Faculdade IMPACTA TECNOLOGIA

Linguagem de Programação 2

Programação Orientada a Objetos

Faculdade IMPACTA TECNOLOGIA

Objetivos

- Entender a evolução dos paradigmas e linguagens de programações
- Explicar os conceitos de objetos, referência e valor
- Compreender as características e comportamentos de conjunto de objetos e suas analogias
- Aprender a utilização de objetos e seus membros em Python
- Entender o comportamento dos objetos na memória

Faculdade IMPACTA TECNOLOGIA

Evolução das linguagens

- As linguagens de programação são anteriores ao advento do primeiro computador moderno.
- A chamada Linguagem de Máquina foi a **primeira** linguagem de programação que surgiu.
- Essa linguagem introduzia uma sequência de **zeros** e **uns** que permitia escrever programas de computador.
- Exemplo:
  - 00111001 00011101 001101010

Evolução das linguagens

- Com essas sequências de zero e uns, é comum imaginar a **quantidade de erros** que eram cometidos e a **dificuldade** de se programar em uma linguagem deste tipo.
- Logo, criou-se a **Linguagem de Montagem**, o **ASSEMBLY**, que contêm comandos um pouco mais inteligentes para compreensão humana, exemplo:
  - ADD AX, BX (Comando para somar o registrador AX no registrador BX)
  - INC (Comando para incrementar uma variável)

Faculdade IMPACTA TECNOLOGIA

Evolução das linguagens

- Com o ASSEMBLY houve melhora na compreensão e criação de programas, mas ainda era uma linguagem próxima do funcionamento da máquina e longe da linguagem natural humana.
- Então, começaram a criar linguagens de alto nível.
- Em particular, os cientistas preocupados em construir uma tradução de fórmulas matemáticas para a linguagem de montagem, criaram a linguagem **FORTRAN** (FORmula Translator).

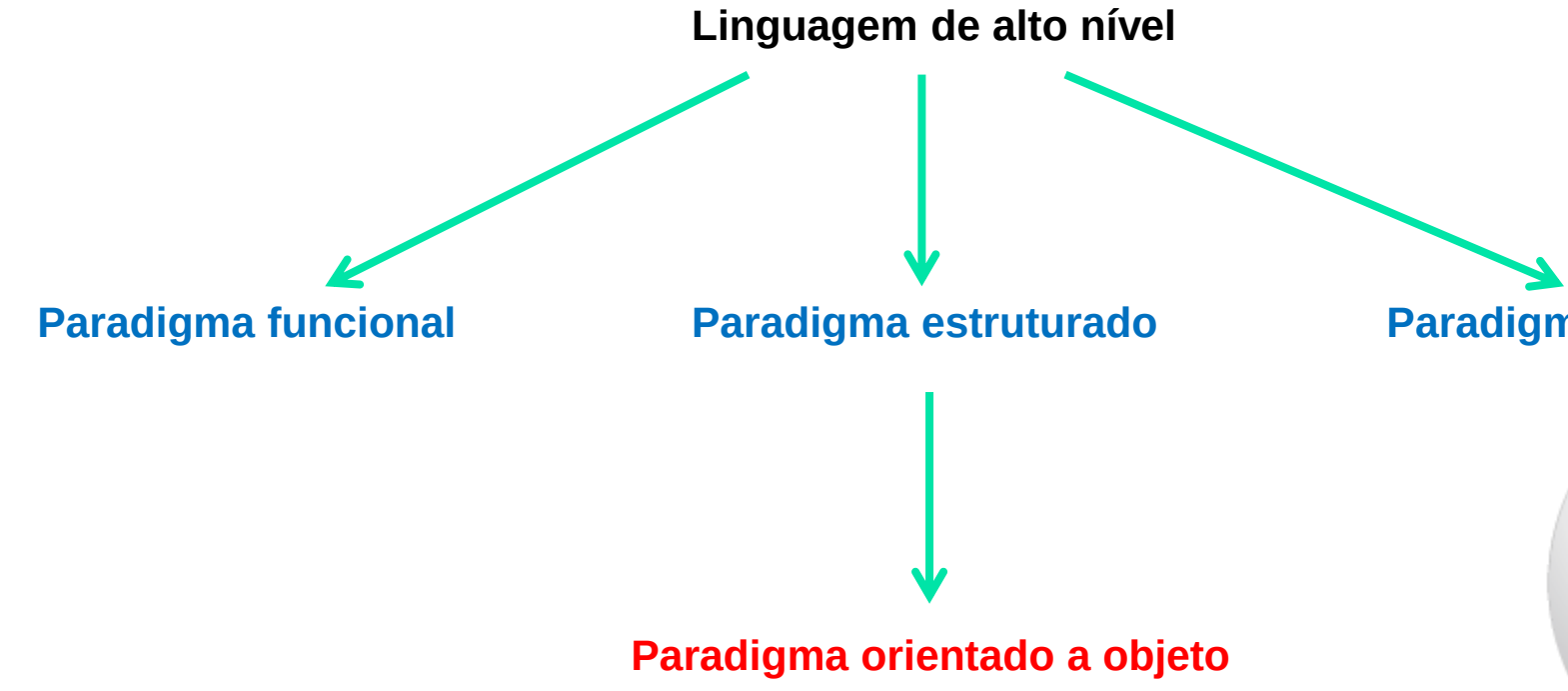
Faculdade IMPACTA TECNOLOGIA

Evolução das linguagens

- O FORTRAN foi a primeira linguagem moderna de alto nível criada na década de 50.
- Na mesma década de 50, outras linguagens com um modelo conceitual um pouco mais sofisticadas foram desenvolvidas, a exemplo, da LISP (LISt Processor), da COBOL (Common Business Oriented Language) e da ALGOL (ALGOrithmic Language).
- Essas linguagens marcaram o início do desenvolvimento das linguagens de alto nível.

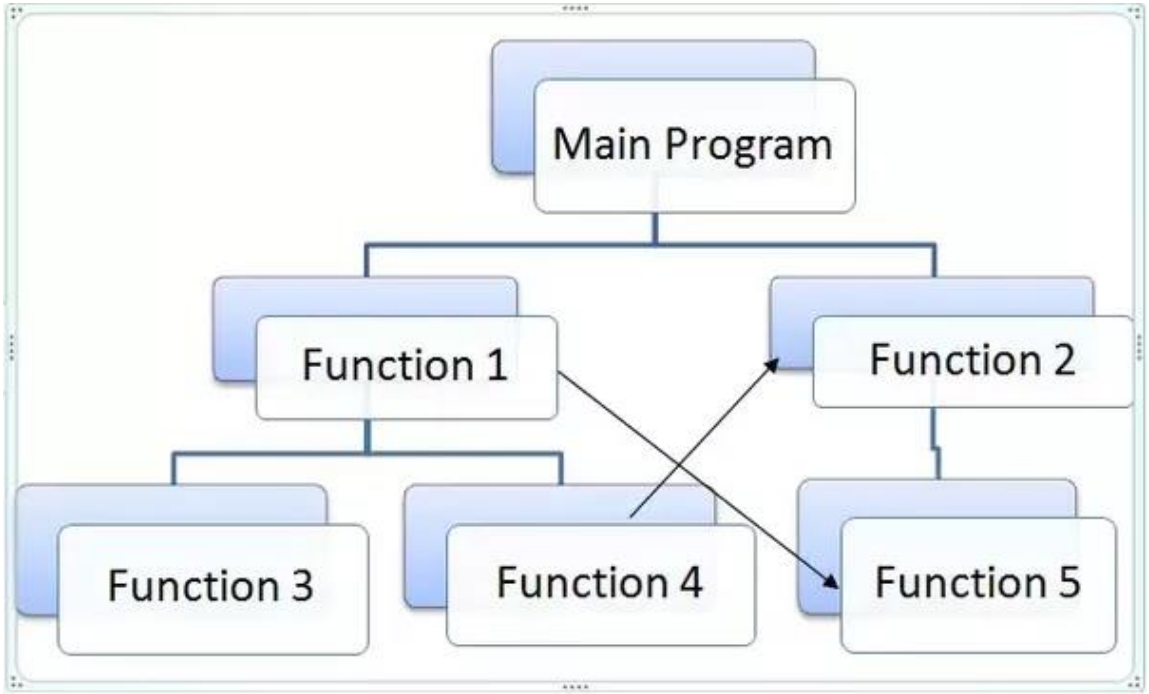
Paradigmas de Programação

- Das linguagens de alto nível, surgiram os **Paradigmas de Linguagens de Programação**, que fornecem e determinam a visão que o programador possui sobre a estruturação e execução do programa



