In Python, methods are behaviors associated with object parameters that modify the state of that object. They are essentially functions that belong to a specific instance of a class. This means that calling a method on a list, for example, only modifies that instance of the list, and not all lists globally.

Methods in Python fall into several categories:

* Instance methods
* Class methods
* Static methods

## ****Instance methods****

**Instance methods** are the most common type of methods in Python. You define instance methods within a class by creating functions inside the class definition. When you instantiate instances of a class, those individual instances can have their methods called so the program can control and modify those instances directly. Instance methods can take a parameter called **self**, which represents the instance the method is being executed on, that allows you to access attributes of the instance using dot notation, like **self.name**, which will access the name attribute of that specific instance of the class object. When you have variables that contain different values for different instances, these are called instance variables.

## ****Class methods****

**Class methods**, on the other hand, are called for the class itself instead of an instance. They are marked with a **@classmethod** decorator and take a **cls** parameter that points to the class—and not any specific instance—when the method is called. One common use-case for class methods is to create and modify data structures that contain records for all instances of a class. Usually, programmers make a list inside the class definition, and methods to add instances of the class to that list in order to keep track of that class.

## ****Static methods****

Lastly, static methods, marked with a **@staticmethod** decorator, do not take a self or a **cls** parameter. Static methods behave like plain functions, except that you can call them directly from the class. It is important to note that you do not have to actually instantiate the class, the methods just reside in there. This is because class definitions are themselves an object (i.e., an instance of abstract base class), which reduces overhead and allows functions to be encapsulated in an easy-to-use encapsulation. Programmers use static methods when the method does not need to access any instance or class-specific data.

## ****Choosing a method type****

The type of method you choose to use—instance, class, or static—depends on what data the method needs to access. Think of these methods as different tools in your toolbox, each with a different use-case depending on the data you need to work with.

* Instance methods are for individual object data
* Class methods for shared data,Static methods for related tasks that don't need to access or modify any object or class data

## ****Key takeaways****

Remember, methods in Python are a way to bundle behavior with objects, allowing you to interact with and modify the state of those objects. However, static methods offer a way to bundle functions together, to be used in general on any other type of object. Bundling functions together helps organize functions in a clean manner and helps package them for reuse in other coding projects.