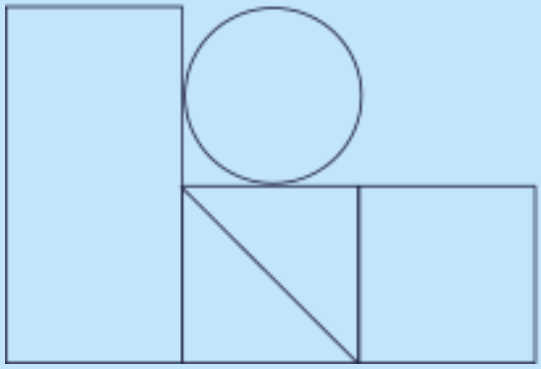
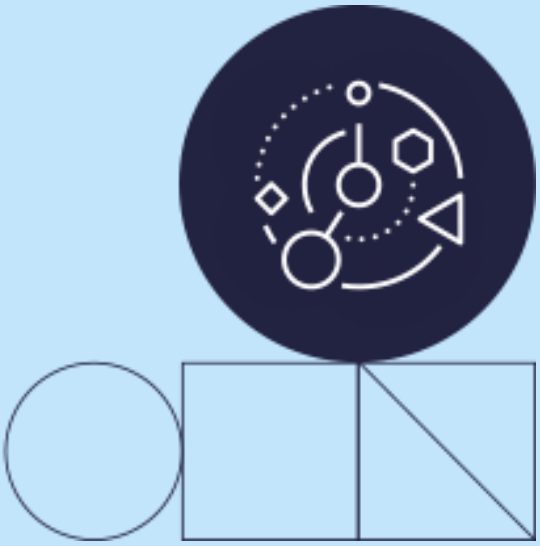
IBM SkillsBuild

IBM

OOP com Python

Construtores e destruidores





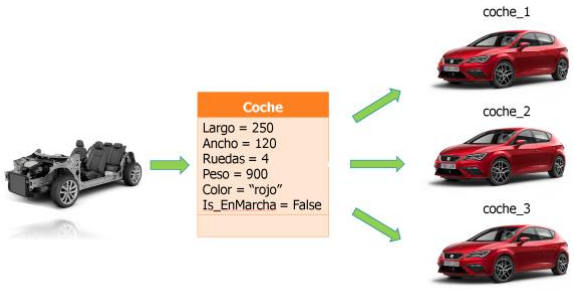
Índice

Construtores

Um construtor é definido como um método especial com o qual damos um estado inicial aos nossos objetos.

Continuando com o exemplo de nossa classe Carro, podemos criar um construtor com o qual damos um estado inicial aos objetos da classe Carro que criamos, de tal forma que cada carro criado já tem uma série de atributos e métodos "por padrão". Mais tarde podemos mudá-los se quisermos, mas "por padrão" todos os carros sairão com os atributos e métodos que colocamos no construtor.

Por exemplo:



A sintaxe para a criação de um construtor é:

```
def __init__(self):
```

Lembramos que mencionamos que um construtor é um método, daí uma função. Dentro desta função, devemos colocar os atributos e métodos iniciais que queremos que nossas instâncias de classe tenham.

Vamos passar o exemplo gráfico do carro para o código:

```
# Creación de la clase Coche
class Coche():

# Declaración del constructor de la clase
Coche
    def __init__(self):
        self.largo = 250
        self.ancho = 120
        self.ruedas = 4
        self.peso = 900
        self.color = "rojo"
        self.is_enMarcha = False

# Declaración de métodos
    def arrancar(self):     #self hace
referencia a la instancia de clase.
        self.is_enMarcha = True    #Es como
si pusiésemos miCoche.is_enMarcha = True

    def estado(self):
        if (self.is_enMarcha == True):
            return "El coche está
arrancado"
        else:
            return "El coche está
parado"

# Declaración de una instancia de clase,
objeto de clase o ejemplar de clase.
coche_1 = Coche()

# Acceso a un atributo de la clase Coche.
Nomenclatura del punto.
coche_1.ruedas = 7
print("El largo del coche es de" ,
coche_1.largo, "cm.")
coche_1.arrancar()
print(coche_1.estado())

# Acceso a un método de la clase Coche.
Nomenclatura del punto.
print("El coche está arrancado:" ,
coche_1.arrancar())
```

Criamos a classe **Carro** e um construtor com o qual damos um estado inicial aos objetos que criamos nessa classe. Em nosso caso, todos os carros que criamos vão ter:

- comprimento = 250
- largura = 120
- rodas = 4
- peso = 900
- cor = "vermelho"
- is_enMarcha = False.