Paradigmas de Programação

- No **paradigma funcional ou procedural**, as funções são o principal bloco de implementação dos programas

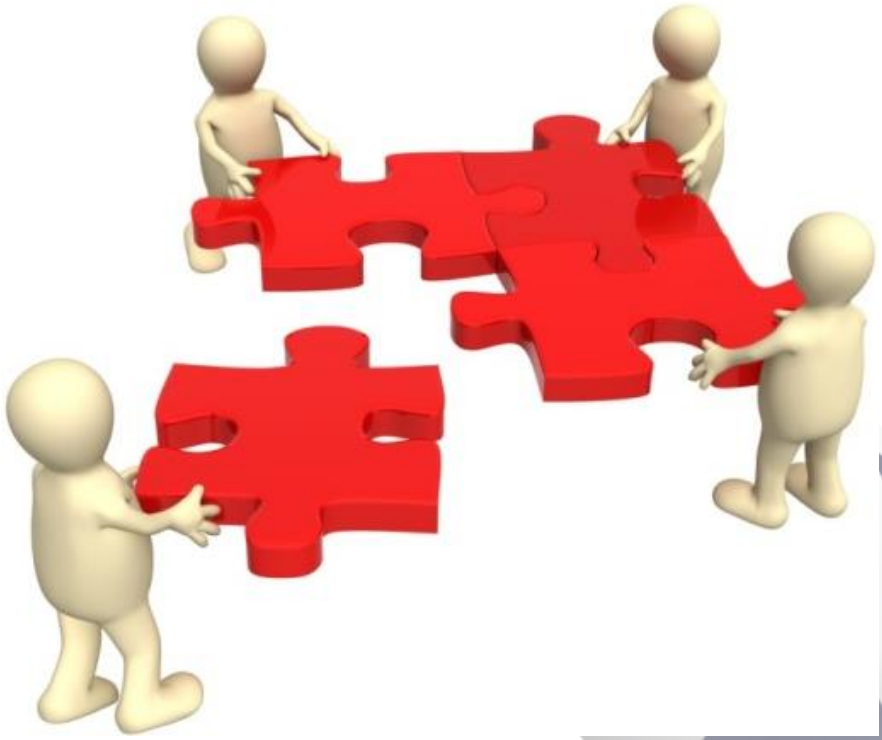


https://www.quora.com/Whats-the-difference-between-Object-Oriented-Programming-and-Procedural-Programming

Faculdade IMPACTA TECNOLOGIA

Paradigmas de Programação

- No **paradigma orientado a objetos**, os programadores abstraem um programa como uma coleção de objetos que interagem entre si.
  - Objetos são o bloco fundamental
  - Funções, chamadas de métodos são componentes



Faculdade IMPACTA TECNOLOGIA

Introdução: Programação Orientado a Objetos

- Mundo real: cercado de Objetos
  - Aluno, professor, sala de aula, livro, carro, animal, televisão...
- Um objeto é algo que tem um grupo de características e um grupo de comportamentos
  - Comportamentos estão relacionados às características
- Em programação orientada a objetos, variáveis de instância definem as características e métodos definem o comportamento

Analogias: Programação Orientado a Objetos

Objetos

Características (atributos)

Raça
Peso
Cor dos olhos

Marca
Cor
Velocidade

Faculdade IMPACTA TECNOLOGIA

Analogias: Programação Orientado a Objetos

Objetos

Comportamento (métodos)

Comer
Dormir

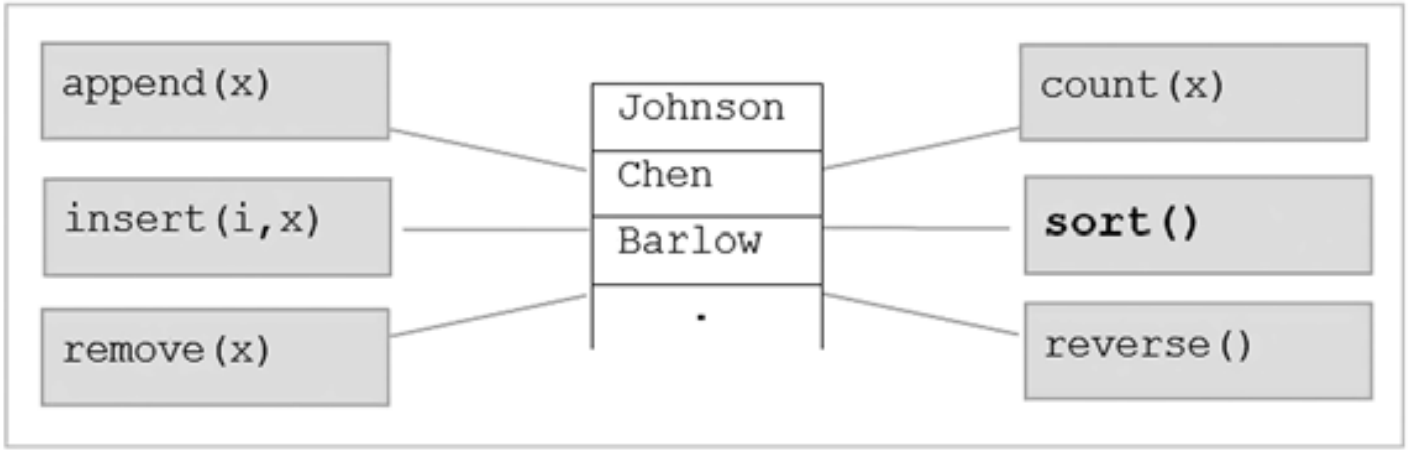
Acelerar
Frear
Pintar

Faculdade IMPACTA TECNOLOGIA

Introdução: Programação Orientado a Objetos

- Em Python, todos os valores em Python são representados como objetos
  - Desde os valores numéricos até listas e objetos complexos
- Vamos lembrar do uso de listas. Como fazemos para:
  - Ordenar uma lista?
  - Adicionar em uma lista?
  - Remover de uma lista?
  - Inverter a lista?

