SpiNNUtils Documentation

Jan 29, 2020

Contents

These pages document the python code for the SpiNNUtils module which is part of the SpiNNaker Project (Combined_documentation).

CHAPTER 1

Contents

1.1 spinn_utilities package

1.1.1 Subpackages

1.1.1.1 spinn_utilities.configs package

1.1.1.1.1 Submodules

1.1.1.1.2 spinn_utilities.configs.camel_case_config_parser module

class spinn_utilities.configs.camel_case_config_parser.**CamelCaseConfigParser**(*defaults=None*, *none_marker='Non*

Bases: ConfigParser.RawConfigParser

get_bool(*section*, *option*)
Get the boolean value of an option.

Parameters

- **section** (*str*) – What section to get the option from.
- **option** (*str*) – What option to read.

Returns The option value.

Return type bool

get_float(*section*, *option*)
Get the float value of an option.

Parameters

- **section** (*str*) – What section to get the option from.
- **option** (*str*) – What option to read.

Returns The option value.

Return type float

get_int(*section*, *option*)
Get the integer value of an option.

Parameters

- **section** (*str*) – What section to get the option from.
- **option** (*str*) – What option to read.

Returns The option value

Return type int

get_str(*section*, *option*)
Get the string value of an option.

Parameters

- **section** (*str*) – What section to get the option from.
- **option** (*str*) – What option to read.

Returns The option value

Return type str or None

optionxform(*optionstr*)

read(*filenames*)

read_files

1.1.1.1.3 spinn_utilities.configs.case_sensitive_parser module

class spinn_utilities.configs.case_sensitive_parser.**CaseSensitiveParser**(*defaults=None*, *dict_type=<class 'collections.OrderedDict'>*, *allow_no_value=False*)

Bases: ConfigParser.RawConfigParser

optionxform(*optionstr*)

1.1.1.1.4 spinn_utilities.configs.no_config_found_exception module

exception spinn_utilities.configs.no_config_found_exception.**NoConfigFoundException**
Bases: exceptions.Exception

Throws when an existing Section has an extra config value

1.1.1.1.5 spinn_utilities.configs.unexpected_config_exception module

exception spinn_utilities.configs.unexpected_config_exception.**UnexpectedConfigException**
Bases: exceptions.Exception

Throws when an existing Section has an extra config value

1.1.1.1.6 Module contents

1.1.1.2 spinn_utilities.testing package

1.1.1.2.1 Submodules

1.1.1.2.2 spinn_utilities.testing.log_checker module

spinn_utilities.testing.log_checker.**assert_logs_contains**(*level_name*, *log_records*, *sub_message*)

spinn_utilities.testing.log_checker.**assert_logs_contains_once**(*level_name*, *log_records*, *message*)

spinn_utilities.testing.log_checker.**assert_logs_error_contains**(*log_records*, *sub_message*)

Checks it the log records contain an ERROR log with this sub message

Note: While this code does not depend on testfixtures you will need testfixtures to generate the input data

Parameters

- **log_records** – list of log records returned bu testfixtures.LogCapture
- **sub_message** – String which should be part of an ERROR log

Rasies AssertionError

spinn_utilities.testing.log_checker.**assert_logs_error_not_contains**(*log_records*, *sub_message*)

Checks it the log records do not contain an ERROR log with this sub message

Note: While this code does not depend on testfixtures you will need testfixtures to generate the input data

Parameters

- **log_records** – list of log records returned bu testfixtures.LogCapture
- **sub_message** – String which should be part of an ERROR log

Rasies AssertionError

spinn_utilities.testing.log_checker.**assert_logs_info_contains**(*log_records*, *sub_message*)

Checks it the log records contain an INFO log with this sub message

Note: While this code does not depend on testfixtures you will need testfixtures to generate the input data

Parameters

- **log_records** – list of log records returned bu testfixtures.LogCapture
- **sub_message** – String which should be part of an INFO log

Rasies AssertionError

spinn_utilities.testing.log_checker.**assert_logs_info_not_contains**(*log_records*, *sub_message*)

Checks it the log records do not contain an INFO log with this sub message

Note: While this code does not depend on testfixtures you will need testfixtures to generate the input data

Parameters

- **log_records** – list of log records returned bu testfixtures.LogCapture
- **sub_message** – String which should be part of an INFO log

Rasies AssertionError

spinn_utilities.testing.log_checker.**assert_logs_not_contains**(*level_name*, *log_records*, *sub_message*)

1.1.1.2.3 Module contents

1.1.2 Submodules

1.1.3 spinn_utilities.abstract_base module

class spinn_utilities.abstract_base.**AbstractBase**

Bases: type

Metaclass for defining Abstract Base Classes (AbstractBases).

