Author: Aaron Valoroso

Title: Python 3 (Special Methods)

Date: January 25th, 2018

object.\_\_new\_\_(cls[, ...]):

Called to create a new instance of class cls. [\_\_new\_\_()](https://docs.python.org/3/reference/datamodel.html#object.__new__) is a static method (special-cased so you need not declare it as such) that takes the class of which an instance was requested as its first argument. The remaining arguments are those passed to the object constructor expression (the call to the class). The return value of [\_\_new\_\_()](https://docs.python.org/3/reference/datamodel.html#object.__new__) should be the new object instance (usually an instance of cls).

object.\_\_init\_\_(self[, ...]):

Called after the instance has been created (by [\_\_new\_\_()](https://docs.python.org/3/reference/datamodel.html#object.__new__)), but before it is returned to the caller. The arguments are those passed to the class constructor expression. If a base class has an [\_\_init\_\_()](https://docs.python.org/3/reference/datamodel.html#object.__init__) method, the derived class’s [\_\_init\_\_()](https://docs.python.org/3/reference/datamodel.html#object.__init__) method, if any, must explicitly call it to ensure proper initialization of the base class part of the instance; for example: super().\_\_init\_\_([args...]).

object.\_\_del\_\_(self):

Called when the instance is about to be destroyed. This is also called a finalizer or (improperly) a destructor. If a base class has a [\_\_del\_\_()](https://docs.python.org/3/reference/datamodel.html#object.__del__) method, the derived class’s [\_\_del\_\_()](https://docs.python.org/3/reference/datamodel.html#object.__del__) method, if any, must explicitly call it to ensure proper deletion of the base class part of the instance.

object.\_\_repr\_\_(self):

Called by the [repr()](https://docs.python.org/3/library/functions.html" \l "repr" \o "repr) built-in function to compute the “official” string representation of an object. If at all possible, this should look like a valid Python expression that could be used to recreate an object with the same value (given an appropriate environment). If this is not possible, a string of the form <...some useful description...> should be returned. The return value must be a string object. If a class defines [\_\_repr\_\_()](https://docs.python.org/3/reference/datamodel.html#object.__repr__) but not [\_\_str\_\_()](https://docs.python.org/3/reference/datamodel.html#object.__str__), then [\_\_repr\_\_()](https://docs.python.org/3/reference/datamodel.html#object.__repr__) is also used when an “informal” string representation of instances of that class is required.

object.\_\_str\_\_(self):

Called by [str(object)](https://docs.python.org/3/library/stdtypes.html" \l "str" \o "str) and the built-in functions [format()](https://docs.python.org/3/library/functions.html#format) and [print()](https://docs.python.org/3/library/functions.html#print) to compute the “informal” or nicely printable string representation of an object. The return value must be a [string](https://docs.python.org/3/library/stdtypes.html#textseq) object.

object.\_\_bytes\_\_(self):

Called by [bytes](https://docs.python.org/3/library/functions.html#func-bytes) to compute a byte-string representation of an object. This should return a [bytes](https://docs.python.org/3/library/stdtypes.html#bytes) object.

object.\_\_format\_\_(self, format\_spec):

Called by the [format()](https://docs.python.org/3/library/functions.html#format) built-in function, and by extension, evaluation of [formatted string literals](https://docs.python.org/3/reference/lexical_analysis.html#f-strings) and the [str.format()](https://docs.python.org/3/library/stdtypes.html" \l "str.format" \o "str.format) method, to produce a “formatted” string representation of an object. The format\_spec argument is a string that contains a description of the formatting options desired. The interpretation of the format\_spec argument is up to the type implementing [\_\_format\_\_()](https://docs.python.org/3/reference/datamodel.html#object.__format__), however most classes will either delegate formatting to one of the built-in types, or use a similar formatting option syntax.

object.\_\_lt\_\_(self, other)[¶](https://docs.python.org/3/reference/datamodel.html#object.__lt__)

object.\_\_le\_\_(self, other)

object.\_\_eq\_\_(self, other)

object.\_\_ne\_\_(self, other)

object.\_\_gt\_\_(self, other)

object.\_\_ge\_\_(self, other)

These are the so-called “rich comparison” methods. The correspondence between operator symbols and method names is as follows: x<y calls x.\_\_lt\_\_(y), x<=y calls x.\_\_le\_\_(y), x==y calls x.\_\_eq\_\_(y), x!=y calls x.\_\_ne\_\_(y), x>y calls x.\_\_gt\_\_(y), and x>=y calls x.\_\_ge\_\_(y).