Introdução: Programação Orientado a Objetos



DIERBACH, C. Introduction to Computer Science Using Python: A Computational Problem-Solving

- Essas rotinas fazem parte do objeto que contém a lista
- Se houver uma variável lista_nomes no programa, lista_nome é um objeto do tipo list

Introdução: Programação Orientado a Objetos

- Todos os objetos do tipo lista criados em um programa Python contém os mesmos métodos
  - Qualquer lista criada pode utilizar os mesmos métodos
- Para ordenar a lista basta chamar o método de ordenação (sort) presente no objeto (lista_nomes), separando por . (ponto)
  - lista_nomes.sort()
- O . (ponto) é um operador utilizado sempre que for necessário acessar um membro de um objeto (variáveis ou métodos)

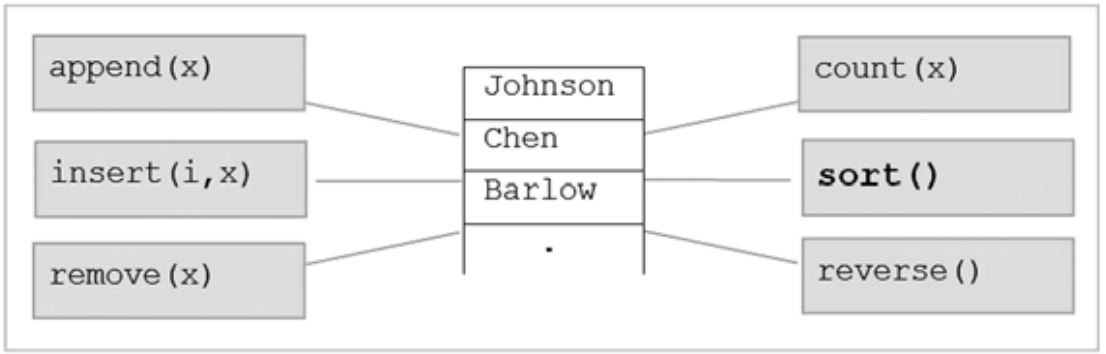
Introdução: Programação Orientado a Objetos

- O método sort vai agir nos dados do objeto do qual ele faz parte
  - No caso, lista_nomes
- Se criarmos uma outra lista, mas agora de números (lista_numeros), ela também terá o método sort
  - lista_nomes e lista_numeros são objetos do tipo list
- Entretanto, os dados nos quais a operação de ordenação será aplicado serão diferentes
  - Cada objeto tem seus dados, e os métodos agem nestes dados

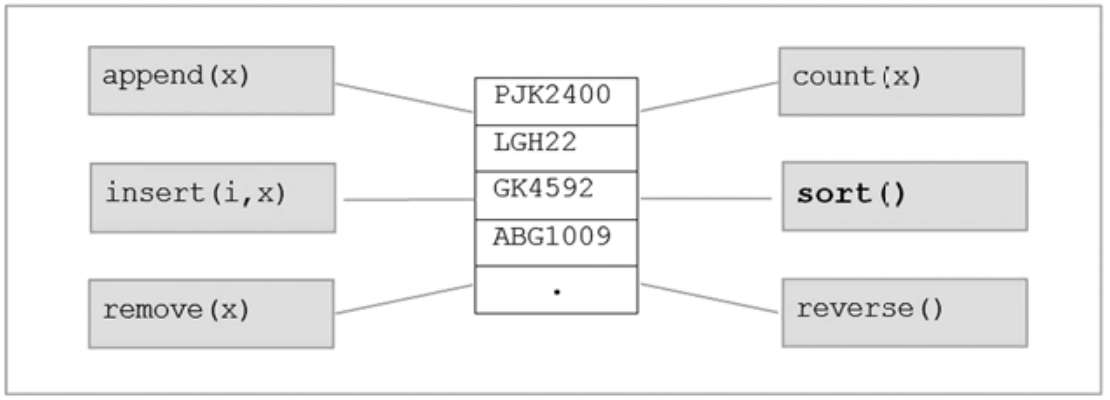
Introdução: Programação Orientado a Objetos

- Portanto, dois objetos do mesmo tipo tem os mesmos métodos. Eles diferem nos valores que guardam (variáveis ou atributos)

lista_nomes



lista_numeros



DIERBACH, C. Introduction to Computer Science Using Python: A Computational Problem-Solving

Para lembrar

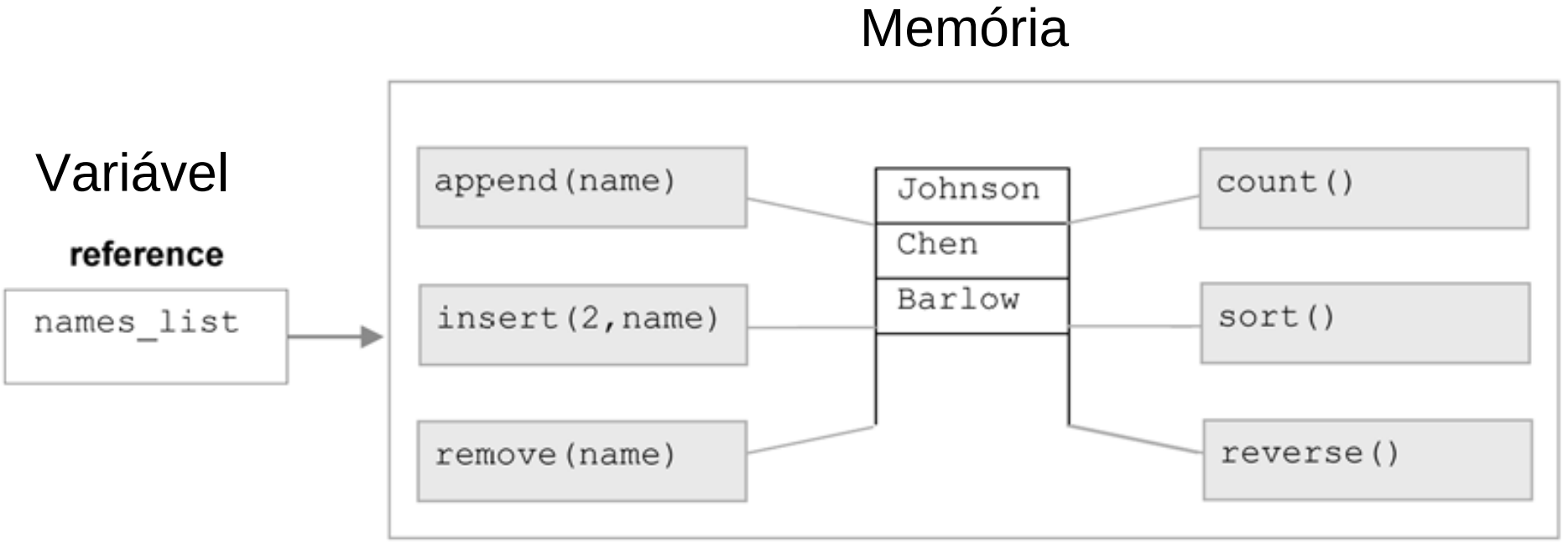
Um **objeto** contém um grupo de atributos ou características, armazenados em **variáveis de instâncias** e um grupo de funções, chamados de **métodos**, que provê o comportamento

Faculdade IMPACTA TECNOLOGIA

Referências a Objetos

- É importante entender como Python representa os objetos e o efeito que isso tem na atribuição
- Em Python os objetos são representados como uma **referência** para um objeto na memória
  - Ou seja, a variável não contém o objeto propriamente dito, mas uma referência para o local da memória onde o objeto está

Referências a Objetos



DIERBACH, C. Introduction to Computer Science Using Python: A Computational Problem-Solving

- A variável, portanto, armazena uma referência para um objeto

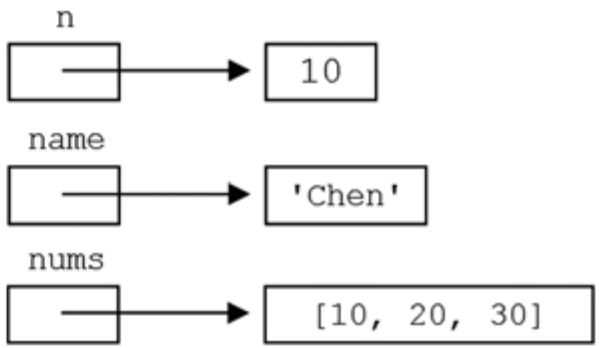
Referências a Objetos

- Qualquer acesso a um objeto é feito pela variável de referência

```
numeric          n = 10

string         name = 'Chen'

list         nums = [10, 20, 30]
```



