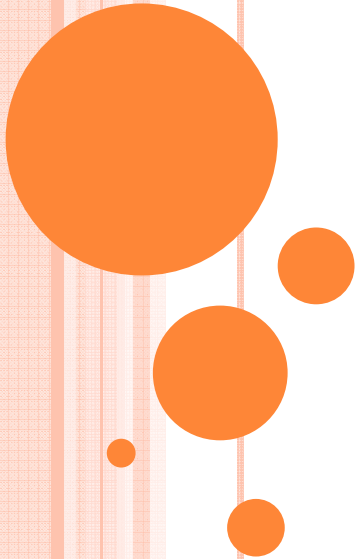


Introdução a Classes e Orientação a Objetos em Python

George Gomes Cabral

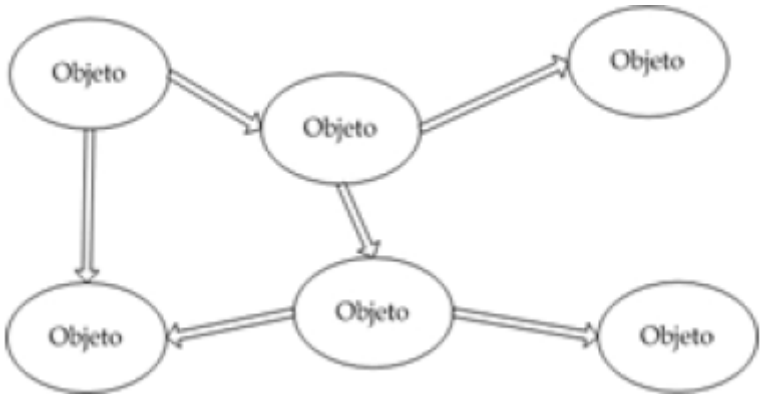
O paradigma da Orientação a Objetos

- Um paradigma é uma forma de abordar um problema.
- Em Python, tudo é objeto !!
- Objetos armazenam dados, mas você também pode fazer requisições a esse objeto, pedindo que ele faça operações sobre si próprio ou sobre outros objetos.
- Teoricamente, você pode representar qualquer elemento conceitual no problema que você está tentando resolver (cachorros, livros, sócios, empréstimos, etc.) como um objeto no seu programa.

O paradigma da Orientação a Objetos

- Um programa é uma coleção de objetos dizendo uns aos outros o que fazer.

- Para fazer uma requisição a um objeto você "manda uma mensagem" para este objeto. Mais concretamente, você pode pensar em uma mensagem como sendo uma chamada de um procedimento ou função pertencente a um objeto em particular.



O paradigma da Orientação a Objetos

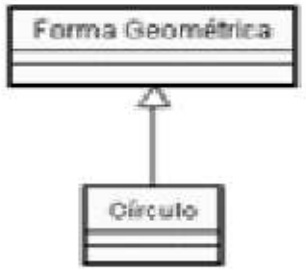
- Um objeto pode ser composto por vários outros objetos

- Em outras palavras: você pode criar um novo tipo de objeto empacotando objetos já existentes. Dessa forma, você pode adicionar complexidade a um programa e escondê-la por trás da simplicidade de uso dos objetos.

O PARADIGMA DA ORIENTAÇÃO A OBJETOS

- Todos os objetos de um dado tipo podem receber as mesmas mensagens.

- Além disso, uma vez que, por exemplo, um objeto do tipo “círculo” é também um objeto do tipo “forma geométrica”, o objeto “círculo” aceita qualquer mensagem endereçada a uma “forma geométrica”. Essa capacidade de “substituição” de um objeto por outro é um dos mais poderosos conceitos em orientação a objetos.



Objeto

- Definição:
  - Um objeto é qualquer coisa, real ou abstrata, sobre a qual armazenamos dados e realizamos operações que manipulam tais dados.
- Unidade básica de modularização do sistema na abordagem OO.
- Um objeto é composto por:
  - Atributos características ou propriedades que definem o objeto.
  - Comportamento conjunto de ações pré-definidas (métodos).

ABSTRAÇÃO

Abstraction focuses upon the essential characteristics of some object, relative to the perspective of the viewer.

Abstração

- É o mecanismo que nos permite representar uma realidade complexa em termos de um modelo simplificado, de modo que detalhes irrelevantes possam ser suprimidos.