A rich comparison method may return the singleton NotImplemented if it does not implement the operation for a given pair of arguments. By convention, False and True are returned for a successful comparison. However, these methods can return any value, so if the comparison operator is used in a Boolean context (e.g., in the condition of an if statement), Python will call [bool()](https://docs.python.org/3/library/functions.html#bool) on the value to determine if the result is true or false.

object.\_\_hash\_\_(self):

Called by built-in function [hash()](https://docs.python.org/3/library/functions.html#hash) and for operations on members of hashed collections including [set](https://docs.python.org/3/library/stdtypes.html#set), [frozenset](https://docs.python.org/3/library/stdtypes.html" \l "frozenset" \o "frozenset), and [dict](https://docs.python.org/3/library/stdtypes.html#dict). [\_\_hash\_\_()](https://docs.python.org/3/reference/datamodel.html#object.__hash__) should return an integer. The only required property is that objects which compare equal have the same hash value; it is advised to mix together the hash values of the components of the object that also play a part in comparison of objects by packing them into a tuple and hashing the tuple.

object.\_\_bool\_\_(self):

Called to implement truth value testing and the built-in operation bool(); should return False or True. When this method is not defined, [\_\_len\_\_()](https://docs.python.org/3/reference/datamodel.html#object.__len__) is called, if it is defined, and the object is considered true if its result is nonzero. If a class defines neither [\_\_len\_\_()](https://docs.python.org/3/reference/datamodel.html#object.__len__) nor [\_\_bool\_\_()](https://docs.python.org/3/reference/datamodel.html#object.__bool__), all its instances are considered true.

object.\_\_getattr\_\_(self, name):

Called when an attribute lookup has not found the attribute in the usual places (i.e. it is not an instance attribute nor is it found in the class tree for self). name is the attribute name. This method should return the (computed) attribute value or raise an [AttributeError](https://docs.python.org/3/library/exceptions.html#AttributeError) exception.

object.\_\_getattribute\_\_(self, name):

Called unconditionally to implement attribute accesses for instances of the class. If the class also defines [\_\_getattr\_\_()](https://docs.python.org/3/reference/datamodel.html#object.__getattr__), the latter will not be called unless [\_\_getattribute\_\_()](https://docs.python.org/3/reference/datamodel.html#object.__getattribute__) either calls it explicitly or raises an [AttributeError](https://docs.python.org/3/library/exceptions.html" \l "AttributeError" \o "AttributeError). This method should return the (computed) attribute value or raise an [AttributeError](https://docs.python.org/3/library/exceptions.html" \l "AttributeError" \o "AttributeError) exception. In order to avoid infinite recursion in this method, its implementation should always call the base class method with the same name to access any attributes it needs, for example, object.\_\_getattribute\_\_(self, name).

object.\_\_setattr\_\_(self, name, value):

Called when an attribute assignment is attempted. This is called instead of the normal mechanism (i.e. store the value in the instance dictionary). name is the attribute name, value is the value to be assigned to it.

object.\_\_delattr\_\_(self, name):

Like [\_\_setattr\_\_()](https://docs.python.org/3/reference/datamodel.html#object.__setattr__) but for attribute deletion instead of assignment. This should only be implemented if del obj.name is meaningful for the object.

object.\_\_dir\_\_(self):

Called when [dir()](https://docs.python.org/3/library/functions.html" \l "dir" \o "dir) is called on the object. A sequence must be returned. [dir()](https://docs.python.org/3/library/functions.html#dir) converts the returned sequence to a list and sorts it.

object.\_\_get\_\_(self, instance, owner):

Called to get the attribute of the owner class (class attribute access) or of an instance of that class (instance attribute access). owner is always the owner class, while instance is the instance that the attribute was accessed through, or None when the attribute is accessed through the owner. This method should return the (computed) attribute value or raise an [AttributeError](https://docs.python.org/3/library/exceptions.html" \l "AttributeError" \o "AttributeError) exception.

object.\_\_set\_\_(self, instance, value):

Called to set the attribute on an instance instance of the owner class to a new value, value.

object.\_\_delete\_\_(self, instance):

Called to delete the attribute on an instance instance of the owner class.

object.\_\_set\_name\_\_(self, owner, name):

Called at the time the owning class owner is created. The descriptor has been assigned to name.

object.\_\_slots\_\_:

This class variable can be assigned a string, iterable, or sequence of strings with variable names used by instances. \_\_slots\_\_ reserves space for the declared variables and prevents the automatic creation of \_\_dict\_\_ and \_\_weakref\_\_ for each instance.

classmethod object.\_\_init\_subclass\_\_(cls)

This method is called whenever the containing class is subclassed. cls is then the new subclass. If defined as a normal instance method, this method is implicitly converted to a class method.