Como mencionado acima, este será o estado inicial. Podemos mudar isso mais tarde e dizer, por exemplo, que **coche_1.color = “verde”** e, a partir desse momento, a cor do **coche_1** valerá **verde**, mas o padrão era **rojo** como o de todos os carros que criamos com nossa classe **Carro**.

Cada classe Python deve ter um construtor Python, é obrigatório. Mas então, como criamos anteriormente os exemplos das aulas de automóveis dos capítulos anteriores? Se não usássemos construtores, eles foram programados erroneamente? A resposta é não. Nos exemplos anteriores não utilizamos construtores, mas isso não produziu nenhum bug porque, se não criarmos um construtor, Python cria um para nós por padrão. Ele o cria vazio, mas ele o cria.

Portanto, como regra geral, sempre criaremos o construtor de nossas classes, mas se não o fizermos, Python o criará para nós. No entanto, ele o cria vazio, sem atributos ou métodos padrão.

Em nosso exemplo da classe **Carro**, demos um conjunto fixo de atributos a todos os espécimes do **carro** nós criamos. Mas podemos criar nosso construtor de uma forma mais flexível e aceitar que decidamos o valor de cada um desses atributos quando instanciamos um objeto **carro**. Ou seja, diremos ao construtor os atributos que uma cópia do carro tem que receber, sem dar um valor a esses atributos no momento, e será quando instanciarmos essa cópia do carro quando lhe diremos o valor de cada atributo. Isto é chamado de construtor com parâmetros.

Vamos ver com um exemplo:

```
class Coche:

# Declaración del constructor con
parámetros
    def __init__(self, largo, ancho,
ruedas, peso, color, is_enMarcha):
        self.ancho = ancho
        self.ruedas = ruedas
        self.peso = peso
        self.color = color
        self.is_enMarcha = is_enMarcha


# Declaración de dos instancias de clase pasándoles
los parámetros requeridos en el constructor.

coche_1 = Coche(400, 160, 4, 1200,
"amarillo", True)
coche_2 = Coche(420, 150, 4, 1500,
"verde", False)
```

Como podemos ver no exemplo, criamos uma classe de carro e em seu construtor dissemos que, quando uma instância de carro é criada, o valor dos parâmetros deve ser passado para ela.; **largo**, **ancho**, **ruedas**, **peso**, **color**, **is_enMarcha**.

Somos obrigados a passar estes parâmetros (e exatamente na mesma ordem) ao instanciar uma amostra de carro, caso contrário, receberemos um erro.

Se mudássemos a declaração de **coche_2** e não passamos um parâmetro obrigatório, por exemplo **is_enMarcha** receberíamos uma mensagem como a seguinte:

```
coche_2 = Coche(420, 150, 4, 1500,
"verde")
```

```
---> 21 coche_2 = Coche(420, 150, 4, 1500, "verde")

TypeError: Coche.__init__() missing 1 required posi
```

Como podemos ver, o programa falhou e nos é mostrada explicitamente a razão do fracasso. Em nosso caso, nos diz que nos falta um argumento (**is_enMarcha**) que está no construtor, mas não o colocamos na questão de **carro**.

Assim, o método construtor nos permitirá atribuir atributos toda vez que criarmos um objeto daquela classe, tornando-o obrigatório. E também nos permitirá chamar métodos da classe sem instanciar, entre outras coisas.

Destruidores

Assim como existe um método construtor, existe um método destruidor cuja tarefa é remover instâncias de uma classe. Ou seja, ele remove um objeto.

O método destruidor é o **__del__()**

```
class Book():
  """
  Clase para trabajar con libros
  """

  def __init__(self, title, author = "",
electronic = False):
    self.title = title
    self.author = author
    self.is_electronic = electronic

  def __del__(self):
    print("Acabas de llamar al método
destructor. El objeto acaba de ser
eliminado")
```

Para apagar um objeto, use a palavra reservada **del**

```
book = Book("Lazarillo de Tormes")
book.title

del book
```

Se tentássemos acessar o objeto **book**, receberíamos um erro porque não é mais uma instância de classe **Book** porque nós o eliminamos.