- Processo de filtragem de detalhes sem importância do objeto, para que apenas as características apropriadas que o descrevem permaneçam.

Abstração

- Para processar algo do mundo real em um computador, temos que extrair as características essenciais.

- Esses dados que caracterizam o objeto são utilizados na representação no sistema.

- Um mesmo objeto, pode ser visualizado de formas distintas. Exemplo: Carro para a oficina, para o detran, para o consumidor.

Exemplo de Objeto

- Pássaro

- Atributos
  - Nome
  - Cor
  - Peso
- Comportamento
  - Piar
  - Voar
  - Comer

Exemplo de Objeto

- Pessoa

- Atributos
  - Nome
  - Cor da pele
  - Peso
- Comportamento
  - Falar
  - Andar
  - Comer

Exercício

- Defina um computador pessoal segundo os princípios de Orientação a Objetos:

Orientação a objetos

- A expressão orientada a objetos significa que o aplicativo é organizado como uma coleção de objetos que incorporam tanto a estrutura como o comportamento dos dados.

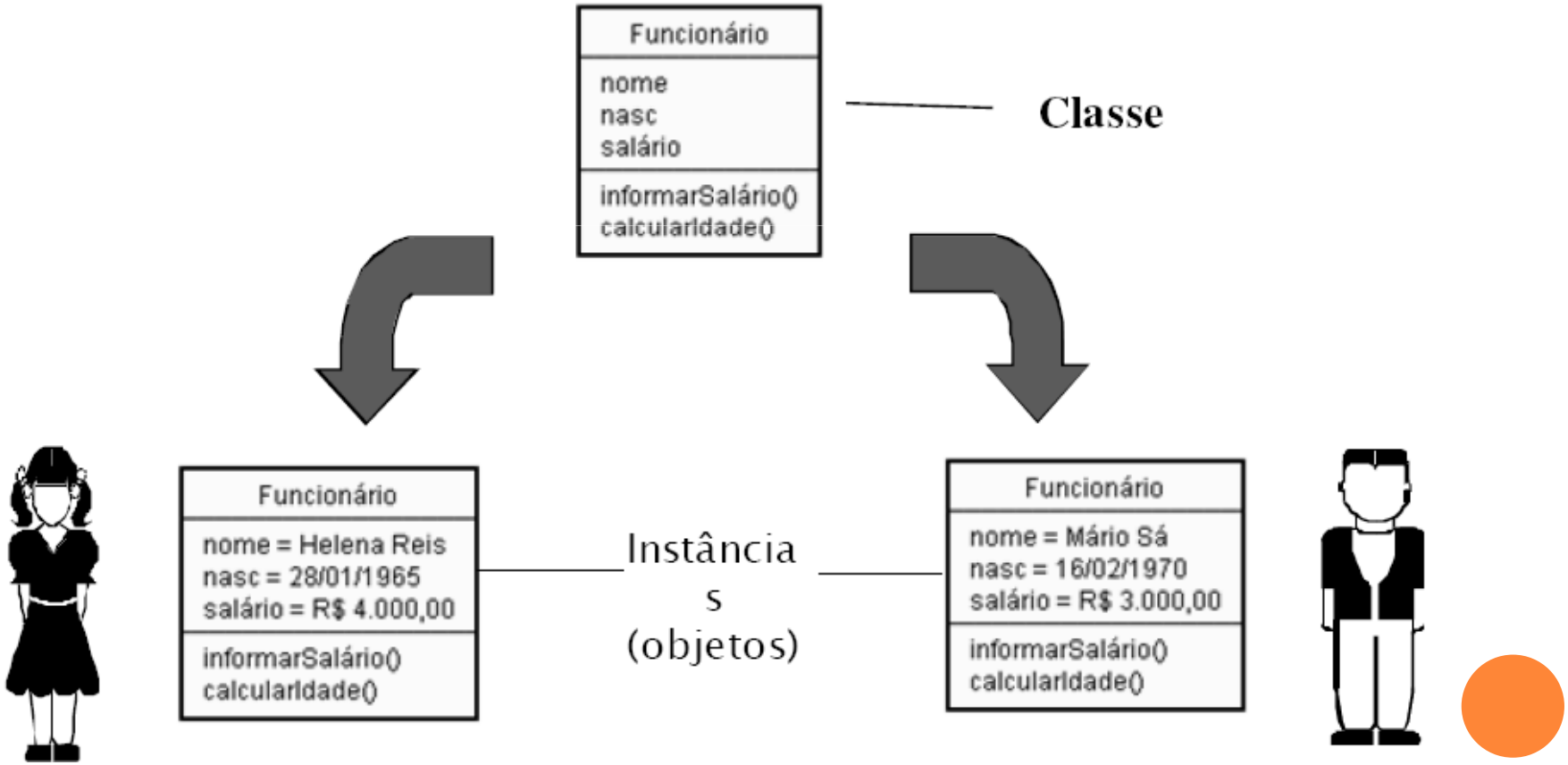
- Sistema de Controle de pizzarias:
  - Sistema que informatiza os pedidos de pizza em um restaurante.
  - Objetos:
    - Pedido,
    - Pizza,
    - Cliente,
    - Garçom