\_\_prepare\_\_?

\_\_class\_\_:

Identifies a new class object.

class.\_\_instancecheck\_\_(self, instance):

Return true if instance should be considered a (direct or indirect) instance of class. If defined, called to implement isinstance(instance, class).

class.\_\_subclasscheck\_\_(self, subclass):

Return true if subclass should be considered a (direct or indirect) subclass of class. If defined, called to implement issubclass(subclass, class).

object.\_\_call\_\_(self[, args...]):

Called when the instance is “called” as a function; if this method is defined, x(arg1, arg2, ...) is a shorthand for x.\_\_call\_\_(arg1, arg2, ...).

object.\_\_len\_\_(self):

Called to implement the built-in function [len()](https://docs.python.org/3/library/functions.html" \l "len" \o "len). Should return the length of the object, an integer >= 0. Also, an object that doesn’t define a [\_\_bool\_\_()](https://docs.python.org/3/reference/datamodel.html#object.__bool__) method and whose [\_\_len\_\_()](https://docs.python.org/3/reference/datamodel.html#object.__len__) method returns zero is considered to be false in a Boolean context.

object.\_\_length\_hint\_\_(self):

Called to implement [operator.length\_hint()](https://docs.python.org/3/library/operator.html" \l "operator.length_hint" \o "operator.length_hint). Should return an estimated length for the object (which may be greater or less than the actual length). The length must be an integer >= 0. This method is purely an optimization and is never required for correctness.

object.\_\_getitem\_\_(self, key):

Called to implement evaluation of self[key]. For sequence types, the accepted keys should be integers and slice objects. Note that the special interpretation of negative indexes (if the class wishes to emulate a sequence type) is up to the [\_\_getitem\_\_()](https://docs.python.org/3/reference/datamodel.html#object.__getitem__) method. If key is of an inappropriate type, [TypeError](https://docs.python.org/3/library/exceptions.html" \l "TypeError" \o "TypeError) may be raised; if of a value outside the set of indexes for the sequence (after any special interpretation of negative values), [IndexError](https://docs.python.org/3/library/exceptions.html#IndexError) should be raised. For mapping types, if key is missing (not in the container), [KeyError](https://docs.python.org/3/library/exceptions.html" \l "KeyError" \o "KeyError) should be raised.

object.\_\_missing\_\_(self, key):

Called by [dict](https://docs.python.org/3/library/stdtypes.html" \l "dict" \o "dict).[\_\_getitem\_\_()](https://docs.python.org/3/reference/datamodel.html#object.__getitem__) to implement self[key] for dict subclasses when key is not in the dictionary.

object.\_\_setitem\_\_(self, key, value):

Called to implement assignment to self[key]. Same note as for [\_\_getitem\_\_()](https://docs.python.org/3/reference/datamodel.html#object.__getitem__). This should only be implemented for mappings if the objects support changes to the values for keys, or if new keys can be added, or for sequences if elements can be replaced. The same exceptions should be raised for improper key values as for the [\_\_getitem\_\_()](https://docs.python.org/3/reference/datamodel.html#object.__getitem__) method.

object.\_\_delitem\_\_(self, key):

Called to implement deletion of self[key]. Same note as for [\_\_getitem\_\_()](https://docs.python.org/3/reference/datamodel.html#object.__getitem__). This should only be implemented for mappings if the objects support removal of keys, or for sequences if elements can be removed from the sequence. The same exceptions should be raised for improper keyvalues as for the [\_\_getitem\_\_()](https://docs.python.org/3/reference/datamodel.html#object.__getitem__) method.

object.\_\_iter\_\_(self):

This method is called when an iterator is required for a container. This method should return a new iterator object that can iterate over all the objects in the container. For mappings, it should iterate over the keys of the container.

object.\_\_reversed\_\_(self):

Called (if present) by the [reversed()](https://docs.python.org/3/library/functions.html#reversed) built-in to implement reverse iteration. It should return a new iterator object that iterates over all the objects in the container in reverse order.

object.\_\_contains\_\_(self, item):

Called to implement membership test operators. Should return true if item is in self, false otherwise. For mapping objects, this should consider the keys of the mapping rather than the values or the key-item pairs.