Use this metaclass to create an AbstractBase. An AbstractBase can be subclassed directly, and then acts as a mix-in class.

This is a trimmed down version of ABC. Unlike ABC you can not register unrelated concrete classes.

spinn_utilities.abstract_base.**abstractmethod**(*funcobj*)

A decorator indicating abstract methods.

Requires that the metaclass is AbstractBase or derived from it. A class that has a metaclass derived from AbstractBase cannot be instantiated unless all of its abstract methods are overridden. The abstract methods can be called using any of the normal 'super' call mechanisms.

Usage:

@add_metaclass(AbstractBase) class C:

@abstractmethod def my_abstract_method(self, . . .): . . .

class spinn_utilities.abstract_base.**abstractproperty**

Bases: property

A decorator indicating abstract properties.

Requires that the metaclass is AbstractBase or derived from it. A class that has a metaclass derived from AbstractBase cannot be instantiated unless all of its abstract properties are overridden. The abstract properties can be called using any of the normal 'super' call mechanisms.

Usage:

#@add_metaclass(AbstractBase) class C:

@abstractproperty def my_abstract_property(self):

...

This defines a read-only property; you can also define a read-write abstract property using the 'long' form of property declaration:

#@add_metaclass(AbstractBase) class C:

def getx(self): ... def setx(self, value): ... x = abstractproperty(getx, setx)

1.1.4 spinn_utilities.conf_loader module

spinn_utilities.conf_loader.**check_config**(*config*, *cfg_file*, *validation_config=None*, *default_config=None*)

Checks the config read up to this point to see if it is outdated

Once one difference is found a full reports is generated and an error

raised

Any section specific list as Dead will cause a error

Any section in the defaults should not have extra values. It will never have less as the defaults are in the config

Errors on any values listed as PreviousValues. These are specific values in specific options no longer supported For example old algorithm names

Parameters

- **config** – Config as read in up to this point
- **cfg_file** – Path of last file read in
- **validation_config** – Path containing the validation rules
- **default_configs** – List of Paths to defaults

spinn_utilities.conf_loader.**install_cfg_and_IOError**(*filename*, *defaults*, *config_locations*)

Installs a local config based on the tamplates and thorws an Error

This method is called when no user config is found.

It will create a file in the users home directory based on the defaults.

Then it prints a helpful messages and thros and error with the same message

Parameters

- **filename** (*str*) – Name under which to save the new config file
- **defaults** (*List[str]*) – List of full paths to the default config files. Each of which MUST have an associated template file with exactly the same path plus .template
- **config_locations** – List of paths the user configs where looked for, Onlty used for the message

raise NoConfigFoundException: Always raised

spinn_utilities.conf_loader.**load_config**(*filename*, *defaults*, *config_parsers=None*, *validation_cfg=None*)

Load the configuration

Parameters **config_parsers** (*list of (str, ConfigParser)*) – The parsers to parse the config with, as a list of (section name, parser); config will only be parsed if the section_name is found in the configuration files already loaded

spinn_utilities.conf_loader.**logging_parser**(*config*)
Create the root logger with the given level.

Create filters based on logging levels

spinn_utilities.conf_loader.**outdated_config**(*cfg_file*, *validation_config*, *default_configs*)
Prints why a config file is outdated and raises an exception

Reads a config file by itself (Without others)

Reports errors in this config based on the validation_config and the defaults_configs

Reports any values listed as PreviousValues. These are specific values in specific options no longer supported For example old algorithm names

Checks all sections not defined as UserSections (Default Machine) IE ones the user is expected to change

Any section specific list as Dead will be reported

Any section in the default config is compared reporting any unexpected values reporting the smaller of values non default or values same as default

Any other section is ignored as assumed being used by an extenstion

Parameters

- **cfg_file** – Path to be checked
- **validation_config** – Path containing the validation rules
- **default_configs** – List of Paths to defaults

