:mod:`json` --- JSON encoder and decoder

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 1); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 4)

Unknown directive type "module".

```
.. module:: json
   :synopsis: Encode and decode the JSON format.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 7)

Unknown directive type "moduleauthor".

```
.. moduleauthor:: Bob Ippolito <bob@redivi.com>
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 8)

Unknown directive type "sectionauthor".

```
.. sectionauthor:: Bob Ippolito <bob@redivi.com>
```

Source code: :source:`Lib/json/__init__.py`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 10); *backlink*

Unknown interpreted text role "source".

JSON (JavaScript Object Notation), specified by RFC 7159 (which obsoletes RFC 4627) and by ECMA-404, is a lightweight data interchange format inspired by JavaScript object literal syntax (although it is not a strict subset of JavaScript [1]).

:mod:`json` exposes an API familiar to users of the standard library :mod:`marshal` and :mod:`pickle` modules.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 21); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 21); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 21); *backlink*

Unknown interpreted text role "mod".

Encoding basic Python object hierarchies:

```
>>> import json
>>> json.dumps(['foo', {'bar': ('baz', None, 1.0, 2)}])
'["foo", {"bar": ["baz", null, 1.0, 2]}]'
>>> print(json.dumps("\"foo\bar"))
"\"foo\bar"
>>> print(json.dumps('\u1234'))
"\u1234"
>>> print(json.dumps('\\'))
"\\"
>>> print(json.dumps({"c": 0, "b": 0, "a": 0}, sort_keys=True))
{"a": 0, "b": 0, "c": 0}
>>> from io import StringIO
>>> io = StringIO()
>>> json.dump(['streaming API'], io)
>>> io.getvalue()
'["streaming API"]'
```

Compact encoding:

```
>>> import json
>>> json.dumps([1, 2, 3, {'4': 5, '6': 7}], separators=(',', ':'))
'[1,2,3,{"4":5,"6":7}]'
```

Pretty printing:

```
>>> import json
>>> print(json.dumps({'4': 5, '6': 7}, sort_keys=True, indent=4))
{
    "4": 5,
    "6": 7
}
```

Decoding JSON:

```
>>> import json
>>> json.loads('["foo", {"bar":["baz", null, 1.0, 2]}]')
```

```
['foo', {'bar': ['baz', None, 1.0, 2]}]
>>> json.loads('"\\"foo\\bar"')
'"foo\x08ar'
>>> from io import StringIO
>>> io = StringIO('["streaming API"]')
>>> json.load(io)
['streaming API']
```

Specializing JSON object decoding:

```
>>> import json
>>> def as_complex(dct):
...     if '__complex__' in dct:
...         return complex(dct['real'], dct['imag'])
...     return dct
...
>>> json.loads('{"__complex__": true, "real": 1, "imag": 2}',
...     object_hook=as_complex)
(1+2j)
>>> import decimal
>>> json.loads('1.1', parse_float=decimal.Decimal)
Decimal('1.1')
```

Extending :class:`JSONEncoder`:

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)json.rst`, line 85); *backlink***
>
> Unknown interpreted text role "class".

```
>>> import json
>>> class ComplexEncoder(json.JSONEncoder):
...     def default(self, obj):
...         if isinstance(obj, complex):
...             return [obj.real, obj.imag]
...         # Let the base class default method raise the TypeError
...         return json.JSONEncoder.default(self, obj)
...
>>> json.dumps(2 + 1j, cls=ComplexEncoder)
'[2.0, 1.0]'
>>> ComplexEncoder().encode(2 + 1j)
'[2.0, 1.0]'
>>> list(ComplexEncoder().iterencode(2 + 1j))
['[2.0', ', 1.0', ']']
```

Using :mod:`json.tool` from the shell to validate and pretty-print:

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)json.rst`, line 103); *backlink***
>
> Unknown interpreted text role "mod".

```
$ echo '{"json":"obj"}' | python -m json.tool
{
    "json": "obj"
}
$ echo '{1.2:3.4}' | python -m json.tool
Expecting property name enclosed in double quotes: line 1 column 2 (char 1)
```

See :ref:`json-commandline` for detailed documentation.

