

Design de Software

Aula 16 – Classes e Objetos 2

2016 - Engenharia

Fábio Ayres <fabioja@insper.edu.br>

Raul Ikeda <rauligs@insper.edu.br>

Objetivo da Aula

- Organizar código em Classes e Objetos
- Codificar em Python

Classes

- Classes são formas de agregar dados e funcionalidades similares
- *Encapsular* (agrupar e isolar) dados (*atributos*) e funcionalidades (*métodos*) melhora a modularização do código e facilita seu reuso

Abstração

Ao se criar uma classe pense nas seguintes perguntas:

- O que objetos dessa classe irão representar?
- Quais são seus métodos?
- Quais são seus atributos?

Exemplo: Veículo

- Vamos desenvolver um programa de computador para simular veículos.



Exemplo: Veículo

Veiculo
peso: float potencia: float
relação_peso_potência(): float

```
1 class Veiculo:
2     """ Classe que representa veículos """
3
4     def __init__(self, peso, potencia, carga):
5         self.peso = peso
6         self.potencia = potencia
7         self.carga = carga
8
9     def relacao_peso_potencia(self):
10         return self.peso / self.potencia
11
```

Exemplo: Veículo

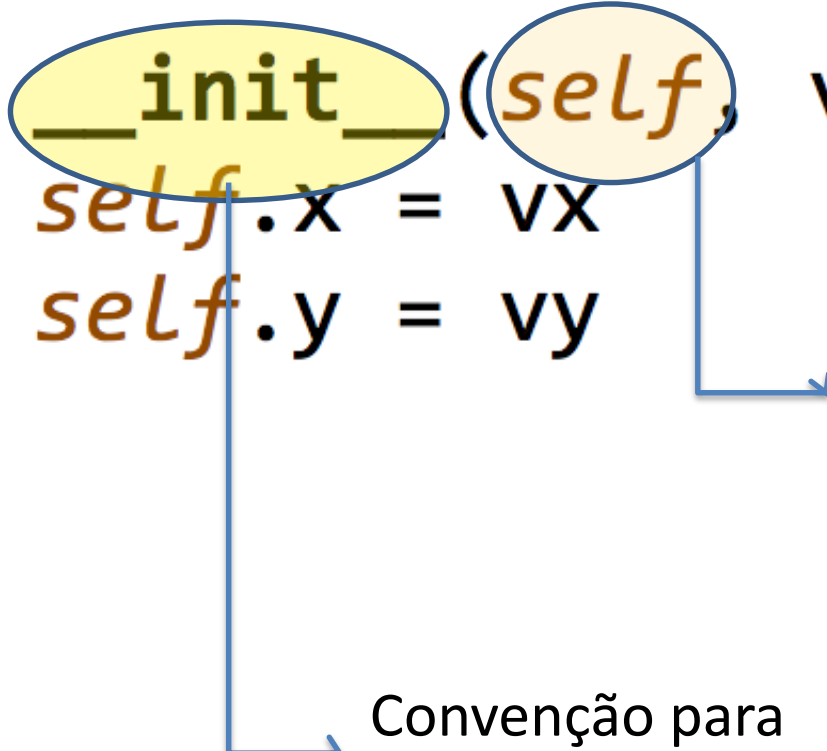
- Construtores são chamados na construção do objeto e podem receber argumentos.

```
8 class Veiculo:
9     """ Classe que representa veículos """
10
11     def __init__(self, peso, potencia, carga):
12         self.peso = peso
13         self.potencia = potencia
14         self.carga = carga
15
```

```
16 veiculo1 = Veiculo(170, 900, 4)
17 veiculo2 = Veiculo(97, 1200, 2.5)
```

Método construtor

```
def __init__(self, vx, vy):  
    self.x = vx  
    self.y = vy
```



O próprio objeto –
variável passada
automaticamente

Convenção para
identificar o
construtor

Exercício

Faça uma classe Barata com as seguintes especificações:

- Atributos:
 - ligado (atributo Booleano),
 - saudação (atributo string),
 - potência (atributo float)
- Métodos:
 - diz_oi(): imprime a saudação no terminal
 - liga(): muda ligado para True
 - desliga(): muda ligado para False
 - muda_potência(nova_potência): altera a potência. Obs: e se a nova potência for negativa?

