

`__init__` => when object is created.
`__add__` => this is called when we use + operator between 2 object of a class.

```
In [83]: class Complex:
          def __init__(self, real, imag):
              self.real = real
              self.imag = imag

          def __add__(self, other):
              return Complex(self.real+other.real, self.imag+other.imag)
```

In [84]: `c1 = Complex(1,2)`

In [85]: `c2 = Complex(2,3)`

In [87]: `c3 = c1+c2`

In [88]: `print(c3.real, c3.imag)`

Handwritten annotations:
 - Arrows from `__init__` to `Complex(1,2)` and `Complex(2,3)`.
 - Arrows from `__add__` to `c1+c2`.
 - Brackets under `self.real+other.real` and `self.imag+other.imag` pointing to `3` and `5` respectively.
 - `Complex(3,5)` written below the brackets.
 - Arrows from `3` and `5` to the output of `print` in [88].

$$\begin{array}{r} C1 = 1 + 2i \\ C2 = 2 + 3i \\ \hline C3 = 3 + 5i \end{array}$$

`__add__` is automatically called when it sees + operator being used around Complex obj.

`__init__` => when object is created.

`__add__` => this is called when we use + operator between 2 object of a class.

```
In [83]: class Complex:
        def __init__(self, real, imag):
            self.real = real
            self.imag = imag

        def __add__(self, other):
            return Complex(self.real+other.real, self.imag+other.imag)
```

Handwritten notes: The `__init__` method is circled in blue with two arrows pointing to it from the right. The `__add__` method is also circled in blue. A large blue arrow labeled "automatically" points from the `__add__` method to the `c1 + c2` expression in the next block. The `return` statement in `__add__` is also circled in blue.

$C_1 = (2, 3)$
 $C_2 = (1, 2)$
 $C_1 + C_2$

```
In [84]: c1 = Complex(1, 2)
```

Handwritten notes: The line `c1 = Complex(1, 2)` is circled in blue. Arrows labeled C_1 and C_2 point from the `c1` and `c2` variables to the `Complex` constructor in the `__add__` method.

```
In [85]: c2 = Complex(2, 3)
```

```
In [87]: c3 = c1 + c2
```

```
In [88]: print(c3.real, c3.imag)
```

3 5