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)json.rst`, line 114); *backlink***
>
> Unknown interpreted text role "ref".

> **Note**
>
> JSON is a subset of YAML 1.2. The JSON produced by this module's default settings (in particular, the default *separators* value) is also a subset of YAML 1.0 and 1.1. This module can thus also be used as a YAML serializer.

> **Note**
>
> This module's encoders and decoders preserve input and output order by default. Order is only lost if the underlying containers are unordered.
>
> Prior to Python 3.7, :class:`dict` was not guaranteed to be ordered, so inputs and outputs were typically scrambled unless :class:`collections.OrderedDict` was specifically requested. Starting with Python 3.7, the regular :class:`dict` became order preserving, so it is no longer necessary to specify :class:`collections.OrderedDict` for JSON generation and parsing.
>
> > **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)json.rst`, line 128); *backlink***
> >
> > Unknown interpreted text role "class".
>
> > **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)json.rst`, line 128); *backlink***
> >
> > Unknown interpreted text role "class".
>
> > **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)json.rst`, line 128); *backlink***
> >
> > Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)json.rst, line 128); *backlink*

Unknown interpreted text role "class".

Basic Usage

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)json.rst, line 139)

Unknown directive type "function".

```
.. function:: dump(obj, fp, *, skipkeys=False, ensure_ascii=True, \
                   check_circular=True, allow_nan=True, cls=None, \
                   indent=None, separators=None, default=None, \
                   sort_keys=False, **kw)

   Serialize *obj* as a JSON formatted stream to *fp* (a ``.write()``-supporting
   :term:`file-like object`) using this :ref:`conversion table
   <py-to-json-table>`.

   If *skipkeys* is true (default: ``False``), then dict keys that are not
   of a basic type (:class:`str`, :class:`int`, :class:`float`, :class:`bool`,
   ``None``) will be skipped instead of raising a :exc:`TypeError`.

   The :mod:`json` module always produces :class:`str` objects, not
   :class:`bytes` objects. Therefore, ``fp.write()`` must support :class:`str`
   input.

   If *ensure_ascii* is true (the default), the output is guaranteed to
   have all incoming non-ASCII characters escaped.  If *ensure_ascii* is
   false, these characters will be output as-is.

   If *check_circular* is false (default: ``True``), then the circular
   reference check for container types will be skipped and a circular reference
   will result in a :exc:`RecursionError` (or worse).

   If *allow_nan* is false (default: ``True``), then it will be a
   :exc:`ValueError` to serialize out of range :class:`float` values (``nan``,
   ``inf``, ``-inf``) in strict compliance of the JSON specification.
   If *allow_nan* is true, their JavaScript equivalents (``NaN``,
   ``Infinity``, ``-Infinity``) will be used.

   If *indent* is a non-negative integer or string, then JSON array elements and
   object members will be pretty-printed with that indent level.  An indent level
   of 0, negative, or ``""`` will only insert newlines.  ``None`` (the default)
   selects the most compact representation. Using a positive integer indent
   indents that many spaces per level.  If *indent* is a string (such as ``"\t"``),
   that string is used to indent each level.

   .. versionchanged:: 3.2
      Allow strings for *indent* in addition to integers.

   If specified, *separators* should be an ``(item_separator, key_separator)``
   tuple.  The default is ``(', ', ': ')`` if *indent* is ``None`` and
   ``(',', ': ')`` otherwise.  To get the most compact JSON representation,
   you should specify ``(',', ':')`` to eliminate whitespace.

   .. versionchanged:: 3.4
      Use ``(',', ': ')`` as default if *indent* is not ``None``.

   If specified, *default* should be a function that gets called for objects that
   can't otherwise be serialized.  It should return a JSON encodable version of
   the object or raise a :exc:`TypeError`.  If not specified, :exc:`TypeError`
   is raised.

   If *sort_keys* is true (default: ``False``), then the output of
   dictionaries will be sorted by key.

   To use a custom :class:`JSONEncoder` subclass (e.g. one that overrides the
   :meth:`default` method to serialize additional types), specify it with the
   *cls* kwarg; otherwise :class:`JSONEncoder` is used.

   .. versionchanged:: 3.6
      All optional parameters are now :ref:`keyword-only <keyword-only_parameter>`.

   .. note::

      Unlike :mod:`pickle` and :mod:`marshal`, JSON is not a framed protocol,
      so trying to serialize multiple objects with repeated calls to
      :func:`dump` using the same *fp* will result in an invalid JSON file.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)json.rst, line 209)

Unknown directive type "function".