DIERBACH, C. Introduction to Computer Science Using Python: A Computational Problem-Solving

Faculdade IMPACTA TECNOLOGIA

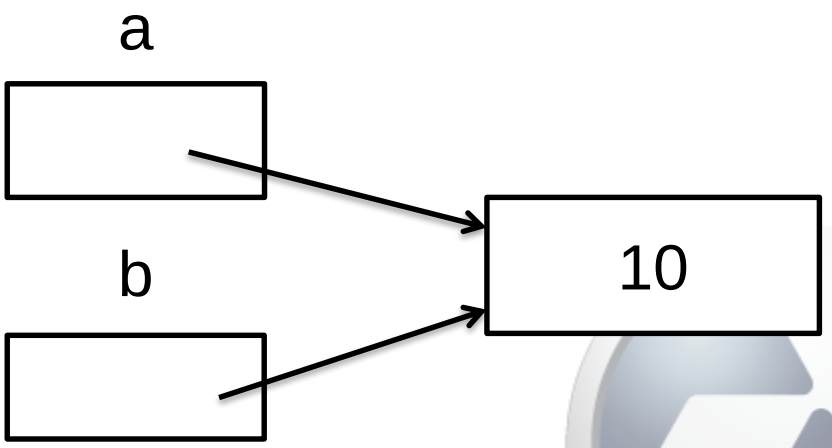
Referências a Objetos

- Em Python existem dois tipos de objetos
  - Imutáveis
    - int, float, bool, tupla, string
  - Mutáveis
    - listas, dicionários, novos objetos
- É importante entender como a referência se comporta para cada um dos tipos

Referências a Objetos - Imutáveis

- Quando um mesmo valor de um tipo imutável é atribuído a duas variáveis diferentes, as duas variáveis referenciam o mesmo local na memória
  - As duas variáveis referenciam a mesma instância
  - Isso é feito para economizar memória
- Por exemplo:

```
a = 10
b = 10
```

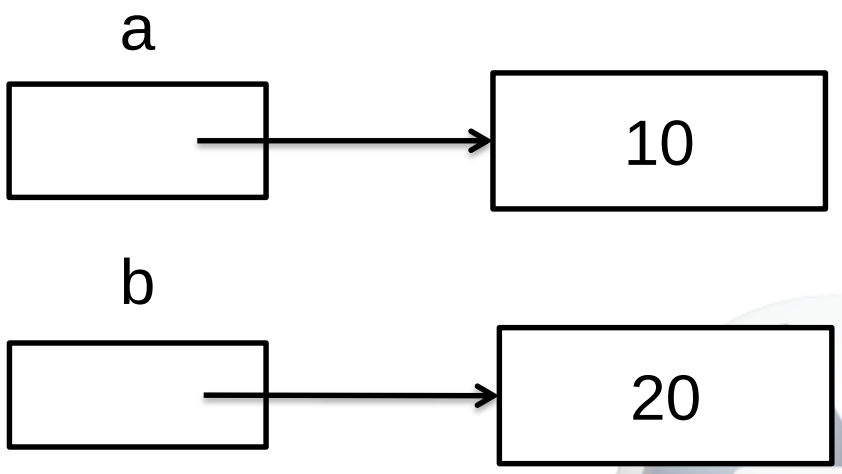


  - Teste utilizando a função id(variavel) do python

Referências a Objetos - Imutáveis

- Se a variável b receber um novo valor (nova instância), ela passa a referenciar um novo espaço na memória, que armazena esta nova instância

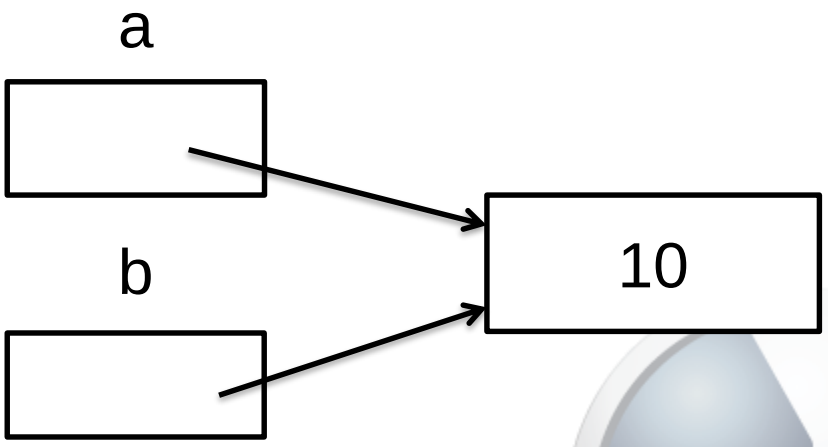
```
b = 20
```



Referências a Objetos - Imutáveis

- Se a variável **b** receber a referência de **a** (ou seja b = a), a variável **b** volta a referenciar a posição de memória onde está o valor 10

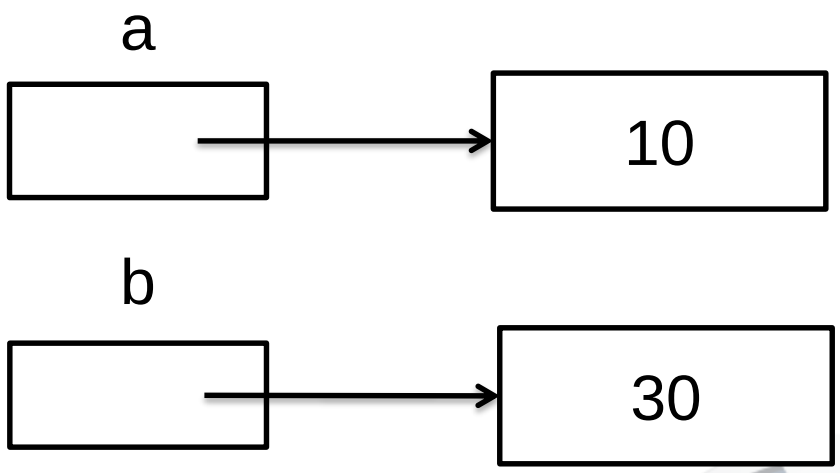
b = a



Referências a Objetos - Imutáveis

- Se a variável **b** receber um outro valor (nova instância), **b** novamente passa a referenciar um novo espaço na memória, que a nova instância

```
b = 30
```



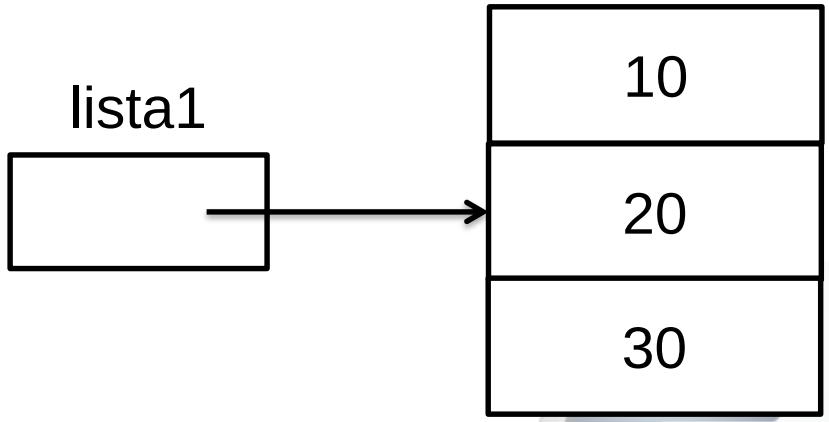
Referências a Objetos - Imutáveis

- Ou seja, para tipo imutáveis, se duas variáveis referenciam a mesma posição de memória (mesma instância) e a primeira receber uma nova referência (nova atribuição), o valor referenciado pela segunda variável permanece o mesmo, alterando apenas a instância referenciada pela primeira variável

