Aspectlib

Aspectlib is a python library that can be used for AOP. Aspectlib also provides aspects that can help with debugging, logging, and testing.

The join points where an advice can run include the execution of: builtin or user-defined functions, class methods, constructors, magic methods, etc…

In order to define an aspect, you need to use an aspectlib decorator on a generator function that will yield an advice. In python, a generator function behaves kind of like an iterator. It uses the keyword **yield**, instead of **return**, in order to pass values to the place where it was called. Here’s an example of a generator function that provides all positive, even numbers, smaller than a specified upper bound:



This is an example of a performance monitoring aspect:



Possible values for an advice:

* *aspectlib.Proceed* / *None*: calls the advised method with the original parameters that were passed in the original function call. Yields the function’s return value, if an exception is not raised. Can be used multiple times, inside the same aspect.
* *aspectlib.Proceed(\*args, \*\*kwargs)*: same as above, but the original parameters that were passed in the original function call can be altered.
* *aspectlib*.*Return*: returns None, closes the generator and the flow of control is returned to where the advised function was called from.
* *aspectlib*.*Return(value)*: same as above, but the return value can be specified.
* *raise exception*: raises an exception.

Optionally, the advised function can be accessed inside the aspect, by specifying a ***bind=True*** parameter to the ***aspectlib.Aspect*** decorator. This feature should not be used for directly calling the function, but may be used for accessing information about the function (ex: the function’s name).



Weaving is done at manually, at run-time, and can be rolled back. This is done through the ***aspectlib.weave()*** function, which returns an ***aspectlib.Rollback*** object.



The ***aspectlib.weave()*** function takes the following parameters:

* ***target***: string, class, function. If the value is a class, all of its methods will be weaved.
* ***aspects:*** aspect or list of aspects
* ***subclasses***: optional bool flag, available only if ***target*** is a class. If ***True***, the subclasses of ***target*** will also be weaved. Defaults to ***True***.
* ***aliases***: optional bool flag. If ***True***, all aliases of ***target*** are also weaved. Defaults to ***True***.
* ***lazy***: optional bool flag, available only if ***target*** is a class. If ***True***, only ***target***’s constructor is weaved, all other methods are weaved after the constructor is called. Defaults to ***False***.
* ***methods***: optional parameter, can be a list of functions, a regex, or a string. Only available if ***target*** is a class. Used to specify what class methods should be weaved. Defaults to ***aspectlib.NORMAL\_METHODS***, a regex that matches all methods that are not magic methods (all methods that do not both start and end with double underscores).

References:

* <https://python-aspectlib.readthedocs.io/en/latest/>
* <https://docs.python.org/>
* <https://wiki.python.org/>