```
.. function:: dumps(obj, *, skipkeys=False, ensure_ascii=True, \
                    check_circular=True, allow_nan=True, cls=None, \
                    indent=None, separators=None, default=None, \
                    sort_keys=False, **kw)

   Serialize *obj* to a JSON formatted :class:`str` using this :ref:`conversion
   table <py-to-json-table>`.  The arguments have the same meaning as in
   :func:`dump`.

   .. note::

      Keys in key/value pairs of JSON are always of the type :class:`str`. When
      a dictionary is converted into JSON, all the keys of the dictionary are
      coerced to strings. As a result of this, if a dictionary is converted
      into JSON and then back into a dictionary, the dictionary may not equal
      the original one. That is, ``loads(dumps(x)) != x`` if x has non-string
      keys.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 227)

Unknown directive type "function".

```
.. function:: load(fp, *, cls=None, object_hook=None, parse_float=None, parse_int=None, parse_constant=None, object_pa

   Deserialize *fp* (a ``.read()``-supporting :term:`text file` or
   :term:`binary file` containing a JSON document) to a Python object using
   this :ref:`conversion table <json-to-py-table>`.

   *object_hook* is an optional function that will be called with the result of
   any object literal decoded (a :class:`dict`).  The return value of
   *object_hook* will be used instead of the :class:`dict`.  This feature can be used
   to implement custom decoders (e.g. `JSON-RPC <http://www.jsonrpc.org>`_
   class hinting).

   *object_pairs_hook* is an optional function that will be called with the
   result of any object literal decoded with an ordered list of pairs.  The
   return value of *object_pairs_hook* will be used instead of the
   :class:`dict`.  This feature can be used to implement custom decoders.
   If *object_hook* is also defined, the *object_pairs_hook* takes priority.

   .. versionchanged:: 3.1
      Added support for *object_pairs_hook*.

   *parse_float*, if specified, will be called with the string of every JSON
   float to be decoded.  By default, this is equivalent to ``float(num_str)``.
   This can be used to use another datatype or parser for JSON floats
   (e.g. :class:`decimal.Decimal`).

   *parse_int*, if specified, will be called with the string of every JSON int
   to be decoded.  By default, this is equivalent to ``int(num_str)``.  This can
   be used to use another datatype or parser for JSON integers
   (e.g. :class:`float`).

   *parse_constant*, if specified, will be called with one of the following
   strings: ``'-Infinity'``, ``'Infinity'``, ``'NaN'``.
   This can be used to raise an exception if invalid JSON numbers
   are encountered.

   .. versionchanged:: 3.1
      *parse_constant* doesn't get called on 'null', 'true', 'false' anymore.

   To use a custom :class:`JSONDecoder` subclass, specify it with the ``cls``
   kwarg; otherwise :class:`JSONDecoder` is used.  Additional keyword arguments
   will be passed to the constructor of the class.

   If the data being deserialized is not a valid JSON document, a
   :exc:`JSONDecodeError` will be raised.

   .. versionchanged:: 3.6
      All optional parameters are now :ref:`keyword-only <keyword-only_parameter>`.

   .. versionchanged:: 3.6
      *fp* can now be a :term:`binary file`. The input encoding should be
      UTF-8, UTF-16 or UTF-32.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 280)

Unknown directive type "function".