CLASSES

- **Classe**: em termos simples, são apenas uma maneira de definir novos tipos que refletem objetos reais no domínio de nossos programas.

- **Instâncias Múltiplas**: classes são fábricas para gerarem um ou mais objetos. Sempre que chamamos o construtor de uma classe, geramos um novo objeto.

CLASSES *VERSUS* INSTÂNCIAS



Herança

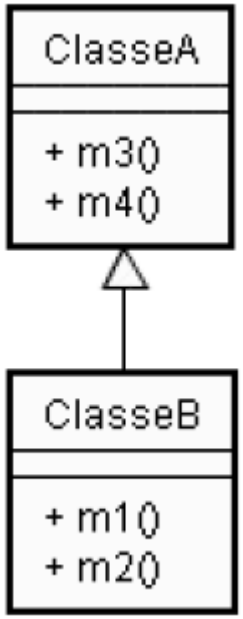
- Robôs pizzaiolos são uma espécie de robô e, assim, possuem as propriedades robóticas comuns.
  - Herdam as propriedades comuns da categoria geral de todos os robôs.
  - Propriedades precisam ser implementadas apenas uma vez para o caso geral, e reutilizadas por todos os robôs a serem construídos no futuro.

Exemplo de Herança

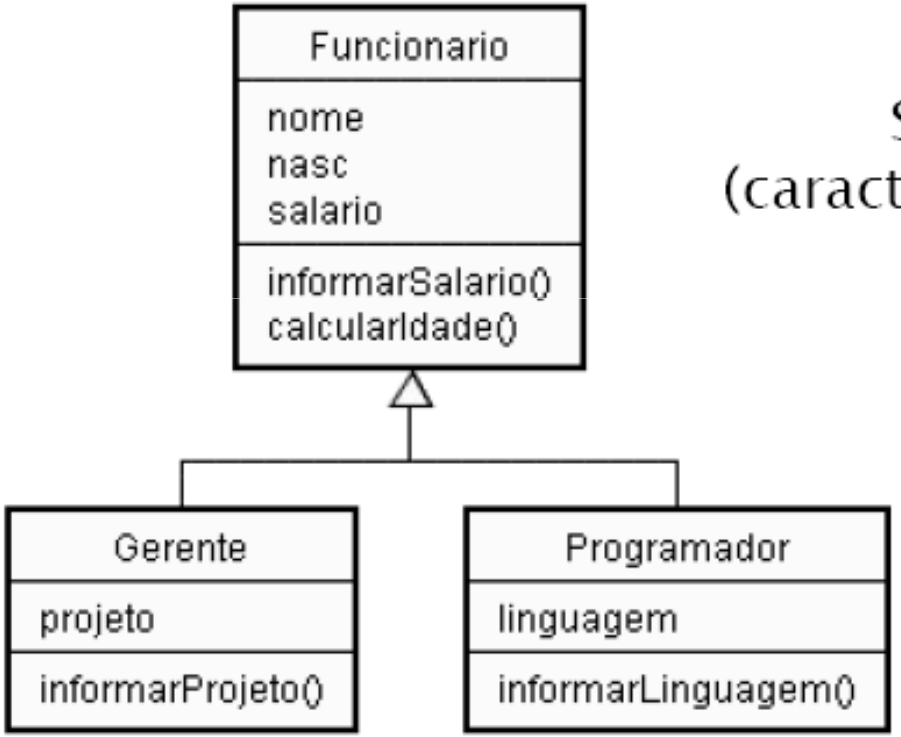
- Suponha que a classe ClasseB herda de ClasseA

- Quais métodos estão disponíveis para uma referência da ClasseB (um objeto do tipo ClasseB) ?