Referências a Objetos - Mutáveis

- O comportamento para tipos mutáveis é diferente
- Vamos utilizar como exemplo uma lista (tipo list) de números, atribuindo a uma variável lista1

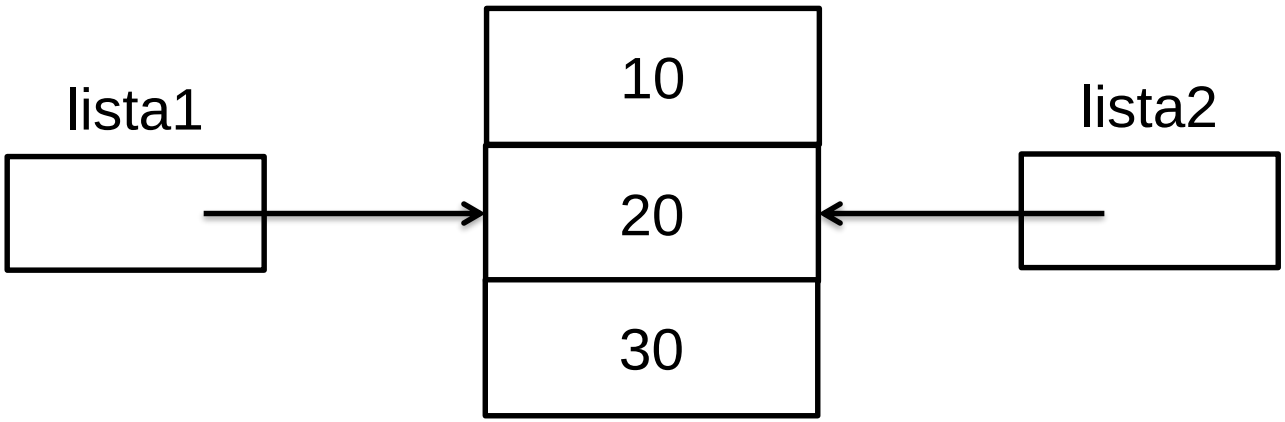
```
lista1 = [10, 20, 30]
```



Referências a Objetos - Mutáveis

- Se atribuirmos a referência de lista1 para uma variável lista2 (lista2 = lista1), ambas as variáveis vão **referenciar a mesma instância,** ou seja, o mesmo espaço na memória

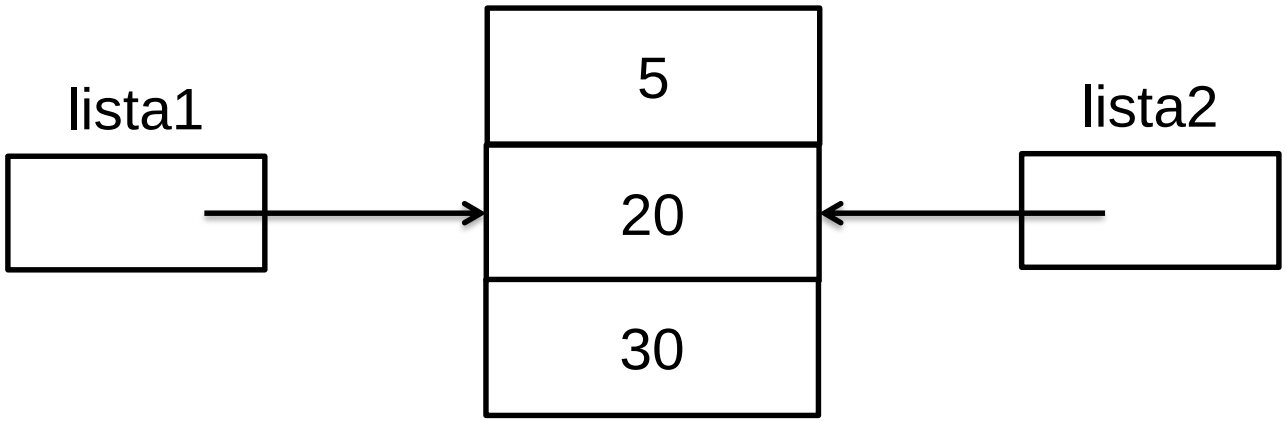
```
lista2 = lista1
```



Referências a Objetos - Mutáveis

- Por se tratar de um tipo mutável, e ambas as variáveis referenciarem a mesma instância qualquer alteração em lista1 refletirá na lista2

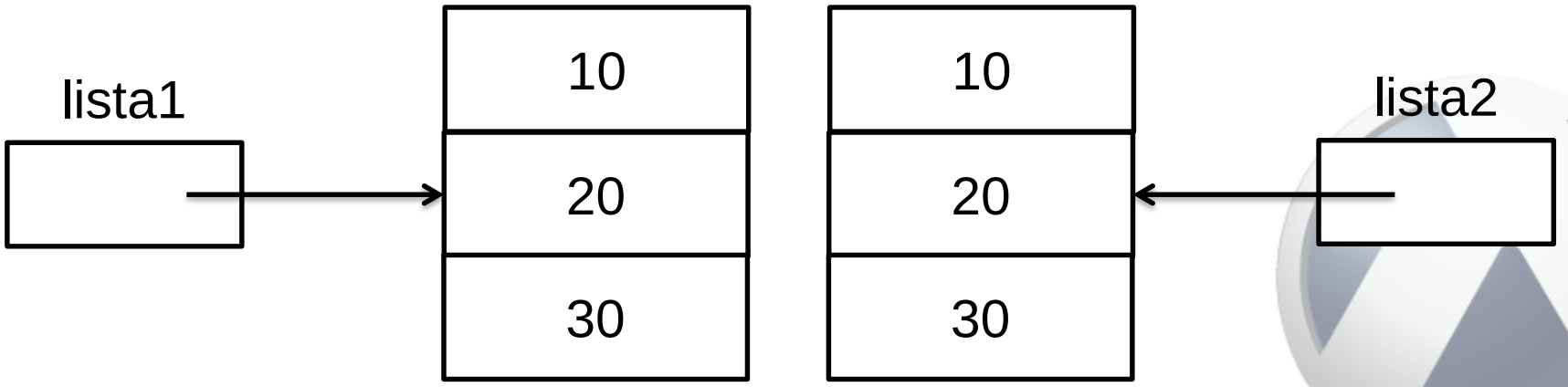
```
lista1[0] = 5
print(lista2[0])
# a alteração do 5 na primeira posição da lista1
# refletirá na lista2
```



Referências a Objetos - Mutáveis

- Este é o comportamento para qualquer tipo mutável, inclusive aqueles objetos que criamos
- Para que lista2 referencie uma nova instância de lista1 com o mesmo conteúdo, é necessário criar a nova instância. Com listas utiliza-se o construtor list()

```
lista2 = list(lista1)
```



Faculdade IMPACTA TECNOLOGIA

Classes

Classe

Objetos (Instâncias)

Classes

- **Classes** são como formas
  - Definem os tipos de objetos
  - Definem quais os dados e o código dos objetos
- **Objeto** = dados + códigos
  - Encapsulamento (segurança)
  - Dados => atributos (características/substantivo)
  - Códigos => métodos (comportamento/verbo)

Classes

- Os objetos são também chamados de *instâncias* das classes
  - Cada objeto tem suas próprias características
- Com a fôrma pronta (classe), é possível criar quantos objetos forem necessários utilizando a mesma fôrma
- Instâncias da mesma classe tem características comuns

Analogias: Programação Orientado a Objetos

- **Características comuns** entre objetos:

Possui raça?
Possui peso?
Possui cor dos olhos?