```
.. function:: loads(s, *, cls=None, object_hook=None, parse_float=None, parse_int=None, parse_constant=None, object_pa

   Deserialize *s* (a :class:`str`, :class:`bytes` or :class:`bytearray`
   instance containing a JSON document) to a Python object using this
   :ref:`conversion table <json-to-py-table>`.

   The other arguments have the same meaning as in :func:`load`.

   If the data being deserialized is not a valid JSON document, a
   :exc:`JSONDecodeError` will be raised.

   .. versionchanged:: 3.6
      *s* can now be of type :class:`bytes` or :class:`bytearray`. The
      input encoding should be UTF-8, UTF-16 or UTF-32.

   .. versionchanged:: 3.9
      The keyword argument *encoding* has been removed.
```

Encoders and Decoders

Simple JSON decoder.

Performs the following translations in decoding by default:

JSON	Python
object	dict
array	list
string	str
number (int)	int
number (real)	float
true	True
false	False
null	None

It also understands `NaN`, `Infinity`, and `-Infinity` as their corresponding `float` values, which is outside the JSON spec.

object_hook, if specified, will be called with the result of every JSON object decoded and its return value will be used in place of the given :class:`dict`. This can be used to provide custom deserializations (e.g. to support JSON-RPC class hinting).

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 333); *backlink*

Unknown interpreted text role "class".

object_pairs_hook, if specified will be called with the result of every JSON object decoded with an ordered list of pairs. The return value of *object_pairs_hook* will be used instead of the :class:`dict`. This feature can be used to implement custom decoders. If *object_hook* is also defined, the *object_pairs_hook* takes priority.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 338); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 344)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.1
   Added support for *object_pairs_hook*.
```

parse_float, if specified, will be called with the string of every JSON float to be decoded. By default, this is equivalent to `float(num_str)`. This can be used to use another datatype or parser for JSON floats (e.g. :class:`decimal.Decimal`).

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 347); *backlink*

Unknown interpreted text role "class".

parse_int, if specified, will be called with the string of every JSON int to be decoded. By default, this is equivalent to `int(num_str)`. This can be used to use another datatype or parser for JSON integers (e.g. :class:`float`).

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 352); *backlink*

Unknown interpreted text role "class".

parse_constant, if specified, will be called with one of the following strings: `'-Infinity'`, `'Infinity'`, `'NaN'`. This can be used to raise an exception if invalid JSON numbers are encountered.

If *strict* is false (`True` is the default), then control characters will be allowed inside strings. Control characters in this context are those with character codes in the 0--31 range, including `'\t'` (tab), `'\n'`, `'\r'` and `'\0'`.

If the data being deserialized is not a valid JSON document, a :exc:`JSONDecodeError` will be raised.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 367); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 370)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.6
   All parameters are now :ref:`keyword-only <keyword-only_parameter>`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 373)

Unknown directive type "method".

```
.. method:: decode(s)

   Return the Python representation of *s* (a :class:`str` instance
   containing a JSON document).

   :exc:`JSONDecodeError` will be raised if the given JSON document is not
   valid.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 381)

Unknown directive type "method".

```
.. method:: raw_decode(s)

   Decode a JSON document from *s* (a :class:`str` beginning with a
   JSON document) and return a 2-tuple of the Python representation
   and the index in *s* where the document ended.

