss files in a Python script. In order to work correctly, ssio.py requires Python 3 and the NumPy library.

## 2 Setup

In order to use the functions in the ssio.py library, the file (located in the pkdgrav codebase under bin/scripts) must be copied to a local directory, or the path to the file can be placed in the PYTHONPATH environment variable. For example, in bash:

```
export PYTHONPATH=/path_to_pkdgrav/bin/scripts
```

where path_to_pkdgrav is the location of the pkdgrav codebase. Afterwards, one can import the ssio.py library from any Python script via:

```
import ssio
```

## 3 ssio.read_SS()

The read_SS() function allows a reduced or full pkdgrav output file to be read in by a Python script. read_SS() returns either a $14 \times N$ NumPy array (where $N$ is the number of particles) of the pkdgrav data file, or, if the optional argument unpack=True, a list of individual NumPy arrays. The function can optionally return the header information as well. As an argument, read_SS() requires the input file name, and optionally takes a string to specify whether the header information of the file should also be returned. The returned header is a list of three values: the timestamp, the number of particles, and the "magic number" ($-1$ for full output and $-10101$ for reduced). For example:

```
import ssio

#to get the header information and the data
header, data = ssio.read_SS('my_file.ss', 'yes')
#note this second argument should be any string that is not 'no'

#to just get the data
data = ssio.read_SS('my_file.ss')
```

If an error is encountered during reading, the `ssio.ReadError` exception is raised, with an explanatory message available. To handle the exception cleanly and print the error message, you could do the following:

```
import ssio

try:
    data = ssio.read_SS('my_file.ss')
except ssio.ReadError as msg:
    print('Read error:', msg)
    exit(1)
```

The data order is as follows: order number (integers from 0 to $N-1$); original order number (or original index); mass (doubles); radius (doubles); position (as a vector of doubles); velocity (doubles vector); spin (doubles vector); and color (integers). Here is sample usage with `unpack=True`:

```
import ssio
header, ord, oord, mass, rad, pos, vel, spin, c = \
    ssio.read_SS('my_file.ss', 'yes', unpack=True)
print(pos[:, 0])  # vector position of first particle
```

## 4 write_SS()

The `write_SS()` function allows the user to write a $14 \times N$ NumPy array to a reduced or full output `pkdgrav` file. As arguments, `write_SS()` takes a NumPy array and the name of the output file. Optionally, `write_SS()` can accept a time to write to the output file's header. `write_SS()` will write to a reduced output file if the filename given has a ".r" extension. For example, to read in a file (specified as a command-line argument), change all the particle colors to red, and then save to a reduced output file:

```
import sys
import ssio

file = sys.argv[1]
header, py_data = read_SS(file, 'yes')
py_data[13, :] = 2
ssio.write_SS(py_data, file + '_recolored.r', time = header[0])
```

If an error is encountered during writing, the `ssio.WriteError` exception is raised, with an explanatory message available, analogously to exception handling for `read_SS()`.