Notes:

1. My PHP OOP rule of thumb

* For a simple site, OOP adds unnecessary complexity
* For a complex site, OOP adds necessary simplicity

Overloading in PHP

* Overloading in PHP provides means to dynamically "create" properties and methods. These dynamic entities are processed via magic methods one can establish in a class for various action types.
* The overloading methods are invoked when interacting with properties or methods that have not been declared or are not visible in the current scope. The rest of this section will use the terms "inaccessible properties" and "inaccessible methods" to refer to this combination of declaration and visibility.

In PHP Overloading means

* Dynamically create property or method that:
* Has not been declared
* Is not visible in the current scope
* Dynamic entities processed using “magic methods”
* PHP's interpretation of "overloading" is different than most object oriented languages. Overloading traditionally provides the ability to have multiple methods with the same name but different quantities and types of arguments.

In PHP

- With properties of an object, there are two events that can be triggered upon

- Access of a missing or out-of-scope property

- When we get a property and when we set a property. It can be useful to do things like use a method or function to determine a value to be returned or execute some sort of fail-safe behavior if we try to set a property that doesn’t exist.

Magic Methods

* Magic methods are a collection of specialized methods with a unique naming convention that are executed in response to a PHP event.
* Each magic method name is prepended with two underscores such as \_\_construct() and \_\_toString().
* About dozen in PHP

Why use magic method?

* We can trigger custom behavior upon specific events, such as attempting to access an unset property or calling method that doesn’t exist.
* We can also customize creation of an object and set defaults that aren’t available at runtime such as the logged-in user or the current time.

Benchmarking

* Benchmarking is the process of assessing an application. There are various benchmarking tools for different platforms and/or different programming languages.
* In computing, a benchmark is the act of running a computer program, a set of programs, or other operations, in order to assess the relative performance of an object, normally by running a number of standard tests and trials against it.
* Benchmarking is usually associated with assessing performance characteristics of computer hardware, for example, the floating point operation performance of a CPU, but there are circumstances when the technique is also applicable to software. Software benchmarks are, for example, run against compilers or database management systems.

Why not magic methods?

* Magic methods do have an overhead, leading to code execution that is anywhere from 3-20 times slower for that particular method call.
* Magic methods ignore scope, meaning that we can accidentally expose a property or method that is normally hidden from the rest of the program.
* Magic methods can break code completion in IDE’s, meaning the IDE can’t follow the logic and may assume that we’re trying to access something that doesn’t exist.

Decide for yourself

* IDE warning vs. contextually responsive code
* Performance deference is negligible outside of benchmarking
* Customize scope controls

Prepend underscore with the property

* Prepend the property name with underscore to visually indicate the scope.

Use of Magic Methods \_\_get() and \_\_set() in Program

* With properties of an object, there are two events that can be triggered upon access of missing or out-of-scope property
* When we get a property and when we set a property. It can be useful to do things like use a method or function to determine a value to be returned or execute some of fail-safe behavior if we try to set a property that doesn’t exist.
* In the address class, a magic get method will useful for calculating a missing value.
* For example, if the postal code is null but there is a city and subdivision, we could attempt to look up the value in a database table.
* However, as the postal code is currently public, the magic get method won’t be triggered therefore, we’ll change the scope of the postal code to protect it, in order to be able to use the magic get.
* Magic methods are public, so we do not have to specify scope when declaring it.
* Magic \_\_ get() method take only one argument: the name of the property in the form of string.
* The magic set method two arguments. The first is the name of the property as a string, and the second is a mixed value to be set function \_\_set name value.
* \_\_set() is run when writing data to inaccessible properties.
* \_\_get() is utilized for reading data from inaccessible properties.
* \_\_get, \_\_set, \_\_call and \_\_callStatic are invoked when the method or property is inaccessible.
* The magic methods are not substitutes for getters and setters. They just allow you to handle method calls or property access that would otherwise result in an error. As such, there are much more related to error handling. Also note that they are considerably slower than using proper getter and setter or direct method calls.
* \_\_get() & \_\_set() are only called on reading/writing inaccessible properties.

Pros

* If you have an object with 3 and only 3 attributes you don't need to use magic setters/getters, but in some advanced cases they are a great way to do very complex things (ORM systems etc...)

Cons

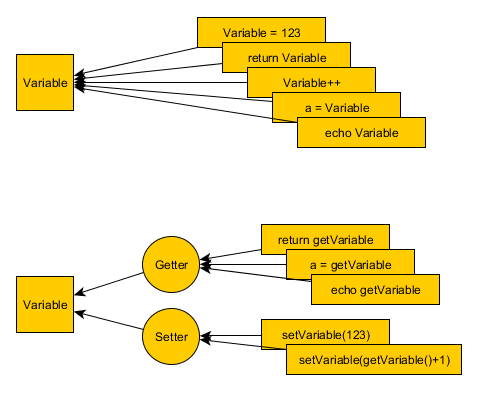
* Text searches don't find the functions
* System is harder to understand, especially for newcomers
* Refactoring tools might fail more often

And no, you must not use magic methods. For PHP, Magic Method are evil. Why?

* Are hard to debug.
* Exists a performance impact.
* Add more code.

Why use getters and setters?

* Scalability: It's easier refactor a getter than search all the var assignments in a project code.
* Debugging: You can put breakpoints at setters and getters.
* Cleaner: Magic functions are not good solution for writing less, your IDE will not suggest the code. Better use templates for fast-writing getters.



Magic Constructor Method

* As of PHP5 developers, can declare a magic method that can be used to initialize a new object before it is used.
* It can be impractical to have to specify each property when populating an object.
* Its object initialization by array that’s why database record is easily populated by object properties because database record that has been returned as an array.

Magic method \_\_toString()

* Rather than explicitly telling an object to display itself, it’s more convenient to act upon the object in a standardized way, without having to figure out what the method to call is or how it works
* In computer science, this concept is known as polymorphism
* In practice, this means there is a common functional interface across multiple classes.
* Catchable fatal error: Object of class Address could not be converted to string in C:\xampp\htdocs\Object-Oriented-PHP\Magic-Methods\Magic\_get()\_set()\demo.php on line 41
* This is because the object does not have a method defined to convert itself to a string
* The \_\_toString() Method – Objects as Strings.
* The \_\_toString() method is called when the code attempts to treat an object like a string.
* This function does not accept any arguments and should return a string.
* The main use of \_\_toString() function is to print the methods and attributes of the class.