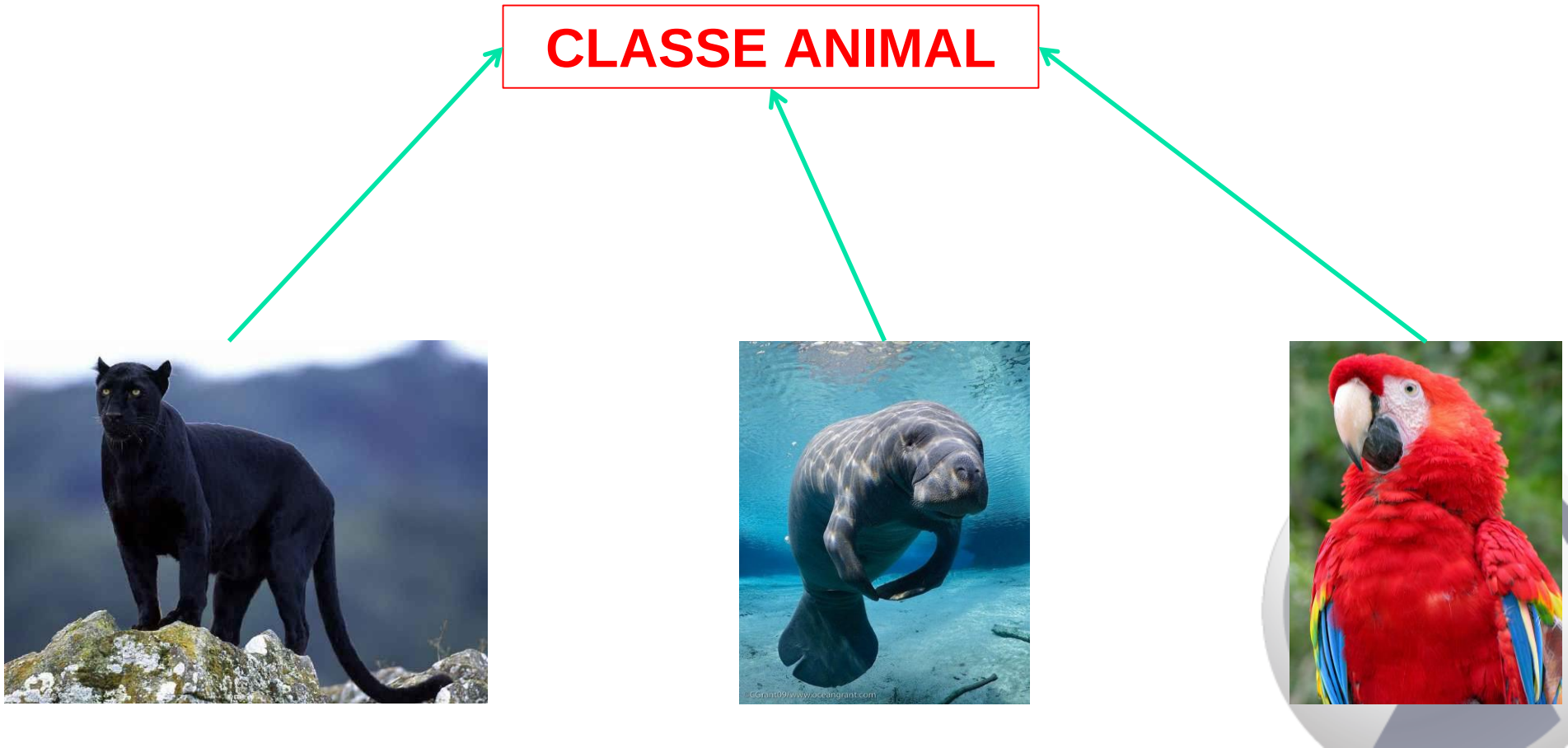
Analogias: Programação Orientado a Objetos

- **Comportamentos comuns** entre objetos:

Comem?
Dormem?

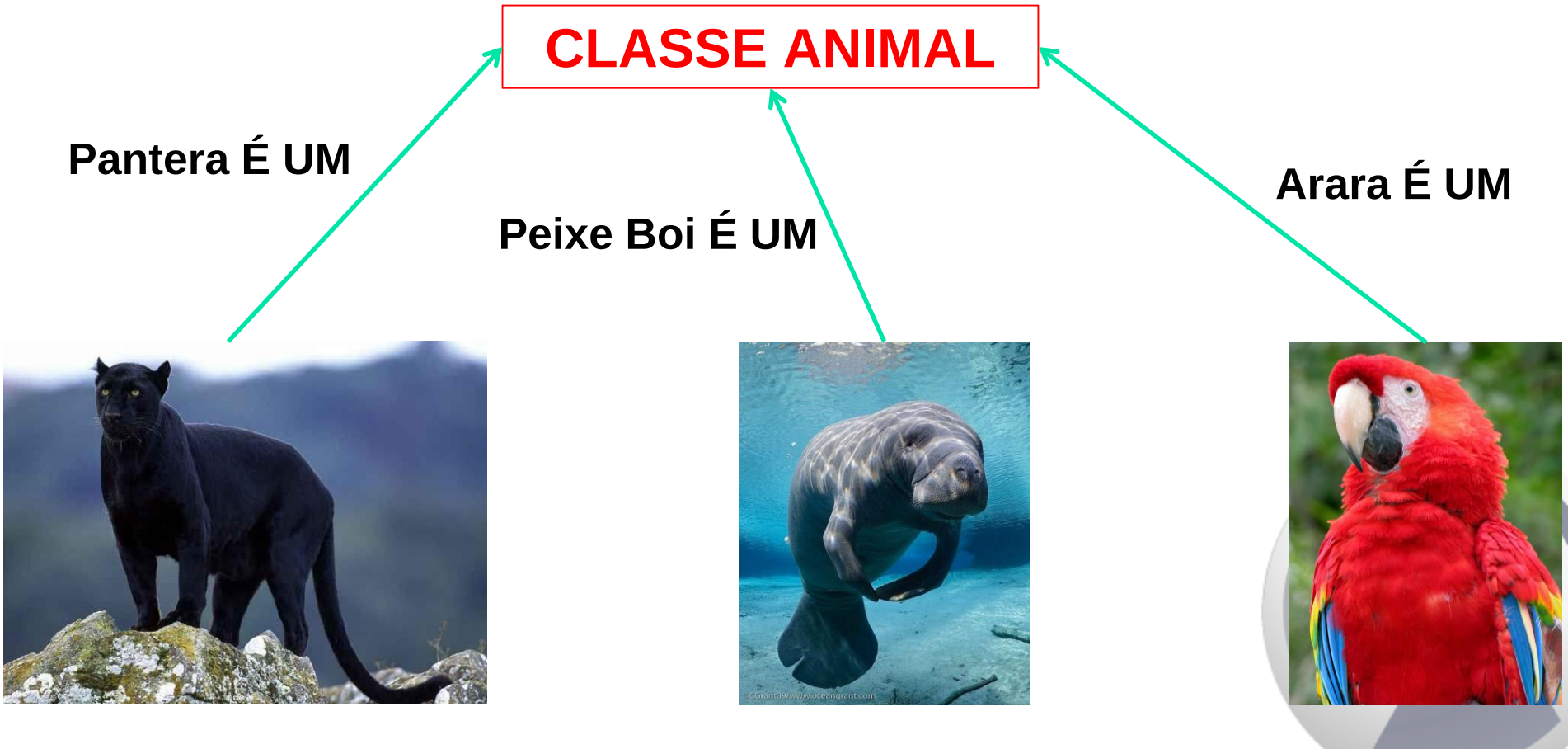
Analogias: Programação Orientado a Objetos

- Assim, **características e comportamentos comuns** entre objetos **definem** uma **classe genérica**.



