

---

# **Gather**

***Release 17.4***

**Jan 25, 2018**



---

## Contents

---

<b>1 Overview</b>	<b>1</b>
<b>2 API</b>	<b>3</b>
<b>Python Module Index</b>	<b>5</b>



# CHAPTER 1

---

## Overview

---

The gather package allows easily gathering up plugins. The basic class defined is the `Collector`.

```
import gather
THINGS = gather.Collector()
```

In order to register an object as a plugin, we merely decorate it:

```
@THINGS.register()
def some_function():
    pass
```

Note that the decorator always returns the function – `some_function` remains perfectly usable.

Finding all the things collected is simple:

```
registered = THINGS.collect()
```

The return value is a dictionary, mapping names to registered objects.

If an alternative name is needed for registration, one can be provided explicitly:

```
@THINGS.register(name='register_as_this_name')
def generic():
    pass
```

When registering functions that expect an argument list, like `sys.argv`, the `run` function can be used to run them directly:

```
gather.run(
    commands=THINGS.collect(),
    version='1.2.3',
    argv=sys.argv[1:],
    output=sys.stdout
)
```

It is important to remember that all the gathering depends on registering an entry point in the `setup.py`:

```
entry_points={
    'gather': [
        "gather=ROOT_PACKAGE",
    ]
}
```

Putting the package name there is enough – `gather` will automatically collect from any sub-modules, recursing any number of levels. This is also enough to register it for any `gather`-using plugins.

### Gather – Collect all your plugins

Gather allows a way to register plugins. It features the ability to register the plugins from any module, in any package, in any distribution. A given module can register plugins of multiple types.

In order to have anything registered from a package, it needs to declare that it supports `gather` in its `setup.py`:

```
entry_points={
    'gather': [
        "dummy=ROOT_PACKAGE:dummy",
    ]
}
```

The `ROOT_PACKAGE` should point to the Python name of the package: i.e., what users are expected to `import` at the top-level.

Note that while having special facilities to run functions as subcommands, Gather can be used to collect anything.

**class** `gather.api.Collector` (*name=None, depth=1*)

A plugin collector.

A collector allows to *register* functions or classes by modules, and *collect*-ing them when they need to be used.

**static all** (*registry, effective\_name, object*)

Assign all of the possible options.

Collect all registered items into a set, and assign that set to a name. Note that even if only one item is assigned to a name, that name will be assigned to a set of length 1.

**collect** (*strategy=<function \_one\_of>*)

Collect all registered functions or classes.

Returns a dictionary mapping names to registered elements.

**static exactly\_one** (*registry, effective\_name, object*)

Raise an error on colliding registration.

If more than one item is registered to the same name, raise a `GatherCollisionError`.

**static one\_of** (*\_registry, \_effective\_name, object*)

Assign one of the possible options.

When given as a collection strategy to `collect`, this will assign one of the options to a name in case more than one item is registered to the same name.

This is the default.

**register** (*name=None, transform=<function <lambda>>*)

Register a class or function

**Parameters**

- **name** (*str*) – optional. Name to register the class or function as. (default is name of object)
- **transform** (*callable*) – optional. A one-argument function. Will be called, and the return value used in collection. Default is identity function

This is meant to be used as a decorator:

```
@COLLECTOR.register()
def specific_subcommand(args):
    pass

@COLLECTOR.register(name='another_specific_name')
def main(args):
    pass
```

`gather.api.run` (*argv, commands, version, output*)

Run the specified subcommand.

**Parameters**

- **argv** (*list of str*) – Arguments to be processed
- **commands** (*mapping of str to callables*) – Commands (usually collected by a Collector)
- **version** (*str*) – Version to display if `--version` is asked
- **output** (*file*) – Where to write output to

**class** `gather.api.Wrapper` (*original, extra*)

Add extra data to an object

**classmethod** `glue` (*extra*)

Glue extra data to an object

**Parameters** `extra` – what to add

**Returns** function of one argument that returns a `Wrapped`

**Return type** `callable`

This method is useful mainly as the `transform` parameter of a `register` call.



**g**

`gather.api`, 3



## A

`all()` (`gather.api.Collector` static method), 3

## C

`collect()` (`gather.api.Collector` method), 3

`Collector` (class in `gather.api`), 3

## E

`exactly_one()` (`gather.api.Collector` static method), 3

## G

`gather.api` (module), 3

`glue()` (`gather.api Wrapper` class method), 4

## O

`one_of()` (`gather.api.Collector` static method), 3

## R

`register()` (`gather.api.Collector` method), 4

`run()` (in module `gather.api`), 4

## W

`Wrapper` (class in `gather.api`), 4