Solução

```
class Barata:
    def __init__(self, está_ligado, minha_saudação, minha_potência):
        self.ligado = está_ligado
        self.saudação = minha_saudação
        self.potencia = minha_potência

    def liga(self):
        self.ligado = True

    def desliga(self):
        self.ligado = False

    def diz_oi(self):
        if self.ligado:
            print(self.saudação)

    def muda_potência(self, nova_potência):
        self.potência = nova_potência
```

Exemplo: Veículo

- Os objetos criados podem ser usados como variáveis normalmente
 - Por exemplo se pode adicionar em uma lista:

```
8 class Veiculo:
9     """ Classe que representa veículos """
10
11     def __init__(self, peso, potencia, carga):
12         self.peso = peso
13         self.potencia = potencia
14         self.carga = carga
15
16     def relacao_peso_potencia(self):
17         return self.peso / self.potencia
18
19 veiculo1 = Veiculo(170,900,4)
20 veiculo2 = Veiculo(97,1200,2.5)
21 veiculos = [veiculo1, veiculo2]
```

Exemplo: Veículo

- Cuidado!

```
8 class Veiculo:
9     """ Classe que representa veículos """
10
11     pneus = [] #Atributo estático
12
13     def __init__(self, peso, potencia, carga):
14         self.peso = peso
15         self.potencia = potencia
16         self.carga = carga
```

Este atributo é compartilhado por **todos os objetos** da mesma classe!!

Estes atributos são pessoais do objeto

```
18 veiculo1 = Veiculo(170, 900, 4)
19 veiculo1.pneus.append("FL")
20
21 veiculo2 = Veiculo(97, 1200, 2.5)
22 print(veiculo2.pneus) #Qual vai ser a saída?
```



['FL']

Exercício: Veículo

- Use um loop for para somar o peso de todos os veículos.

```
1 class Veiculo:
2     """ Classe que representa veículos """
3
4     def __init__(self, peso, potencia, carga):
5         self.peso = peso
6         self.potencia = potencia
7         self.carga = carga
8
9     def relacao_peso_potencia(self):
10         return self.peso / self.potencia
11
12
13 veiculo1 = Veiculo(900, 170, 2)
14 veiculo2 = Veiculo(1200, 97, 2.5)
15
16 veiculos = [veiculo1, veiculo2]
17
```