- Resposta: m1(), m2(), m3() e m4()



Exemplo de Herança



Superclasse
(características comuns)

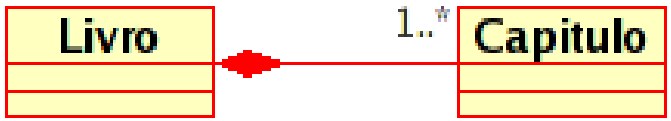
Subclasses
(características
específicas)

Composição

- Os robôs pizzaiolos são na verdade uma coleção de componentes que trabalham em conjunto como uma equipe.

- Por exemplo, ele precisa de braços para enrolar a massa, motores para manusear o forno, etc. Nosso robô é um exemplo de composição (ele contém outros objetos). Cada componente (integrante da composição) poderia ser desenvolvido como uma classe definindo seu próprio comportamento e seus relacionamentos.

- Ex.: Um livro é composto de vários capítulos. Tanto o livro quanto o capítulo podem ser representados por objetos.



Encapsulamento

- Na terminologia da orientação a objetos, diz-se que um objeto possui uma interface.
- A interface de um objeto é o que ele mostra aos outros objetos, sem descrever como o objeto conhece ou faz.
- A interface de um objeto define os serviços que ele pode realizar e conseqüentemente as mensagens que ele recebe.

Encapsulamento

- Encapsulamento é a proteção dos atributos ou métodos de uma classe.
- Em Python existem somente o público e o privado e eles são definidos no próprio nome do atributo ou método.
- Atributos ou métodos iniciados por dois sublinhados e são privados . Todas as outras formas são públicas.

Encapsulamento

```
class A:
  a = 1 # atributo publico
  __b = 2 # atributo privado a class A

class B(A):
  __c = 3 # atributo privado a B

  def __init__(self):
    print self.a
    print self.__c

a = A()
print a.a # imprime 1

b = B()
print b.__b # Erro, pois __b é privado a classe A.
print b.__c # Erro, __c é um atributo privado, somente chamado pela classe.

print b._B__c # Imprime __c = 3, muito pouco utilizada, mas existe.
```

Orientação a Objetos em Python

IMPORTANDO OBJETOS (CLASSES, FUNÇÕES, ETC.)

- Um módulo é o arquivo de código fonte propriamente dito (arquivo com extensão .py). Um módulo pode conter vários objetos, como classes, funções e variáveis. A importação desses objetos por outro módulo pode ser feita de duas maneiras:

from nome_modulo import nome_objeto

- Para acessar o objeto precisamos apenas digitar o nome do objeto. Exemplo: x = nome_objeto()
- **From nome_modulo import *** importa todos os objetos do módulo.

import nome_modulo

- Para acessar um objeto do módulo precisamos digitar o nome do módulo antes. Ex.: x = nome_modulo.nome_objeto()

Classes *versus* Módulos

Módulos

- São Pacotes de dados
- São criados escrevendo-se arquivos em Python
- São usados por meio de Importação

Classes

- Implementam novos tipos de objetos
- São criadas por meio da instrução **class**
- São usadas por meio de chamadas
- Sempre residem dentro de um módulo

Desenvolvimento de classes

- Primeiro Exemplo:

```
Class FirstClass:     #Define um objeto classe (não uma instância)
        nome = None   # atributo de FirstClass com valor inicial nulo
        def setData(self, value): # define métodos  (funções) de classe
                self.data = value   #self é a instância
        def display(self):
                print self.data
```

- FirstClass é uma classe que tem dois métodos (funções).
  - Normalmente, as funções dentro de uma classe são chamadas de métodos; elas são instruções *def* normais, mas o primeiro argumento recebe automaticamente um objeto instância implícito, quando chamadas.

DESENVOLVIMENTO DE CLASSES

```
>> x = FirstClass()     # self é x, é uma palavra reservada que diz que
                        # estamos referenciando a própria instância de
                        # nossa classe (no caso, o objeto x)
>> y = FirstClass()     # self é y
```

- Chamando-se essa classe dessa maneira são gerados dois objetos instância; x e y.

```
>> x.setData("King Arthur")
>> y.setData(3.14159)

>>x.display()     # self.data é diferente em cada instância
King Arthur
>>y.display()
3.14159
```

CONSTRUTOR DA CLASSE

- O construtor de uma classe é um método responsável por inicializar uma nova instância dessa classe. Se ele não for definido o Python atribui a essa classe um construtor vazio (que apenas cria o objeto).