Analogias: Programação Orientado a Objetos

- Conceitualmente, esta definição é denominada **relacionamento** do tipo **É UM**.



Analogias: Programação Orientado a Objetos

- Desta forma, podemos definir uma classe ANIMAL e, a partir dela, definir **objetos** que lhe são **específicos**.

CLASSE ANIMAL

Pantera

Peixe Boi

Arara

Analogias: Programação Orientado a Objetos

- Porém, cada objeto pode ter **característica e/ou comportamentos específicos** que não refletem a classe genérica.

Atributos e Métodos específicos

Arara têm pêlos?
Pantera têm penas?
Arara nada?
Peixe Boi voa?
Peixe Boi salta em árvores?

cor dos pêlos

Saltar em árvores

cor da pele

Nadar no fundo do rio

cor das penas

Voar pelo céu

Exemplo: Programação Orientado a Objetos

- Classe: Animal
- Instâncias: meu_animal, animal_da_fit
- Atributos:
  - nome, tipo e cor
- Métodos:
  - dormir()
  - comer()

Faculdade IMPACTA TECNOLOGIA

Prática: Programação Orientado a Objetos

```
#Criação da classe Animal sem bloco de código (pass)
class Animal:
      pass

#Criação de 2 instâncias de Animal
meu_animal = Animal()
animal_da_fit = Animal()

#mostra a localização da memória para as instâncias
print(meu_animal)
print(animal_da_fit)
```

Obs: Apesar de criar uma animal com pass, é possível definir atributos e métodos através de suas instâncias.

Atributos: Programação Orientado a Objetos

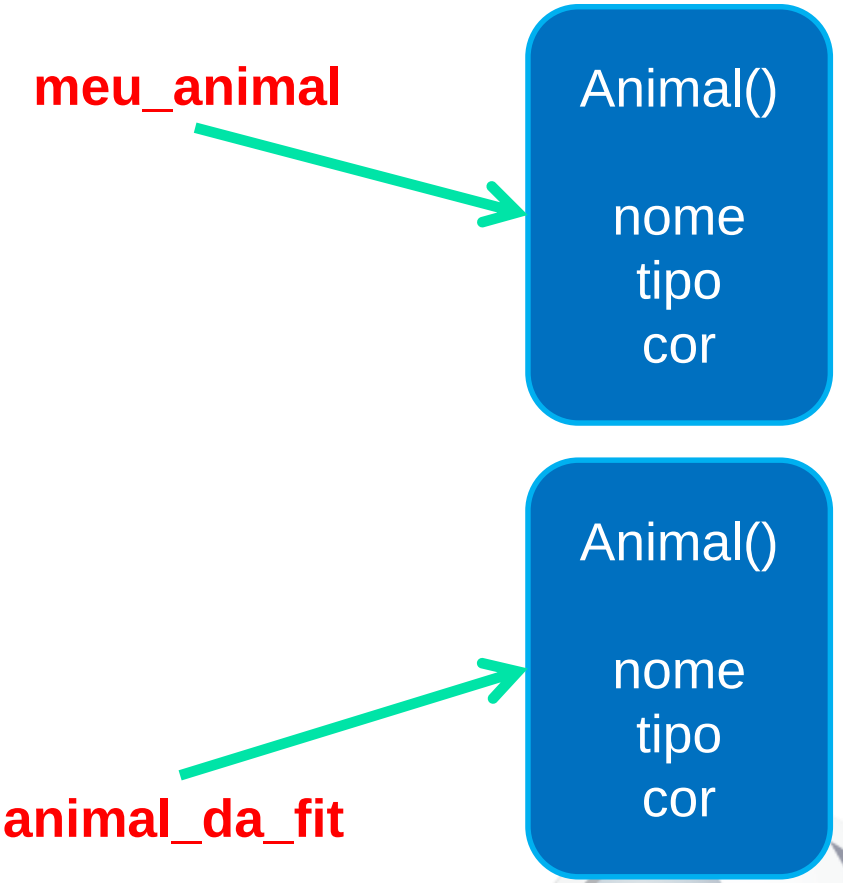
- A instância de **animal** por meio do operador ponto "." permite realizar chamadas ou definir atributos.

```
#Definição de 3 atributos (características) da instância
meu_animal.nome = "Rex"
meu_animal.tipo = "Urso"
meu_animal.cor = "Branco"
```

- Atributos definidos:
  - Nome = Rex
  - Tipo = Urso
  - Cor = Branco

Objetos: Programação Orientado a Objetos

```
#Definição de 3 atributos
meu_animal = Animal()
meu_animal.nome = “Rex”
meu_animal.tipo = “Urso”
meu_animal.cor = “Branco”
```

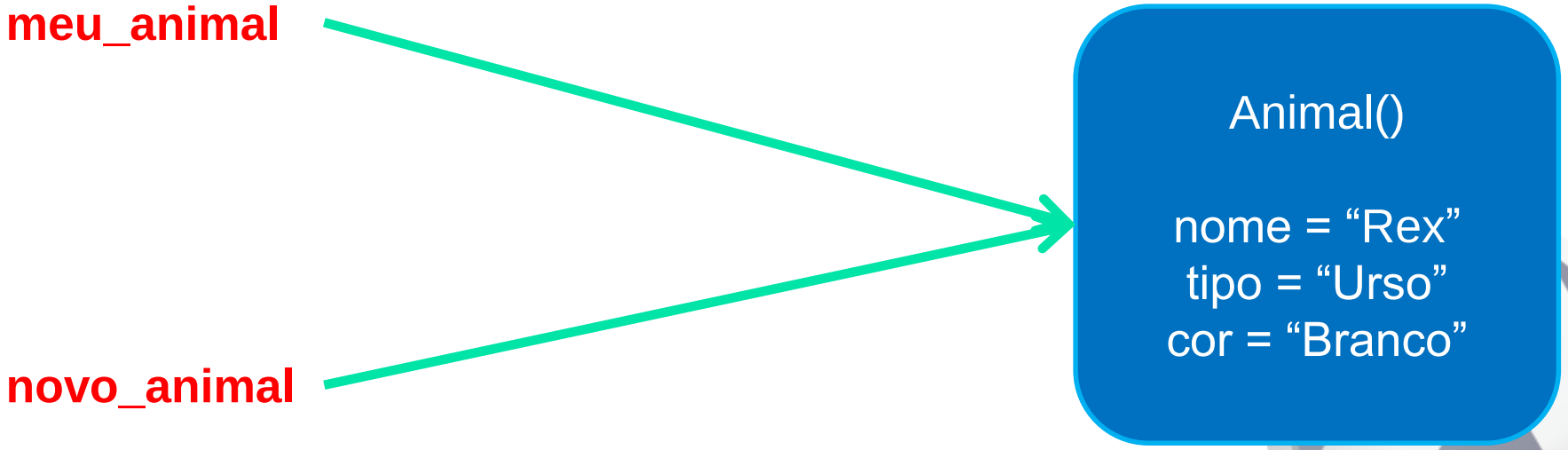



```
#Definição de 3 atributos
animal_da_fit = Animal()
animal_da_fit.nome = “Policia”
animal_da_fit.tipo = “Cachorro”
animal_da_fit.cor = “Preto”
```

