

8.12. `reprlib` — Alternate `repr()` implementation

Source code: [Lib/reprlib.py](#)

The `reprlib` module provides a means for producing object representations with limits on the size of the resulting strings. This is used in the Python debugger and may be useful in other contexts as well.

This module provides a class, an instance, and a function:

`class reprlib.Repr`

Class which provides formatting services useful in implementing functions similar to the built-in `repr()`; size limits for different object types are added to avoid the generation of representations which are excessively long.

`reprlib.aRepr`

This is an instance of `Repr` which is used to provide the `repr()` function described below. Changing the attributes of this object will affect the size limits used by `repr()` and the Python debugger.

`reprlib.repr(obj)`

This is the `repr()` method of `aRepr`. It returns a string similar to that returned by the built-in function of the same name, but with limits on most sizes.

In addition to size-limiting tools, the module also provides a decorator for detecting recursive calls to `__repr__()` and substituting a placeholder string instead.

`@reprlib.recursive_repr(fillvalue="...")`

Decorator for `__repr__()` methods to detect recursive calls within the same thread. If a recursive call is made, the `fillvalue` is returned, otherwise, the usual `__repr__()` call is made. For example:

```
>>> class MyList(list):
...     @recursive_repr()
...     def __repr__(self):
...         return '<' + '|'.join(map(repr, self)) + '>'
...
>>> m = MyList('abc')
>>> m.append(m)
>>> m.append('x')
>>> print(m)
<'a'|'b'|'c'|...|'x'>
```

New in version 3.2.

8.12.1. Repr Objects

[Repr](#) instances provide several attributes which can be used to provide size limits for the representations of different object types, and methods which format specific object types.

Repr.maxlevel

Depth limit on the creation of recursive representations. The default is 6.

Repr.maxdict

Repr.maxlist

Repr.maxtuple

Repr.maxset

Repr.maxfrozenset

Repr.maxdeque

Repr.maxarray

Limits on the number of entries represented for the named object type. The default is 4 for [maxdict](#), 5 for [maxarray](#), and 6 for the others.

Repr.maxlong

Maximum number of characters in the representation for an integer. Digits are dropped from the middle. The default is 40.

Repr.maxstring

Limit on the number of characters in the representation of the string. Note that the “normal” representation of the string is used as the character source: if escape sequences are needed in the representation, these may be mangled when the representation is shortened. The default is 30.

Repr.maxother

This limit is used to control the size of object types for which no specific formatting method is available on the [Repr](#) object. It is applied in a similar manner as [maxstring](#). The default is 20.

Repr.repr(obj)

The equivalent to the built-in [repr\(\)](#) that uses the formatting imposed by the instance.

Repr.repr1(obj, level)

Recursive implementation used by [repr\(\)](#). This uses the type of *obj* to determine which formatting method to call, passing it *obj* and *level*. The type-specific

methods should call `repr1()` to perform recursive formatting, with `level - 1` for the value of `level` in the recursive call.

`Repr.repr_TYPE(obj, level)`

Formatting methods for specific types are implemented as methods with a name based on the type name. In the method name, **TYPE** is replaced by `'_'.join(type(obj).__name__.split())`. Dispatch to these methods is handled by `repr1()`. Type-specific methods which need to recursively format a value should call `self.repr1(subobj, level - 1)`.

8.12.2. Subclassing Repr Objects

The use of dynamic dispatching by `Repr.repr1()` allows subclasses of `Repr` to add support for additional built-in object types or to modify the handling of types already supported. This example shows how special support for file objects could be added:

```
import reprlib
import sys

class MyRepr(reprlib.Repr):

    def repr_TextIOWrapper(self, obj, level):
        if obj.name in {'<stdin>', '<stdout>', '<stderr>'}:
            return obj.name
        return repr(obj)

aRepr = MyRepr()
print(aRepr.repr(sys.stdin))          # prints '<stdin>'
```