03

A classe Retangulo e encapsulamento

Veja a classe `Retangulo` abaixo:

```
class Retangulo:

    def __init__(self, x, y):
        self.__x = x
        self.__y = y
        self.__area = x * y

    def obter_area(self):
        return self.__area
```

Assumindo que a classe foi carregada corretamente, podemos executar o seguinte código:

```
r = Retangulo(7,6)
r.area = 7
r.obter_area()
```

Qual é o resultado da execução? Se tiver com dúvida, faça o teste!

A erro de execução

B 7

C 6

✓ 42

Correto! A classe `Retangulo` está correta e define os atributos `__x` , `__y` e `__area` , além do construtor e o método `obter_area()` . Nada impede então criar um objeto:

```
r = Retangulo(7,6)
```

Agora, se você tenta acessar um atributo `area` , que na verdade não declaramos, o Python cria automaticamente um novo atributo e inicializa com o valor 7:

```
r.area = 7
```

Na linha em cima o objeto *ganha um novo atributo* com o nome `area` . Ou seja, temos um atributo `__area` E um novo com o nome `area` . No entanto, ao chamar `r.obter_area()` , continuamos acessar o atributo `__area` que foi inicializado com o produto de `7*6` !

PRÓXIMA ATIVIDADE