Lecture

Intro to Special Methods

Special Methods

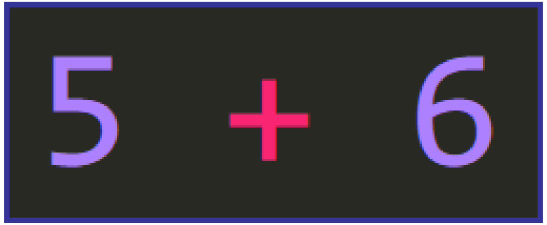
♦ Add “magic” to your class.

♦ You can customized them,

but you do not call them directly.

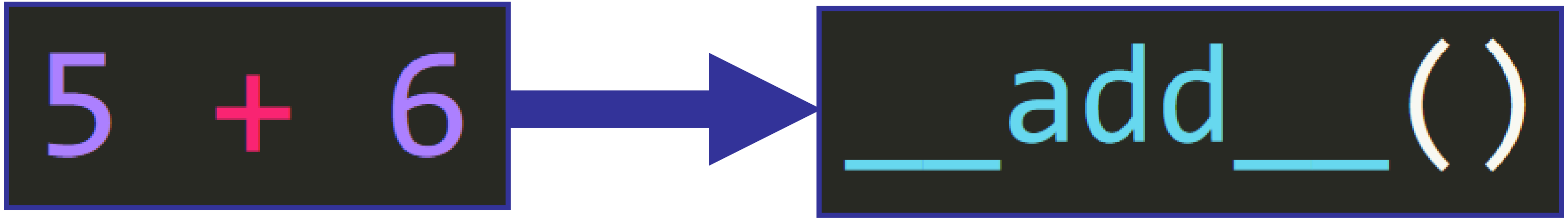
♦ They run as the result of an **action**.

Special Methods



```
5 + 6
```

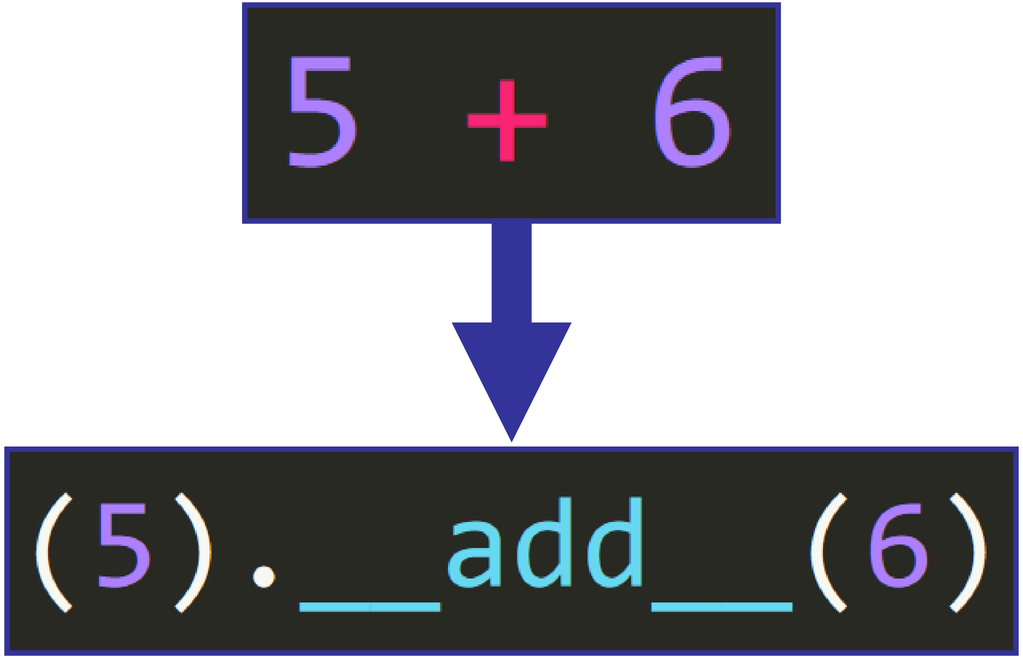
Special Methods



```
5 + 6
```

```
__add__()
```