The following are all different ways for overloaded operators. These methods are called to implement the binary arithmetic operations

(+, -, \*, @, /, //, %, [divmod()](https://docs.python.org/3/library/functions.html" \l "divmod" \o "divmod), [pow()](https://docs.python.org/3/library/functions.html#pow), \*\*, <<, >>, &, ^, |).

object.\_\_add\_\_(self, other)

object.\_\_sub\_\_(self, other)

object.\_\_mul\_\_(self, other)

object.\_\_matmul\_\_(self, other)

object.\_\_truediv\_\_(self, other)

object.\_\_floordiv\_\_(self, other)

object.\_\_mod\_\_(self, other)

object.\_\_divmod\_\_(self, other)

object.\_\_pow\_\_(self, other[, modulo])

object.\_\_lshift\_\_(self, other)

object.\_\_rshift\_\_(self, other)

object.\_\_and\_\_(self, other)

object.\_\_xor\_\_(self, other)

object.\_\_or\_\_(self, other)

These allow you to have an object and arbitrary value to be added to the object.

object.\_\_radd\_\_(self, other)

object.\_\_rsub\_\_(self, other)

object.\_\_rmul\_\_(self, other)

object.\_\_rmatmul\_\_(self, other)

object.\_\_rtruediv\_\_(self, other)

object.\_\_rfloordiv\_\_(self, other)

object.\_\_rmod\_\_(self, other)

object.\_\_rdivmod\_\_(self, other)

object.\_\_rpow\_\_(self, other)

object.\_\_rlshift\_\_(self, other)

object.\_\_rrshift\_\_(self, other)

object.\_\_rand\_\_(self, other)

object.\_\_rxor\_\_(self, other)

object.\_\_ror\_\_(self, other)

These methods are called to implement the augmented arithmetic assignments (+=, -=, \*=, @=, /=, //=, %=, \*\*=, <<=, >>=, &=, ^=, |=).

object.\_\_iadd\_\_(self, other)

object.\_\_isub\_\_(self, other)

object.\_\_imul\_\_(self, other)

object.\_\_imatmul\_\_(self, other)

object.\_\_itruediv\_\_(self, other)

object.\_\_ifloordiv\_\_(self, other)

object.\_\_imod\_\_(self, other)

object.\_\_ipow\_\_(self, other[, modulo])

object.\_\_ilshift\_\_(self, other)

object.\_\_irshift\_\_(self, other)

object.\_\_iand\_\_(self, other)

object.\_\_ixor\_\_(self, other)

object.\_\_ior\_\_(self, other)

Called to implement the unary arithmetic operations (-, +, [abs()](https://docs.python.org/3/library/functions.html#abs) and ~).

object.\_\_neg\_\_(self)

object.\_\_pos\_\_(self)

object.\_\_abs\_\_(self)

object.\_\_invert\_\_(self)

Called to implement the built-in functions [complex()](https://docs.python.org/3/library/functions.html#complex), [int()](https://docs.python.org/3/library/functions.html" \l "int" \o "int), [float()](https://docs.python.org/3/library/functions.html#float) and [round()](https://docs.python.org/3/library/functions.html#round). Should return a value of the appropriate type.

object.\_\_complex\_\_(self)

object.\_\_int\_\_(self)

object.\_\_float\_\_(self)

object.\_\_round\_\_(self[, n])

object.\_\_index\_\_(self):

Called to implement [operator.index()](https://docs.python.org/3/library/operator.html" \l "operator.index" \o "operator.index), and whenever Python needs to losslessly convert the numeric object to an integer object (such as in slicing, or in the built-in [bin()](https://docs.python.org/3/library/functions.html#bin), [hex()](https://docs.python.org/3/library/functions.html#hex) and [oct()](https://docs.python.org/3/library/functions.html" \l "oct" \o "oct) functions). Presence of this method indicates that the numeric object is an integer type. Must return an integer.

object.\_\_enter\_\_(self):

Enter the runtime context related to this object. The [with](https://docs.python.org/3/reference/compound_stmts.html#with) statement will bind this method’s return value to the target(s) specified in the [as](https://docs.python.org/3/reference/compound_stmts.html#as) clause of the statement, if any.

object.\_\_exit\_\_(self, exc\_type, exc\_value, traceback):

Exit the runtime context related to this object. The parameters describe the exception that caused the context to be exited. If the context was exited without an exception, all three arguments will be [None](https://docs.python.org/3/library/constants.html#None).

object.\_\_await\_\_(self):

Must return an [iterator](https://docs.python.org/3/glossary.html#term-iterator). Should be used to implement [awaitable](https://docs.python.org/3/glossary.html" \l "term-awaitable) objects. For instance, [asyncio.Future](https://docs.python.org/3/library/asyncio-task.html" \l "asyncio.Future" \o "asyncio.Future) implements this method to be compatible with the [await](https://docs.python.org/3/reference/expressions.html#await) expression.