

Discover Members of a Function/Class

If you ever need to find out what members a function or class has, you can use the following trick (this example uses the "json" class):

```
# New unknown script package I have never used before
import json

# Find out all members in this package
for item in dir(json):
    print item
```

Which will output the following:

```
JSONDecoder
JSONEncoder
__all__
__author__
__builtins__
__doc__
__file__
__name__
__package__
__path__
__version__
_default_decoder
_default_encoder
decoder
dump
dumps
encoder
load
loads
scanner
```

You can even print the docstring from a class using the syntax like the following example for "json.dump":

```
print json.dump.__doc__
```

Which should output:

```
"""
Serialize ``obj`` as a JSON formatted stream to ``fp`` (a
``.write()``-supporting file-like object).

If ``skipkeys`` is true then ``dict`` keys that are not basic types
(``str``, ``unicode``, ``int``, ``long``, ``float``, ``bool``, ``None``)
will be skipped instead of raising a ``TypeError``.

If ``ensure_ascii`` is true (the default), all non-ASCII characters in the
output are escaped with ``\uXXXX`` sequences, and the result is a ``str``
instance consisting of ASCII characters only. If ``ensure_ascii`` is
``False``, some chunks written to ``fp`` may be ``unicode`` instances.
This usually happens because the input contains unicode strings or the
``encoding`` parameter is used. Unless ``fp.write()`` explicitly
understands ``unicode`` (as in ``codecs.getwriter``) this is likely to
cause an error.

If ``check_circular`` is false, then the circular reference check
for container types will be skipped and a circular reference will
result in an ``OverflowError`` (or worse).

If ``allow_nan`` is false, then it will be a ``ValueError`` to
serialize out of range ``float`` values (``nan``, ``inf``, ``-inf``)
in strict compliance of the JSON specification, instead of using the
JavaScript equivalents (``NaN``, ``Infinity``, ``-Infinity``).

If ``indent`` is a non-negative integer, then JSON array elements and
object members will be pretty-printed with that indent level. An indent
level of 0 will only insert newlines. ``None`` is the most compact
representation. Since the default item separator is ``', '`` , the
output might include trailing whitespace when ``indent`` is specified.
You can use ``separators=(',', ': ')`` to avoid this.
```

If `separators` is an `(item_separator, dict_separator)` tuple then it will be used instead of the default `(' ', ': ')` separators. `(' ', ': ')` is the most compact JSON representation.

`encoding` is the character encoding for str instances, default is UTF-8.

`default(obj)` is a function that should return a serializable version of obj or raise `TypeError`. The default simply raises `TypeError`.

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

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

"""

Source: <https://forum.inductiveautomation.com/t/dos-and-donts-when-developing-first-project-with-ignition/93592/33>

Revision #1

Created 2024-09-22 02:27:10 UTC by Michael Flagler

Updated 2024-09-22 02:34:30 UTC by Michael Flagler