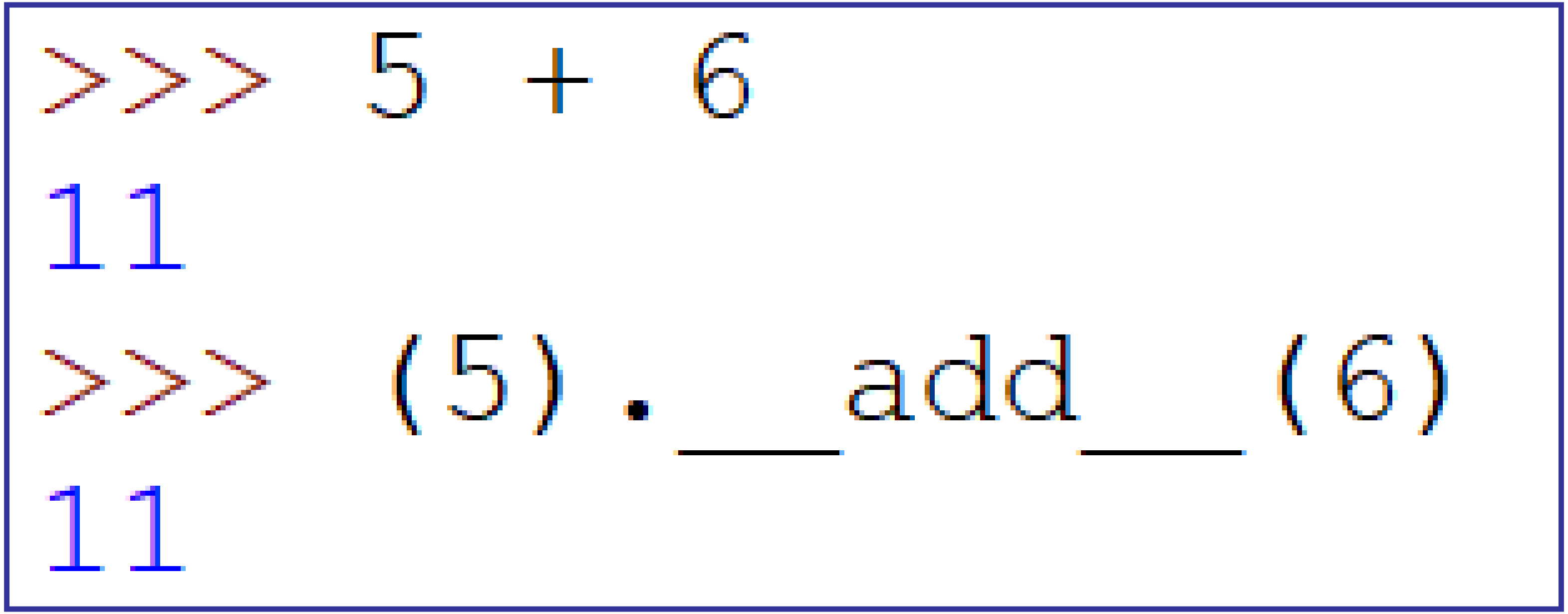
Special Methods



```
5 + 6
```

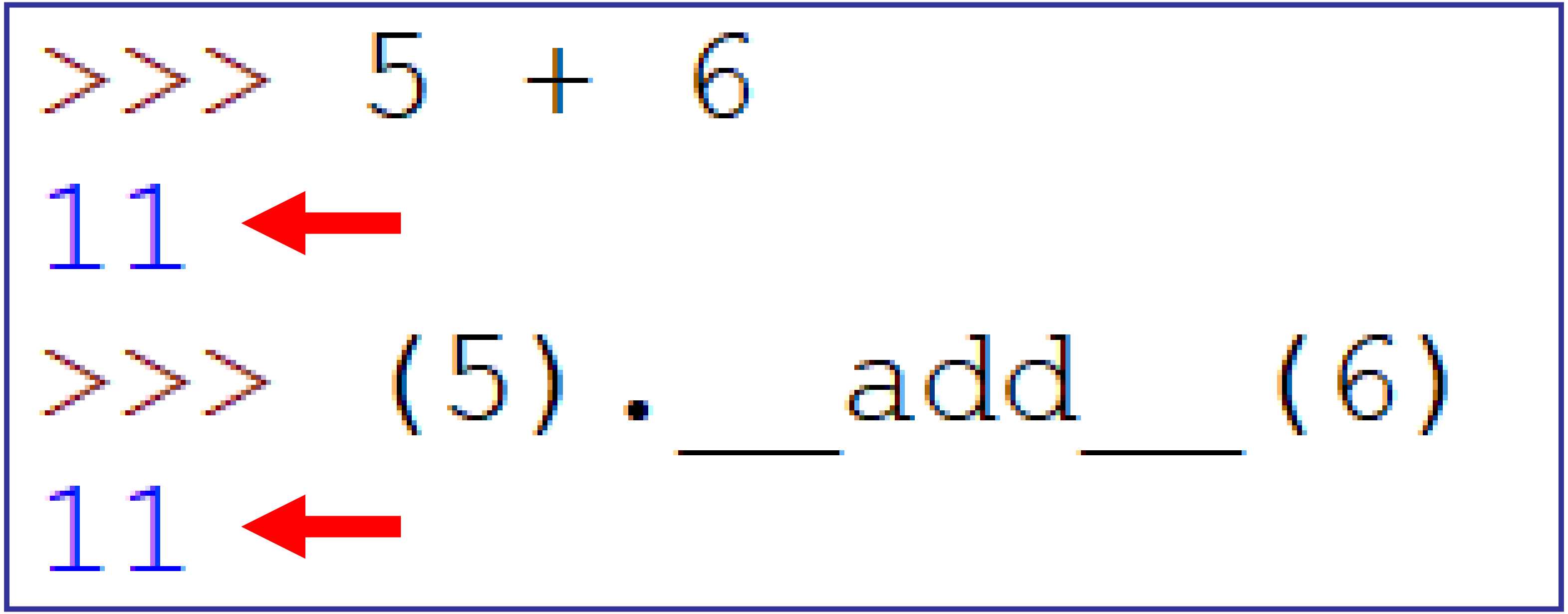
```
(5).__add__(6)
```

Special Methods



```
>>> 5 + 6
11
>>> (5).__add__(6)
11
```

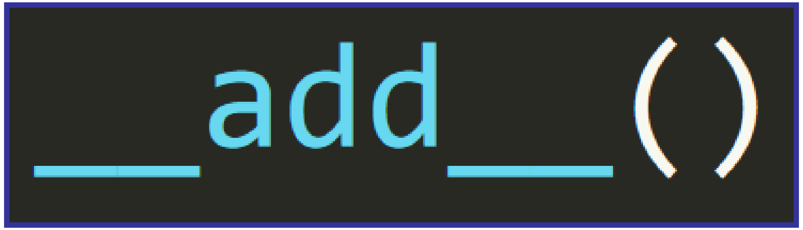
Special Methods



```
>>> 5 + 6
11
>>> (5).__add__(6)
11
```