   This can be used to decode a JSON document from a string that may have
   extraneous data at the end.
```

Extensible JSON encoder for Python data structures.

Supports the following objects and types by default:

Python	JSON
dict	object
list, tuple	array
str	string

Python	JSON
int, float, int- & float-derived Enums	number
True	true
False	false
None	null

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 417)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.4
   Added support for int- and float-derived Enum classes.
```

To extend this to recognize other objects, subclass and implement a :meth:`default` method with another method that returns a serializable object for `o` if possible, otherwise it should call the superclass implementation (to raise :exc:`TypeError`).

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 420); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 420); *backlink*

Unknown interpreted text role "exc".

If *skipkeys* is false (the default), a :exc:`TypeError` will be raised when trying to encode keys that are not :class:`str`, :class:`int`, :class:`float` or `None`. If *skipkeys* is true, such items are simply skipped.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 425); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 425); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 425); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 425); *backlink*

Unknown interpreted text role "class".

If *ensure_ascii* is true (the default), the output is guaranteed to have all incoming non-ASCII characters escaped. If *ensure_ascii* is false, these characters will be output as-is.

If *check_circular* is true (the default), then lists, dicts, and custom encoded objects will be checked for circular references during encoding to prevent an infinite recursion (which would cause a :exc:`RecursionError`). Otherwise, no such check takes place.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 433); *backlink*

Unknown interpreted text role "exc".

If *allow_nan* is true (the default), then `NaN`, `Infinity`, and `-Infinity` will be encoded as such. This behavior is not JSON specification compliant, but is consistent with most JavaScript based encoders and decoders. Otherwise, it will be a :exc:`ValueError` to encode such floats.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 438); *backlink*

Unknown interpreted text role "exc".

If *sort_keys* is true (default: `False`), then the output of dictionaries will be sorted by key; this is useful for regression tests to ensure that JSON serializations can be compared on a day-to-day basis.

If *indent* is a non-negative integer or string, then JSON array elements and object members will be pretty-printed with that indent level. An indent level of 0, negative, or `""` will only insert newlines. `None` (the default) selects the most compact representation. Using a positive integer indent indents that many spaces per level. If *indent* is a string (such as `"\t"`), that string is used to indent each level.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 455)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.2
   Allow strings for *indent* in addition to integers.
```

If specified, *separators* should be an `(item_separator, key_separator)` tuple. The default is `(', ', ': ')` if *indent* is `None` and `(',', ': ')` otherwise. To get the most compact JSON representation, you should specify `(',', ':')` to eliminate whitespace.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 463)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.4
   Use ``(',', ': ')`` as default if *indent* is not ``None``.
```

If specified, *default* should be a function that gets called for objects that can't otherwise be serialized. It should return a JSON encodable version of the object or raise a :exc:`TypeError`. If not specified, :exc:`TypeError` is raised.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 466); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 466); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 471)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.6
   All parameters are now :ref:`keyword-only <keyword-only_parameter>`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 475)

Unknown directive type "method".

```
.. method:: default(o)

   Implement this method in a subclass such that it returns a serializable
   object for *o*, or calls the base implementation (to raise a
   :exc:`TypeError`).

   For example, to support arbitrary iterators, you could implement
   :meth:`default` like this::

      def default(self, o):
         try:
             iterable = iter(o)
         except TypeError:
             pass
         else:
             return list(iterable)
         # Let the base class default method raise the TypeError
         return json.JSONEncoder.default(self, o)
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 495)

Unknown directive type "method".

```
.. method:: encode(o)

   Return a JSON string representation of a Python data structure, *o*.  For
   example::

     >>> json.JSONEncoder().encode({"foo": ["bar", "baz"]})
     '{"foo": ["bar", "baz"]}'
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 504)

Unknown directive type "method".

```
.. method:: iterencode(o)

   Encode the given object, *o*, and yield each string representation as
   available.  For example::

         for chunk in json.JSONEncoder().iterencode(bigobject):
             mysocket.write(chunk)
```

Exceptions

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 516)