Referências: Programação Orientado a Objetos

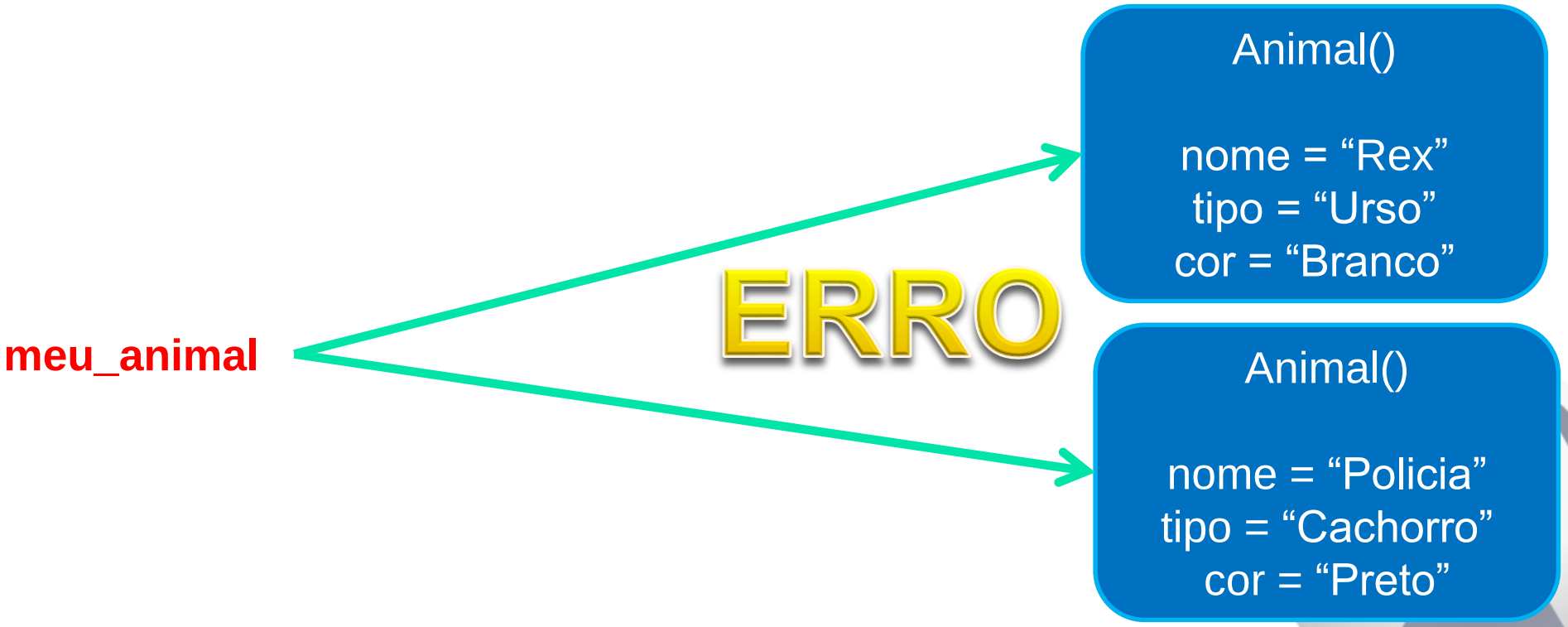
- É possível ter **duas variáveis** referenciando o mesmo **objeto** na memória:

```
#Atribuição da cópia da referência de meu_animal
novo_animal = meu_animal
```



Referências: Programação Orientado a Objetos

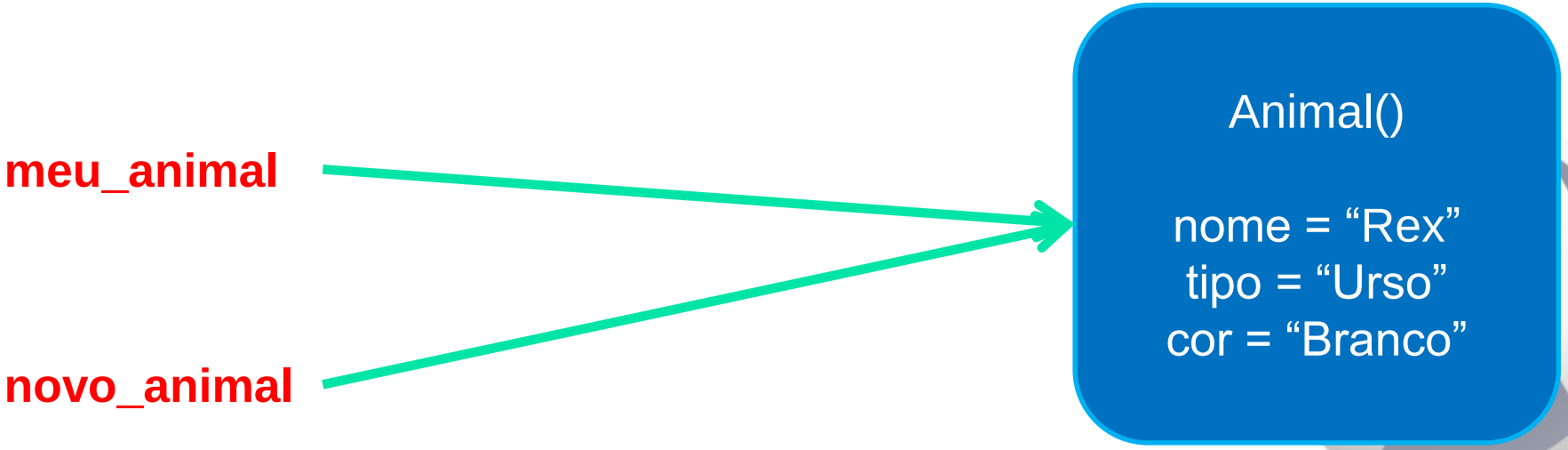
- Mas NUNCA a mesma **variável referenciando dois objetos diferentes**:



Referências: Programação Orientado a Objetos

- Ao **atribuir uma cópia** da referência da variável meu_animal à variável novo_animal, qualquer alteração de atributos do objeto promovido por uma, resultará na mesma percepção de alteração pela outra variável.

```
#Atribuição da cópia da referência de meu_carro
novo_animal = meu_animal
```



Faculdade IMPACTA TECNOLOGIA

Prática: Programação Orientado a Objetos

```
#Atribuição da referência meu_animal
novo_animal = meu_animal

#Criação de 2 instâncias de Animal
novo_animal.nome = “Pegasus”
novo_animal.cor = “Cinza”

#mostrar a localização da memória para as instâncias
print(meu_animal.nome)
print(meu_animal.cor)
```

Obs: Neste momento, o nome e a cor do objeto Animal() alteraram seu valor, tanto para a referência novo_animal quanto para a referência meu_animal. Isto porque ambos referenciam o mesmo objeto.

Exercício 1

- O que o programa abaixo imprime?

```
class Pato:
      pass

pato = Pato()
patinho = Pato()

if pato == patinho:
      print(“Estamos no mesmo endereço!”)
else:
      print(“Estamos em endereços diferentes”)
```

Faculdade IMPACTA TECNOLOGIA

Exercício 2

- A justificativa para a resposta do Exercício **1** é:
  - A) Objetos diferentes são alocados em endereços de memória iguais.
  - B) Variáveis com nomes diferentes não podem apontar para um mesmo objeto.
  - C) Objetos diferentes são alocados em endereços de memórias diferentes.
  - D) Em Python, não se podem comparar duas instâncias de uma mesma classe.

Faculdade IMPACTA TECNOLOGIA

Obrigado!

Prof. MSc. Fernando Sousa