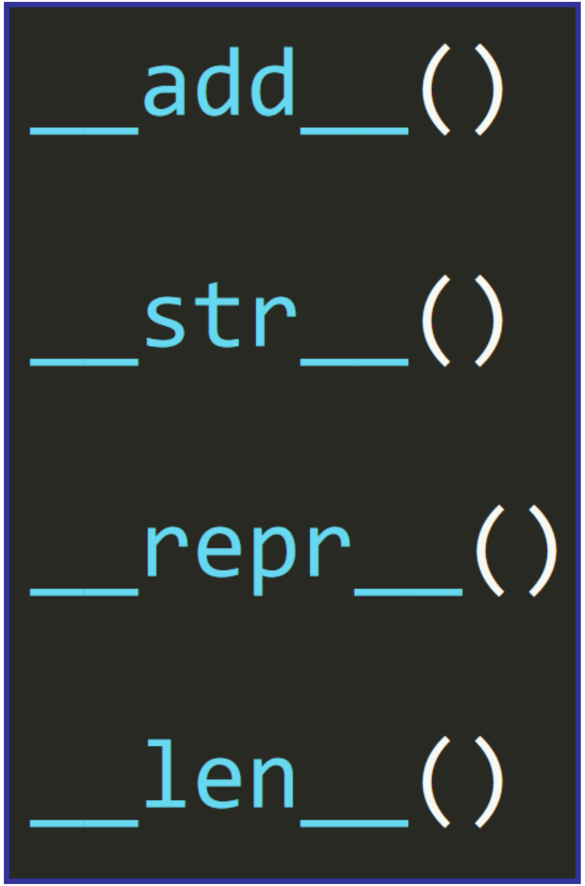
Special Methods

Strange syntax?



```
__add__()
```

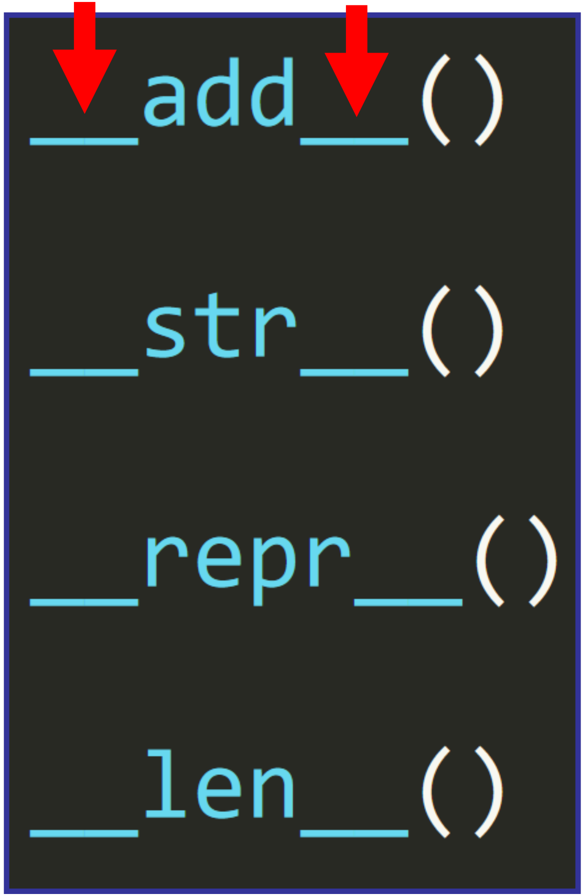
Special Methods



```
__add__()

__str__()

__repr__()

__len__()
```

Special Methods

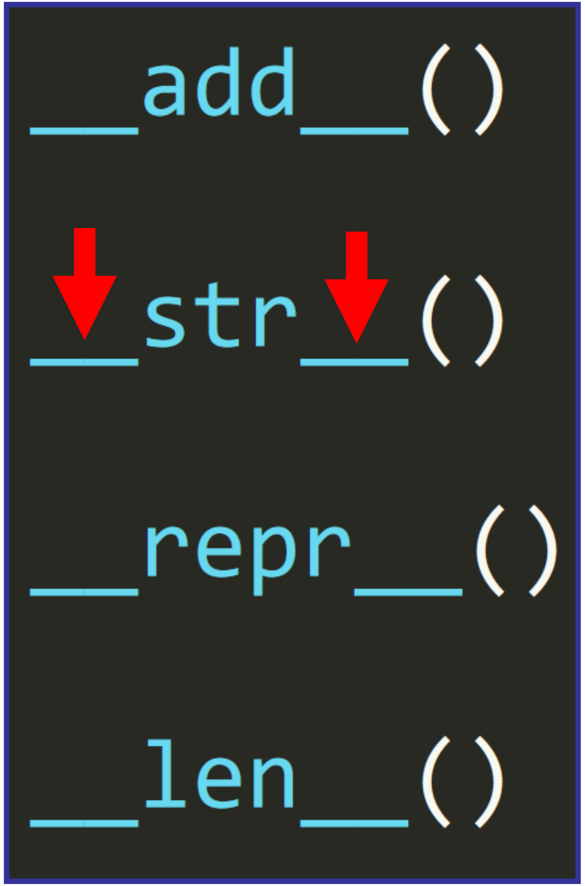
Double leading and trailing underscores



```
__add__()

__str__()

__repr__()

__len__()
```