Unknown directive type "exception".

```
.. exception:: JSONDecodeError(msg, doc, pos)

   Subclass of :exc:`ValueError` with the following additional attributes:

   .. attribute:: msg

      The unformatted error message.

   .. attribute:: doc
```

```
      The JSON document being parsed.

   .. attribute:: pos

      The start index of *doc* where parsing failed.

   .. attribute:: lineno

      The line corresponding to *pos*.

   .. attribute:: colno

      The column corresponding to *pos*.

   .. versionadded:: 3.5
```

Standard Compliance and Interoperability

The JSON format is specified by RFC 7159 and by ECMA-404. This section details this module's level of compliance with the RFC. For simplicity, :class:`JSONEncoder` and :class:`JSONDecoder` subclasses, and parameters other than those explicitly mentioned, are not considered.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 546); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 546); *backlink*

Unknown interpreted text role "class".

This module does not comply with the RFC in a strict fashion, implementing some extensions that are valid JavaScript but not valid JSON. In particular:

- Infinite and NaN number values are accepted and output;
- Repeated names within an object are accepted, and only the value of the last name-value pair is used.

Since the RFC permits RFC-compliant parsers to accept input texts that are not RFC-compliant, this module's deserializer is technically RFC-compliant under default settings.

Character Encodings

The RFC requires that JSON be represented using either UTF-8, UTF-16, or UTF-32, with UTF-8 being the recommended default for maximum interoperability.

As permitted, though not required, by the RFC, this module's serializer sets *ensure_ascii=True* by default, thus escaping the output so that the resulting strings only contain ASCII characters.

Other than the *ensure_ascii* parameter, this module is defined strictly in terms of conversion between Python objects and :class:`Unicode strings <str>`, and thus does not otherwise directly address the issue of character encodings.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 573); *backlink*

Unknown interpreted text role "class".

The RFC prohibits adding a byte order mark (BOM) to the start of a JSON text, and this module's serializer does not add a BOM to its output. The RFC permits, but does not require, JSON deserializers to ignore an initial BOM in their input. This module's deserializer raises a :exc:`ValueError` when an initial BOM is present.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 578); *backlink*

Unknown interpreted text role "exc".

The RFC does not explicitly forbid JSON strings which contain byte sequences that don't correspond to valid Unicode characters (e.g. unpaired UTF-16 surrogates), but it does note that they may cause interoperability problems. By default, this module accepts and outputs (when present in the original :class:`str`) code points for such sequences.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 584); *backlink*

Unknown interpreted text role "class".

Infinite and NaN Number Values

The RFC does not permit the representation of infinite or NaN number values. Despite that, by default, this module accepts and outputs `Infinity`, `-Infinity`, and `NaN` as if they were valid JSON number literal values:

```
>>> # Neither of these calls raises an exception, but the results are not valid JSON
>>> json.dumps(float('-inf'))
'-Infinity'
>>> json.dumps(float('nan'))
'NaN'
>>> # Same when deserializing
>>> json.loads('-Infinity')
-inf
>>> json.loads('NaN')
nan
```

In the serializer, the *allow_nan* parameter can be used to alter this behavior. In the deserializer, the *parse_constant* parameter can be used to alter this behavior.

Repeated Names Within an Object

The RFC specifies that the names within a JSON object should be unique, but does not mandate how repeated names in JSON objects should be handled. By default, this module does not raise an exception; instead, it ignores all but the last name-value pair for a given name:

```
>>> weird_json = '{"x": 1, "x": 2, "x": 3}'
>>> json.loads(weird_json)
{'x': 3}
```

The *object_pairs_hook* parameter can be used to alter this behavior.

Top-level Non-Object, Non-Array Values

The old version of JSON specified by the obsolete RFC 4627 required that the top-level value of a JSON text must be either a JSON object or array (Python :class:`dict` or :class:`list`), and could not be a JSON null, boolean, number, or string value. RFC 7159 removed that restriction, and this module does not and has never implemented that restriction in either its serializer or its deserializer.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 632); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 632); *backlink*

Unknown interpreted text role "class".