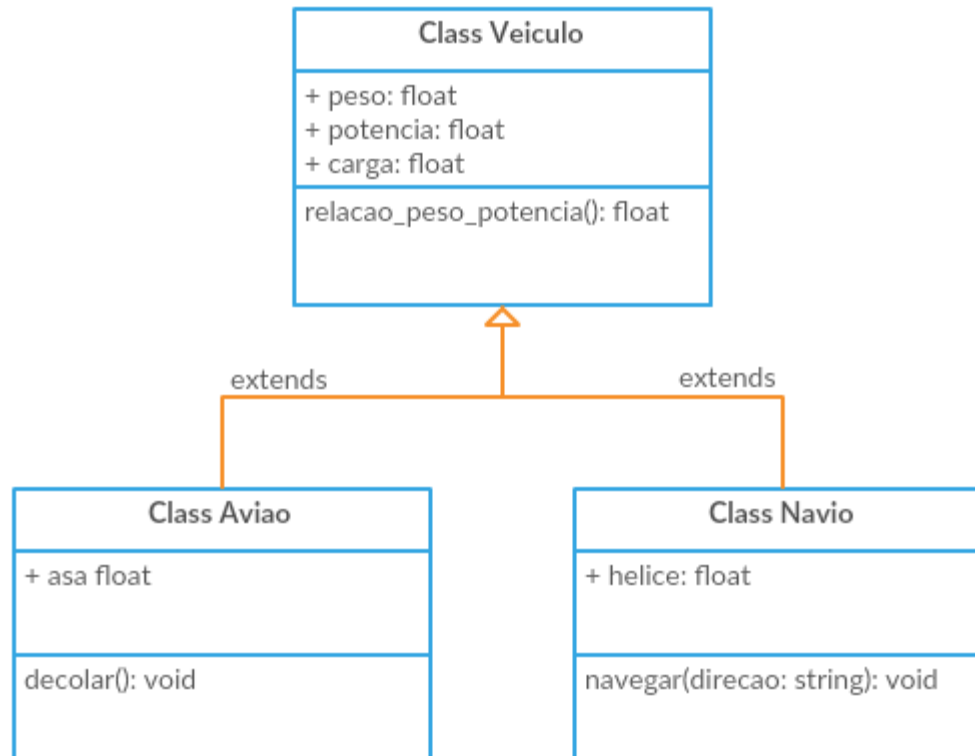
Exercício: Veículo

- Crie o método ***getPotenciaKW*** que retorne o valor da potência em quilowatt.

$$1 \text{ CV} = 736 \text{ W}$$

Exemplo: Veículo

- Classes podem se especializar por herança.



Exemplo: Veículo

- Classes podem se especializar por herança.

```
8 class Veiculo:
9     """ Classe que representa veículos """
10
11     def __init__(self, peso, potencia, carga):
12         self.peso = peso
13         self.potencia = potencia
14         self.carga = carga
15
16 class Aviao(Veiculo):
17     """ Sub-classe que representa aviões """
18     def __init__(self, peso, potencia, carga, asa):
19         self.asa = asa #tamanho da asa
20         Veiculo.__init__(self, peso, potencia, carga)
21
22 class Navio(Veiculo):
23     """ Sub-classe que representa navios """
24     def __init__(self, peso, potencia, carga, helice):
25         self.helice = helice #diametro da helice
26         super().__init__(peso, potencia, carga)
```


Pergunta 1

O que o seguinte programa retorna na tela:

```
1 class Relogio:
2     def __init__(self, hora, minuto, segundo):
3         self.hora = hora
4         self.minuto = minuto
5         self.segundo = segundo
6     def adiantar(self, tempo):
7         self.hora = (self.hora + tempo)%24
8
9 t = Relogio(23, 10, 30)
10 t.adiantar(2)
11 print(t.hora, ":", t.minuto, ":", t.segundo)
```

A) 24 : 10 : 30

B) 1 : 10 : 30

C) 25 : 10 : 30

D) t : t : t

E) Não sei

Pergunta 2

O seguinte programa funciona corretamente?

VERDADEIRO

FALSO

```
1 class Pessoa:
2     def __init__(nome, idade):
3         self.nome = nome
4         self.idade = idade
5
6 p1 = Pessoa("Jose", 20)
7 p2 = Pessoa("Mane", 30)
8
9 pessoas = {}
10 pessoas[p1.nome] = p1
11 pessoas[p2.nome] = p2
12
13 print(pessoas["Jose"].idade)
```

Pergunta 3

O que aparece no console?

```
1 class Animal:
2     def falar(self):
3         print("Nada")
4
5 class Gato(Animal):
6     def falar(self):
7         print("Miau")
8
9 class Cachorro(Animal):
10    def falar(self):
11        print("AuAu")
12
13 bicho = Gato()
14 bicho.falar()
```

- Importante:
 - Continuem a pensar nas 15 ideias INDIVIDUALMENTE.
 - Áreas:
 - Jogos
 - Aplicativos Web
 - Aplicativos Desktop (Cliente/Servidor)
 - Aplicativos Científicos

Insper

www.insper.edu.br