Special Methods

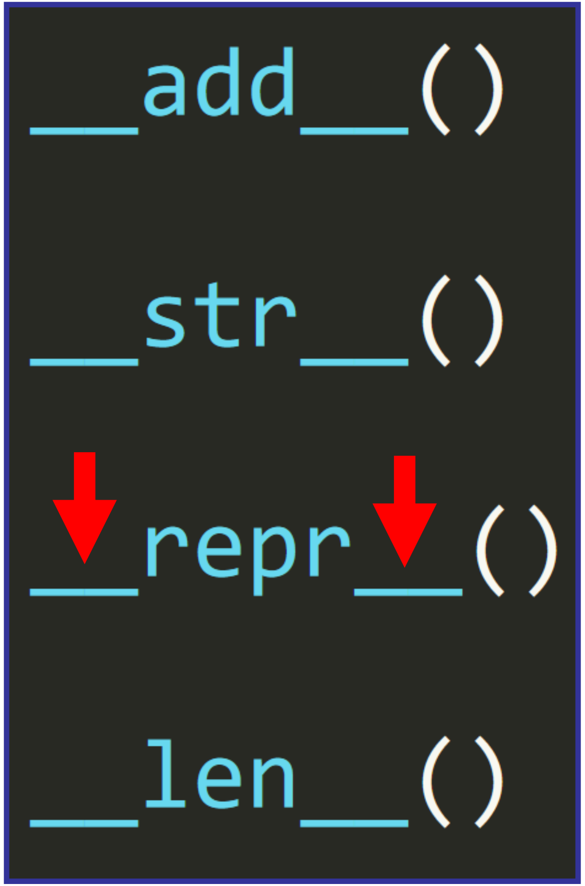
Double leading and trailing underscores



```
__add__()

__str__()

__repr__()

__len__()
```

Special Methods

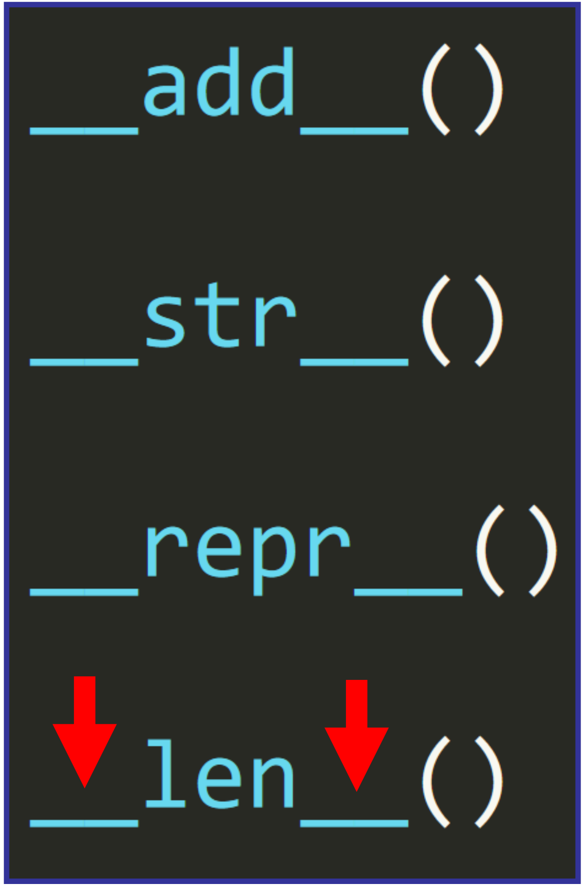
<u>Double</u>
leading
and
trailing
underscores



```
__add__()

__str__()

__repr__()

__len__()
```

Special Methods

<u>Double</u>
leading
and
trailing
underscores



```
__add__()

__str__()

__repr__()

__len__()
```

Special Methods

```python
class Dog:

    def __init__(self, name, age):
        self.name = name
        self.age = age

    def __str__(self):
        return f"I'm a Dog. My name is {self.name}"
```

Special Methods

```
class Dog:

    def __init__(self, name, age):
        self.name = name
        self.age = age

    def __str__(self):
        return f"I'm a Dog. My name is {self.name}"
```

Documentation