Regardless, for maximum interoperability, you may wish to voluntarily adhere to the restriction yourself.

Implementation Limitations

Some JSON deserializer implementations may set limits on:

- the size of accepted JSON texts
- the maximum level of nesting of JSON objects and arrays
- the range and precision of JSON numbers
- the content and maximum length of JSON strings

This module does not impose any such limits beyond those of the relevant Python datatypes themselves or the Python interpreter itself.

When serializing to JSON, beware any such limitations in applications that may consume your JSON. In particular, it is common for JSON numbers to be deserialized into IEEE 754 double precision numbers and thus subject to that representation's range and precision limitations. This is especially relevant when serializing Python :class:`int` values of extremely large magnitude, or when serializing instances of "exotic" numerical types such as :class:`decimal.Decimal`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 656); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 656); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 666)

Unknown directive type "program".

```
.. program:: json.tool
```

Command Line Interface

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 671)

Unknown directive type "module".

```
.. module:: json.tool
    :synopsis: A command line to validate and pretty-print JSON.
```

Source code: :source:`Lib/json/tool.py`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 674); *backlink*

Unknown interpreted text role "source".

The :mod:`json.tool` module provides a simple command line interface to validate and pretty-print JSON objects.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 678); *backlink*

Unknown interpreted text role "mod".

If the optional `infile` and `outfile` arguments are not specified, :attr:`sys.stdin` and :attr:`sys.stdout` will be used respectively:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 681); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 681); *backlink*

Unknown interpreted text role "attr".

```
$ echo '{"json": "obj"}' | python -m json.tool
{
    "json": "obj"
}
$ echo '{1.2:3.4}' | python -m json.tool
Expecting property name enclosed in double quotes: line 1 column 2 (char 1)
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 693)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.5
   The output is now in the same order as the input. Use the
   :option:`--sort-keys` option to sort the output of dictionaries
   alphabetically by key.
```

Command line options

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 702)

Unknown directive type "cmdoption".

```
.. cmdoption:: infile

   The JSON file to be validated or pretty-printed:

   .. code-block:: shell-session

      $ python -m json.tool mp_films.json
      [
          {
              "title": "And Now for Something Completely Different",
              "year": 1971
          },
          {
              "title": "Monty Python and the Holy Grail",
              "year": 1975
          }
      ]

   If *infile* is not specified, read from :attr:`sys.stdin`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 722)

Unknown directive type "cmdoption".

```
.. cmdoption:: outfile

   Write the output of the *infile* to the given *outfile*. Otherwise, write it
   to :attr:`sys.stdout`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 727)

Unknown directive type "cmdoption".

```
.. cmdoption:: --sort-keys

   Sort the output of dictionaries alphabetically by key.

   .. versionadded:: 3.5
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 733)

Unknown directive type "cmdoption".

```
.. cmdoption:: --no-ensure-ascii

   Disable escaping of non-ascii characters, see :func:`json.dumps` for more information.

   .. versionadded:: 3.9
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 739)

Unknown directive type "cmdoption".

```
.. cmdoption:: --json-lines

   Parse every input line as separate JSON object.

   .. versionadded:: 3.8
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 745)

Unknown directive type "cmdoption".

```
.. cmdoption:: --indent, --tab, --no-indent, --compact

   Mutually exclusive options for whitespace control.

   .. versionadded:: 3.9
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)json.rst, line 751)

Unknown directive type "cmdoption".

```
.. cmdoption:: -h, --help

   Show the help message.
```

Footnotes

[1] As noted in the errata for RFC 7159, JSON permits literal U+2028 (LINE SEPARATOR) and U+2029 (PARAGRAPH SEPARATOR) characters in strings, whereas JavaScript (as of ECMAScript Edition 5.1) does not.