- Exemplo:

```
class Pessoa:
    def __init__(self, nome, idade, genero):  # O construtor recebe 3
                                               # parâmetros
        self.nome = nome    # o atributo nome da instancia que está
                            # sendo criada recebe o valor do
                            # parâmetro nome, e assim por diante...
        self.idade = idade
        self.genero = genero
```

- A classe pessoa tem três atributos (variáveis) que representam o estado de uma pessoa

CONSTRUTOR DA CLASSE

- Para a classe pessoa, a criação de uma nova instância deve ser da seguinte forma:

```
x = Pessoa("Augusta", 23, "F")
y = Pessoa("Fábio", 30, "M")
```

  - x é uma instância (objeto) da classe Pessoa que tem nome Augusta, idade 23 e gênero F. y é uma outra instância (objeto) da classe Pessoa que tem nome Fábio, idade 30 e gênero M.

Herança de classes

- Antes de se herdar uma determinada classe, precisamos importá-la para nosso módulo.

- As superclasses são listadas entre parênteses no cabeçalho de uma instrução **class**.

  - Nesse caso, a classe C1 tem como superclasses (classes pai) as classes C2 e C3.

    >>class C1(C2, C3): .....

  - Todos os objetos definidos nas classes C2 e C3 estarão agora disponíveis na classe C1 através do mecanismo de herança, ou seja, C1 também é um C2 e C3.

HERANÇA DE CLASSES

- Na busca por um atributo chamado a partir da instância de um determinado objeto, o interpretador irá procurar esse atributo de baixo para cima (a partir das subclasses) e da esquerda para a direita.

  - Exemplo:

    ```
    >>x = C1()
    >>x.display()
    ```

  - Nesse caso o interpretador irá buscar por esse método na instância x, depois na classe C1, C2 e C3, respectivamente.

HERANÇA DE CLASSES

- As classes herdam atributos e métodos de suas superclasses

```
import mod1    #mod1 é o módulo (arquivo .py) que contém a classe FirstClass
class SecondClass(mod1.FirstClass):


    def __init__(self, nova_data):
        self.data = nova_data
        self.display()
```

- A classe SecondClass herdou o atributo data e o método display da classe FirstClass

Herança de classes

- Outro exemplo:

```
import mod1    #mod1 é o módulo (arquivo .py) que contém a classe FirstClass
class SecondClass(mod1.FirstClass):  #herda setData
    def display(self):                  # altera o método display
                                        # definido em FirstClass
        print('valor atual = "%s" ' % self.data
```

- SecondClass define o método display para imprimir com um formato diferente. Mas como SecondClass define um atributo de mesmo nome (método display), ela efetivamente anula e substitui o método display em FirstClass

CHAMANDO CONSTRUTORES DA SUPERCLASSE

```
class SuperClasse:
    def  __init__(self, x):
            .... código do construtor da superclasse....

class SubClasse(Super):
    def __init__(self, x, y):
            Super.__init__(self, x)
            .... código do construtor da subclasse....
```

- Nesse caso, quando uma nova instância da classe SubClasse for criada, será invocado o construtor da classe SubClasse que por sua vez chamará o construtor de sua superclasse (SuperClasse)

EXERCÍCIOS

1. Descreva o que os objetos têm em comum:
   a) bicicleta, carro, caminhão, avião, planador, motocicleta, cavalo
   b) prego, parafuso, pino, “percevejo”
   c) tenda, caverna, barraco, celeiro, casa

2. Identifique classes nos seguintes sistemas:
   Obs.: Instancie alguns objetos.
   a) Esta sala de aula;
   b) Um sistema de transporte urbano;
   c) Um ecossistema;
   d) Um sistema de estacionamento de veículos
   e) Um sistema aéreo

3. Classes possuem propriedade (atributos). Identifique propriedades pertencentes à classe PESSOA nos seguintes sistemas:
   a) Um sistema de controle de notas e freqüências ;
   b) Um sistema de registro civil;
   c) Um sistema de correio.