# Special Methods in Python

- **Special methods** have double underscores at the beginning and end of their names. For example: __str__.
- They are called automatically by Python when certain operations are performed or when a specific syntax is used.
- They allow you to customize the behavior of your objects for built-in operations. For example, getting the length of an object, and adding objects of a specific type.
- __init__() is a special method too. It is called automatically when an instance of a class is created.
- Some examples of special methods are:
    - **__str__():** to get a user-friendly representation of the object.
    - **__repr__():** to get a developer-friendly representation of the object.
    - **__len__():** to get the length of an object.
    - **__add__():** to add two objects.
    - **__getitem__():** to get an element from an object, as if it was a sequence.
    - **__bool__():** to make the object evaluate to True or False, based on a specific condition.
    - **__iter__():** to make an object an iterable.
    - **__next__():** to retrieve the next element from an iterator.

**Fig. 1 Special Methods**

This table summarizes some of the most commonly used special methods and when they are called automatically:

| Special Method | Called by |
|---|---|
| __str__() | str() and print() |
| __len__() | len() |
| __getitem__() | <obj>[<index>] |
| __add__() | + |
| __bool__() | bool() |

**Fig. 2 Special Methods for Arithmetic Operations**

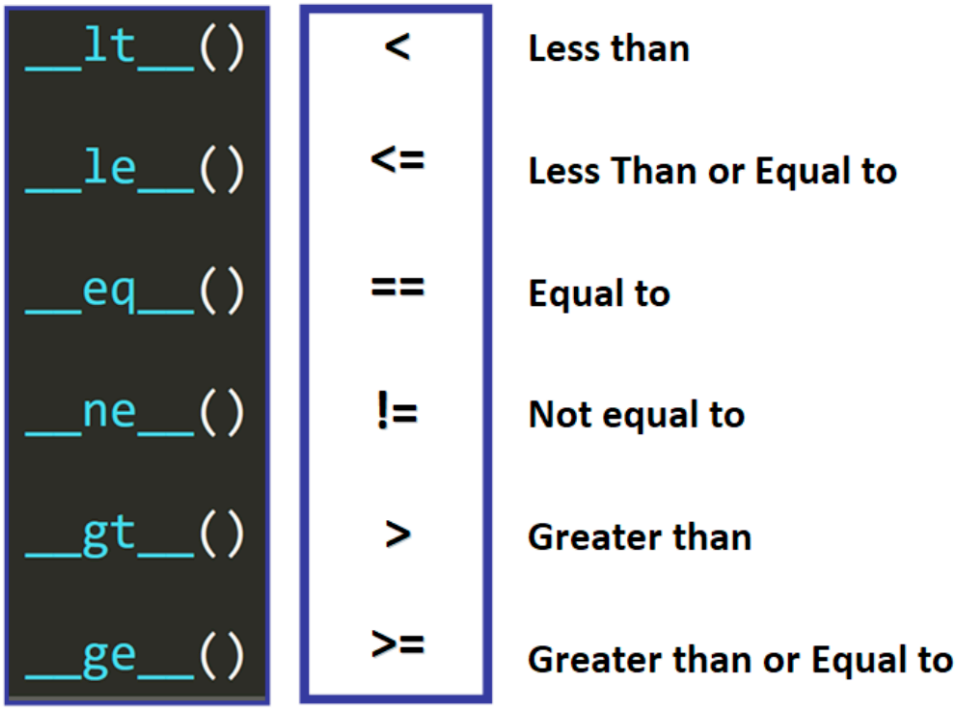


**Fig. 3 Implementing the Special Method __add__()**

```
class Coordinate:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __add__(self, other):
        """Add two Coordinate objects."""
        return Coordinate(self.x + other.x, self.y + other.y)

    def __str__(self):
        return f"({self.x}, {self.y})"
```