PEW PEW Documentation

Release 0.3.7a

pewpew

Dec 10, 2020

CONTENTS

PEW stands for Process Event-Wise. As this library does multiprocessing, this is PEW PEW.

- Open Source License: MIT license
- Documentation: https://pewpew.readthedocs.io.

The need for a multiprocess-based framework for event-wise processing arose in particle physics dataset whose atomic datatypes are very large. Large datatypes lead to long IO time which bottlenecks processing time, so having multiple IO processes and independent analysis processes is advantageous.

This framework is thus aimed at data management development and processing frameworks.

Contents:

CHAPTER

ONE

INSTALLATION

Until this package leaves alpha, it won't be posted to PyPI. For now, this can be installed with:

~~~ bash pip install git+https://github.com/kwierman/pewpew ~~~

Or, listing it in dependencies (but not in distutils) with:

~~~ git+https://github.com/kwierman/pewpew@master ~~~

Or, by cloning and installing:

~~~ bash git clone git+https://github.com/kwierman/pewpew cd pewpew make install # alternatively python setup.py install ~~~
~~~

CHAPTER

TWO

USAGE

The idea of this package is to subclass the base *pewpew.base.StreamElement* class and then instantiate analysis classes as processes and start each in turn.

An example is given here of a concrete implementation.

```
class MyProcess(StreamElement):
    def __init__(self, exit_flag, inqueue=None, outqueue=None, **kwargs):
        """ You must absolutely call the StreamElement constructor
        """
        super(MyProcess, self).__init__(exit_flag, inqueue=None,
                                        outqueue=None, **kwargs)
        self.myvar = kwargs.get("my_variable", "default_value")

    def process(self, data):
        """ Override this class (it's necessary) to process data
        """
        data['data']['my field'] = data['data']['my_input_field']*1e9
```

One may then link modules together and start the processes

```
global_exit_flag = exit_flag()

reader = Reader(global_exit_flag)
proc = MyProcess(global_exit_flag)
writer = Writer(global_exit_flag)
reader.set_output(proc)
writer.set_input(proc)

reader.start()
writer.start()
proc.start()


reader.join()
proc.join()
writer.join()
```

CHAPTER

THREE

PEWPEW

3.1 pewpew package

3.1.1 Subpackages

pewpew.celery package

Submodules

pewpew.celery.worker module

Module contents

pewpew.csv package

Submodules

pewpew.csv.reader module

class pewpew.csv.reader.**Reader**(*exit_flag=None*, *inqueue=None*, *outqueue=None*, ***kwargs*)

Bases: *pewpew.base.StreamElement*

converters = []

log = <Logger pewpew.csv.reader (WARNING)>

on_start()

Override this method to perform an action at the process' beginning of execution.

process(*data=None*)

Abstract method. Implement this for the primary action this process will take on *data*

data: list or dict or None

Input data to be acted on. Primary data generators can accept None as an input, and produce data.

dict or list: Data to be processed downstream.

pewpew.csv.writer module

class pewpew.csv.writer.**Writer**(*exit_flag=None*, *inqueue=None*, *outqueue=None*, ***kwargs*)

Bases: *pewpew.base.StreamElement*

property filename

log = <Logger pewpew.csv.writer (WARNING)>

on_completion()

Override this method to perform an action at the process' end of execution.

on_start()

Override this method to perform an action at the process' beginning of execution.

property path

process(*data*)

Abstract method. Implement this for the primary action this process will take on *data*

data: list or dict or None

Input data to be acted on. Primary data generators can accept None as an input, and produce data.

dict or list: Data to be processed downstream.

Module contents

Specific implementation written for HDF5 reading and writing with *pewpew*.

pewpew.examples package

Module contents

pewpew.hdf5 package

Submodules

pewpew.hdf5.reader module

pewpew.hdf5.writer module

Module contents

pewpew.json package

Submodules

pewpew.json.MetadataDump module

class pewpew.json.MetadataDump.**MetadataDump**(*exit_flag=None*, *inqueue=None*, *outqueue=None*, ***kwargs*)

Bases: *pewpew.base.StreamElement*

log = <Logger pewpew.json.metadata (WARNING)>

on_start()
Override this method to perform an action at the process' beginning of execution.

property path

process(*data*)
Abstract method. Implement this for the primary action this process will take on *data*

data: list or dict or None

Input data to be acted on. Primary data generators can accept None as an input, and produce data.

dict or list: Data to be processed downstream.

Module contents

3.1.2 Submodules

3.1.3 pewpew.base module

The base class for PEWPEW processing. See *Usage*.

class pewpew.base.**StreamElement**(*exit_flag=None*, *inqueue=None*, *outqueue=None*, ***kwargs*)
Bases: multiprocessing.context.Process

Subclass this abstract class for concrete implementation of pewpew processing

check_input_flags()
Checks to see if the inputs have exited or not. Useful as the exiting condition is the Queue is empty and the inputs have all finished.

bool: True if at least one input is OK.

event_loop(***kwargs*)

get_data(***kwargs*)

on_completion()
Override this method to perform an action at the process' end of execution.

on_input_completed(***kwargs*)

on_start()
Override this method to perform an action at the process' beginning of execution.

abstract process(*data*)
Abstract method. Implement this for the primary action this process will take on *data*

data: list or dict or None

Input data to be acted on. Primary data generators can accept None as an input, and produce data.

dict or list: Data to be processed downstream.

put_data(***kwargs*)

run()
Called by multiprocessing.Process. Executes main event loop for process.

set_input(*other*)
Add an input *StreamElement* to this one. Creates a Queue between StreamElements in the event there is not an existing one.

other: ***StreamElement*** An other Stream Element which will stream queued data into this one.

set_output(*other*)
Sets *self* as an input *StreamElement* to *other*. Creates a Queue between StreamElements in the event there is not an existing one.

other: ***StreamElement*** An other Stream Element which will stream queued data from this one.

signal_exit_on_failure()
Helper decorator which sets appropriate flags when exceptions occur in daughter processes.

valid_data(*data*)
Validates whether data is valid for the data stream.

data [list or dict] Input data which must be validated

bool : True if valid data

pewpew.base.**exit_flag**()
Convenience function for creating the exit flag data type instance.

3.1.4 pewpew.cli module

3.1.5 Module contents

CHAPTER

FOUR

CONTRIBUTIONS

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

INDEX

V

W