Returns

spinn_utilities.conf_loader.**read_a_config**(*config*, *cfg_file*, *validation_config=None*, *default_config=None*)
Reads in a config file and then directly its machine_spec_file

Parameters

- **config** – config to do the reading
- **cfg_file** – path to file which should be read in

Returns list of files read including and machine_spec_files

1.1.5 spinn_utilities.executable_finder module

class spinn_utilities.executable_finder.**ExecutableFinder**(*binary_search_paths*)
Bases: object

Manages a set of folders in which to search for binaries, and allows for binaries to be discovered within this path

Parameters **binary_search_paths** (*iterable of str*) – The initial set of folders to search for binaries.

add_path(*path*)
Adds a path to the set of folders to be searched. The path is added to the end of the list, so it is searched after all the paths currently in the list.

Parameters **path** (*str*) – The path to add

Returns Nothing is returned

Return type None

binary_paths

get_executable_path(*executable_name*)
Finds an executable within the set of folders. The set of folders is searched sequentially and the first match is returned.

Parameters **executable_name** (*str*) – The name of the executable to find

Returns The full path of the discovered executable, or None if no executable was found in the set of folders

Return type str

1.1.6 spinn_utilities.helpful_functions module

spinn_utilities.helpful_functions.**get_valid_components**(*module*, *terminator*)
Get possible components

Parameters

- **module** –
- **terminator** –

Return type dict

spinn_utilities.helpful_functions.**set_up_output_application_data_specifics**(*where_to_write_applic*
max_application_binar
app_id,
n_calls_to_run,
this_run_time_string)

Parameters

- **where_to_write_application_data_files** – the location where all app data is by default written to
- **max_application_binaries_kept** – The max number of report folders to keep active at any one time
- **app_id** – the id used for identifying the simulation on the SpiNNaker machine
- **n_calls_to_run** – the counter of how many times run has been called.
- **this_run_time_string** – the time stamp string for this run

Returns the run folder for this simulation to hold app data

spinn_utilities.helpful_functions.**set_up_report_specifics**(*default_report_file_path*,
max_reports_kept,
app_id, *n_calls_to_run*,
this_run_time_string=None)

Parameters

- **default_report_file_path** – The location where all reports reside
- **max_reports_kept** – The max number of report folders to keep active at any one time
- **app_id** – the id used for identifying the simulation on the SpiNNaker machine

- **n_calls_to_run** – the counter of how many times run has been called.
- **this_run_time_string** – holder for the timestamp for future runs

Returns The folder for this run, the time_stamp

spinn_utilities.helpful_functions.**write_finished_file**(*app_data_runtime_folder*, *report_default_directory*)

1.1.7 spinn_utilities.log module

class spinn_utilities.log.**ConfiguredFilter**(*conf*)
Bases: object

filter(*record*)
Get the level for the deepest parent, and filter appropriately.

class spinn_utilities.log.**ConfiguredFormatter**(*conf*)
Bases: logging.Formatter

static construct_logging_parents(*conf*)
Create a dictionary of module names and logging levels.

static deepest_parent(*parents*, *child*)
Greediest match between child and parent.

static level_of_deepest_parent(*parents*, *child*)
The logging level of the greediest match between child and parent.

1.1.8 spinn_utilities.logger_utils module

spinn_utilities.logger_utils.**reset**()

spinn_utilities.logger_utils.**warn_once**(*logger*, *msg*)

1.1.9 spinn_utilities.ordered_set module

class spinn_utilities.ordered_set.**OrderedSet**(*iterable=None*)
Bases: _abcoll.MutableSet

add(*key*)
Add an element.

discard(*key*)
Remove an element. Do not raise an exception if absent.

pop(*last=True*)
Return the popped value. Raise KeyError if empty.

update(*iterable*)

1.1.10 spinn_utilities.overrides module

class spinn_utilities.overrides.**overrides**(*super_class_method*, *extend_doc=True*, *additional_arguments=None*)
Bases: object

A decorator for indicating that a method overrides another method in a super class. This checks that the method does actually exist, copies the doc-string for the method, and enforces that the method overridden is specified, making maintenance easier.

