OOP ADVANCED

Comparison methods

Function or operator	Magic method	Implementation meaning or purpose
==	__eq__(self, other)	equality operator
!=	__ne__(self, other)	inequality operator
<	__lt__(self, other)	less-than operator
>	__gt__(self, other)	greater-than operator
<=	__le__(self, other)	less-than-or-equal-to operator
>=	__ge__(self, other)	greater-than-or-equal-to operator

OOP ADVANCED

Unary operators and functions

Function or operator	Magic method	Implementation meaning or purpose
+	__pos__(self)	**unary** positive, like a = +b
-	__neg__(self)	**unary** negative, like a = -b
abs()	__abs__(self)	behavior for abs() function
round(a, b)	__round__(self, b)	behavior for round() function

OOP ADVANCED

Common, binary operators and functions

Function or operator	Magic method	Implementation meaning or purpose
+	__add__(self, other)	addition operator
-	__sub__(self, other)	subtraction operator
*	__mul__(self, other)	multiplication operator
//	__floordiv__(self, other)	integer division operator
/	__div__(self, other)	division operator
%	__mod__(self, other)	modulo operator
**	__pow__(self, other)	exponential (power) operator

OOP ADVANCED

Augumented operators and functions

By augumented assignment we should understand a sequence of unary operators and assignments like `a += 20`

Function or operator	Magic method	Implementation meaning or purpose
+=	__iadd__(self, other)	addition and assignment operator
-=	__isub__(self, other)	subtraction and assignment operator
*=	__imul__(self, other)	multiplication and assignment operator
//=	__ifloordiv__(self, other)	integer division and assignment operator
/=	__idiv__(self, other)	division and assignment operator
%=	__imod__(self, other)	modulo and assignment operator
**=	__ipow__(self, other)	exponential (power) and assignment operator

OOP ADVANCED

Type conversion methods

Python offers a set of methods responsible for the conversion of built-in data types.

Function	Magic method	Implementation meaning or purpose
int()	__int__(self)	conversion to integer type
float()	__float__(self)	conversion to float type
oct()	__oct__(self)	conversion to string, containing an octal representation
hex()	__hex__(self)	conversion to string, containing a hexadecimal representation

OOP ADVANCED

Object introspection

Python offers a set of methods responsible for representing object details using ordinary strings.

Function	Magic method	Implementation meaning or purpose
str()	__str__(self)	responsible for handling str() function calls
repr()	__repr__(self)	responsible for handling repr() function calls
format()	__format__(self, formatstr)	called when new-style string formatting is applied to an object
hash()	__hash__(self)	responsible for handling hash() function calls
dir()	__dir__(self)	responsible for handling dir() function calls
bool()	__nonzero__(self)	responsible for handling bool() function calls

OOP ADVANCED

Object retrospection

Following the topic of object introspection, there are methods responsible for object reflection.

Function	Magic method	Implementation meaning or purpose
isinstance(object, class)	__instancecheck__(self, object)	responsible for handling isinstance() function calls
issubclass(subclass, class)	__subclasscheck__(self, subclass)	responsible for handling issubclass() function calls

OOP ADVANCED

Object attribute access

Access to object attributes can be controlled via the following magic methods

Expression example	Magic method	Implementation meaning or purpose
object.attribute	__getattr__(self, attribute)	responsible for handling access to a non-existing attribute
object.attribute	__getattribute__(self, attribute)	responsible for handling access to an existing attribute
object.attribute = value	__setattr__(self, attribute, value)	responsible for setting an attribute value
del object.attribute	__delattr__(self, attribute)	responsible for deleting an attribute

OOP ADVANCED

Methods allowing access to containers

Containers are any object that holds an arbitrary number of other objects; containers provide a way to access the contained objects and to iterate over them. Container examples: list, dictionary, tuple, and set.

Expression example	Magic method	Implementation meaning or purpose
len(container)	__len__(self)	returns the length (number of elements) of the container
container[key]	__getitem__(self, key)	responsible for accessing (fetching) an element identified by the key argument
container[key] = value	__setitem__(self, key, value)	responsible for setting a value to an element identified by the key argument
del container[key]	__delitem__(self, key)	responsible for deleting an element identified by the key argument
for element in container	__iter__(self)	returns an iterator for the container
item in container	__contains__(self, item)	responds to the question: does the container contain the selected item?