Programação Orientada à Objetos: Construtores

Processo de Abstração:

1. Definição de Classes (criação de atributos e métodos)
2. Criação de instâncias de classes (Objetos)
3. Inicialização dos atributos das instancias de classes (Objetos)
4. Utilização (chamada) dos métodos dessas instâncias(Objetos)

Programação Orientada à Objetos: Construtores

3. Inicialização dos atributos das instancias de classes (Objetos)

Por que devemos inicializar os atributos de uma instância de classe?

O que pode dar errado se não fizermos?

```python
class Gato:
    raça  = None
    nome  = None
    peso  = None
    idade = None
    def mudar_nome(self,nome):
        self.nome=nome
    def engordar(self,peso):
        self.peso+=peso
    def envelhecer(self):
        self.idade +=1
```

```python
meu_gato=Gato()
meu_gato.mudar_nome("tom")
meu_gato.engordar(3)
meu_gato.envelhecer()
print("nome:",meu_gato.nome)
print("peso:{}".format(meu_gato.peso))
print("idade:{}".format(meu_gato.idade))
```

```
Traceback (most recent call last):
  File "/home/rogerio/PycharmProjects/POO_266/gato.py", line 15, in <modul
    meu_gato.engordar(3)
  File "/home/rogerio/PycharmProjects/POO_266/gato.py", line 9, in engorda
    self.peso+=peso
TypeError: unsupported operand type(s) for +=: 'NoneType' and 'int'
```

```
meu_gato=Gato()
print("nome:",meu_gato.nome)
print("peso:{}".format(meu_gato.peso))
print("idade:{}".format(meu_gato.idade))
```

Omitindo a chamada dos métodos...

```
nome: None
peso:None
idade:None

Process finished with exit code 0
```

tem sentido?

Programação Orientada à Objetos: Construtores

Construtores:

- Métodos especiais utilizados para inicializar os dados (atributos) principais de uma instancia de classe (objeto) no momento da criação dessa instância.

- O compilador python não obriga essa inicialização.
- Cabe ao programador definir quais são esses atributos obrigatórios.

Programação Orientada à Objetos: Construtores

Construtores:

- Em python, o construtor de uma classe é definido pelo método interno: __init__(**self**).

se refere ao próprio objeto

```
def __init__(self,valor_1,valor_2,...,valor_n):
    self.atributo_1 = valor_1
    self.atributo_2 = valor_2
    ...
    self.atributo_n = valor_n
```

Programação Orientada à Objetos: Construtores

- O método __init__(self) é responsável por inicializar o objeto, tanto é que já recebe a própria instância (self) criada pelo construtor como argumento.

- Dessa maneira garantimos que toda instância de uma Classe tenha os atributos que definimos.

definindo o construtor para a classe Gato...

```
class Gato:
    raça  = None
    nome  = None
    peso  = None
    idade = None
    def __init__(self,peso,idade):
        self.peso = peso
        self.idade = idade

    def mudar_nome(self,nome):
        self.nome=nome
    def engordar(self,peso):
        self.peso+=peso
    def envelhecer(self):
        self.idade +=1
```

criando um objeto do tipo Gato...

```
meu_gato=Gato()
```

```
Traceback (most recent call last):
  File "/home/rogerio/PycharmProjects/POO_266/gato.py", line 17, in <module
    meu_gato=Gato()
TypeError: __init__() missing 2 required positional arguments: 'peso' and '
```

O que aconteceu com a execução?

criando um objeto do tipo Gato de forma correta...

```
17  meu_gato=Gato(0.1,0)
18  print("nome:",meu_gato.nome)
19  print("peso:{}".format(meu_gato.peso))
20  print("idade:{}".format(meu_gato.idade))
21  meu_gato.mudar_nome("mimi")
22  meu_gato.engordar(3)
23  meu_gato.envelhecer()
24  print("nome:",meu_gato.nome)
25  print("peso:{}".format(meu_gato.peso))
26  print("idade:{}".format(meu_gato.idade))
```

resultado da execução ...

```
nome: None    # resultado da linha 18
peso:0.1      # resultado da linha 19
idade:0       # resultado da linha 20
nome: mimi    # resultado da linha 24
peso:3.1      # resultado da linha 25
idade:1       # resultado da linha 26

Process finished with exit code 0
```

Programação Orientada à Objetos: Construtores

Exercícios