Parameters

- **super_class_method** – The method to override in the superclass
- **extend_doc** – True the method doc string should be appended to the super-method doc string, False if the documentation should be set to the super-method doc string only if there isn't a doc string already
- **additional_arguments** – Additional arguments taken by the subclass method over the superclass method e.g. that are to be injected

1.1.11 spinn_utilities.package_loader module

spinn_utilities.package_loader.**all_modules**(*directory*, *prefix*, *remove_pyc_files=False*)

List all the python files found in this directory giving then the prefix

Any file that ends in either .py or .pyc is assume a python module and added to the result set

Parameters

- **directory** – path to check for python files
- **prefix** – package prefix top add to the file name

Returns set of python package names

spinn_utilities.package_loader.**load_module**(*name*, *remove_pyc_files=False*, *exclusions=[]*, *gather_errors=True*)

Loads this modules and all its children

Parameters

- **name** – name of the modules
- **remove_pyc_files** – True if .pyc files should be deleted
- **exclusions** – a list of modules to exclude
- **gather_errors** – True if errors should be gathered, False to report on first error

spinn_utilities.package_loader.**load_modules**(*directory*, *prefix*, *remove_pyc_files=False*, *exclusions=[]*, *gather_errors=True*)

Loads all the python files found in this directory giving then the prefix

Any file that ends in either .py or .pyc is assume a python module and added to the result set

Parameters

- **directory** – path to check for python files
- **prefix** – package prefix top add to the file name
- **remove_pyc_files** – True if .pyc files should be deleted
- **exclusions** – a list of modules to exclude
- **gather_errors** – True if errors should be gathered, False to report on first error

1.1.12 spinn_utilities.progress_bar module

class spinn_utilities.progress_bar.**ProgressBar**(*total_number_of_things_to_do*, *string_describing_what_being_progressed*, *step_character='='*, *end_character='|'*)

Bases: object

Progress bar for telling the user where a task is up to

MAX_LENGTH_IN_CHARS = 60

end()

Close the progress bar, updating whatever is left if needed

Return type None

over(*collection*)

Simple wrapper for the cases where the progress bar is being used to show progress through the iteration over a single collection. The progress bar should have been initialised to the size of the collection being iterated over.

Parameters **collection** – The base collection (any iterable) being iterated over

Returns An iterable. Expected to be directly used in a for.

update(*amount_to_add=1*)

Update the progress bar by a given amount

Parameters **amount_to_add** –

Return type None

1.1.13 spinn_utilities.safe_eval module

class spinn_utilities.safe_eval.**SafeEval**(**args*, ***kwargs*)

Bases: object

This provides expression evaluation capabilities while allowing the set of symbols exposed to the expression to be strictly controlled.

Sample of use:

```
>>> import math
>>> def evil_func(x):
        print "HAHA!"
        return x/0.0
```

```
>>> eval_safely = SafeEval(math)
>>> eval_safely.eval("math.sqrt(x)", x=1.23)
1.1090536506409416
>>> eval_safely.eval("evil_func(1.23)")
Traceback (most recent call last):
  ...
NameError: name 'evil_func' is not defined
```

Parameters

- **args** – The symbols to use to populate the global reference table. Note that all of these symbols must support the __name__ property, but that includes any function, method of

an object, or module. If you want to make an object available by anything other than its inherent name, define it in the eval() call.

- **kwargs** – Define the symbols with explicit names. Needed because some symbols (e.g., constants in numpy) do not have names that we can otherwise look up easily.

eval(*expression*, ***kwargs*)
Evaluate an expression and return the result.

Parameters

- **expression** (*str*) – The expression to evaluate
- **kwargs** – The extra symbol bindings to use for this evaluation. This is useful for passing in particular parameters to an individual evaluation run.

Returns The expression result

1.1.14 spinn_utilities.socket_address module

class spinn_utilities.socket_address.**SocketAddress**(*notify_host_name*, *notify_port_no*, *listen_port*)

Bases: object

Data holder for a socket interface for notification protocol.

listen_port
The port to listen to for responses

notify_host_name
The notify host name

notify_port_no
The notify port no

1.1.15 spinn_utilities.timer module

class spinn_utilities.timer.**Timer**
Bases: object

A timer used for performance measurements

measured_interval

start_timing()

take_sample()

1.1.16 Module contents

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

Index

I

L

M

N

O

P

R

S

T

U

W