
FIAP
MBA

Programação Funcional e Orientação à Objetos

Dheny R. Fernandes

Agenda

FIAP MBA+

1. **Programação Funcional**
    1. Lambda
    2. Map
    3. Reduce
    4. Filter
    5. Zip
    6. List/Set/Dict comprehension
2. **Orientação à Objetos**
    1. Classe
    2. Intuição
    3. Criando uma classe
    4. Herança

Programação Funcional

Programação funcional é um paradigma que trata a computação como uma avaliação de funções matemáticas.

Tais funções podem ser aplicadas em sequências de dados (geralmente listas, numpy arrays e pandas DataFrames).

Funções:

- Lambda
- Map
- Reduce
- Filter
- Zip
- List comprehension

lambda é uma função anônima composta apenas por expressões.

Sintaxe:

lambda <lista de variáveis**>: <**expressões **>**

```
#exemplos lambda

quadrado = lambda num: num ** 2
multiplicacao = lambda x,y: x*y
```

O mapeamento consiste em aplicar uma função a todos os itens de uma sequência, gerando outra lista contendo os resultados e com o mesmo tamanho da lista inicial.

Sintaxe:

map(função, lista)

Retorna uma lista em forma de objeto. Use o comando *list()* para imprimir corretamente.

Programação Funcional - Map

Exemplos:

```
import math
lista1 = [1, 4, 9, 16, 25]
lista2 = map(math.sqrt, lista1)
print(list(lista2))
```

```
nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
from math import log10
print (list(map(log10, nums)))
print (list(map(lambda x: x / 3, nums)))
```

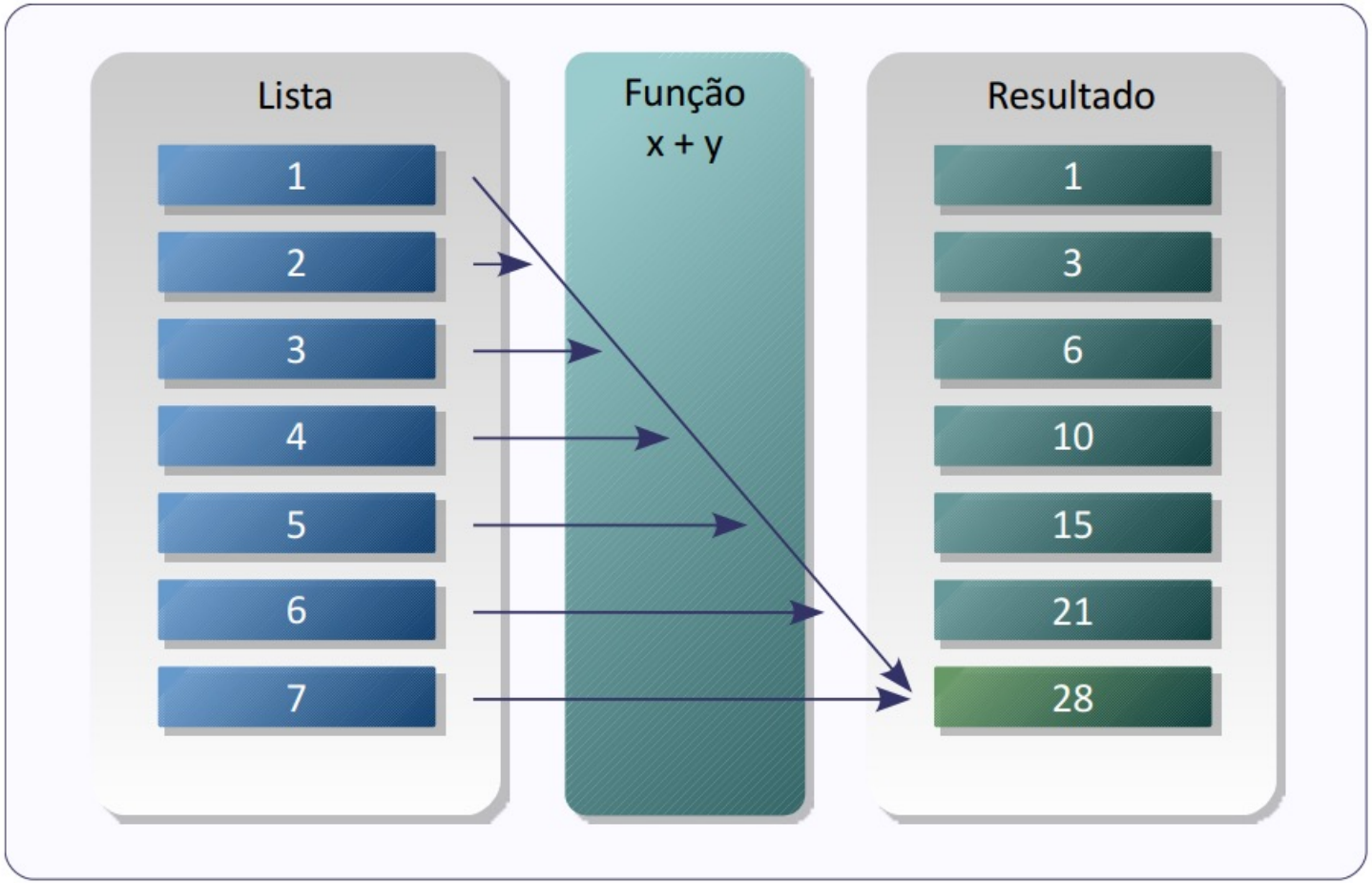
Redução significa aplicar uma função que recebe dois parâmetros, nos dois primeiros elementos de uma sequência, aplicar novamente a função usando como parâmetros o resultado do primeiro par e o terceiro elemento, seguindo assim até o final da sequência. O resultado final da redução é apenas um elemento.

Deixou de ser *built-in* no Python 3. Necessário importar *functools.*

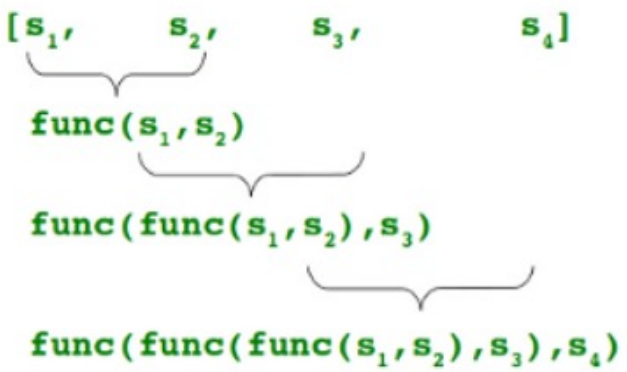
Sintaxe:

- *reduce(função, lista)*

Programação Funcional - Reduce



Programação Funcional - Reduce




```
from functools import reduce
nums = range(100)
print (reduce(lambda x, y: x + y, nums))

f = lambda a,b: a if (a > b) else b
print(reduce(f, [47,11,42,102,13]))
```

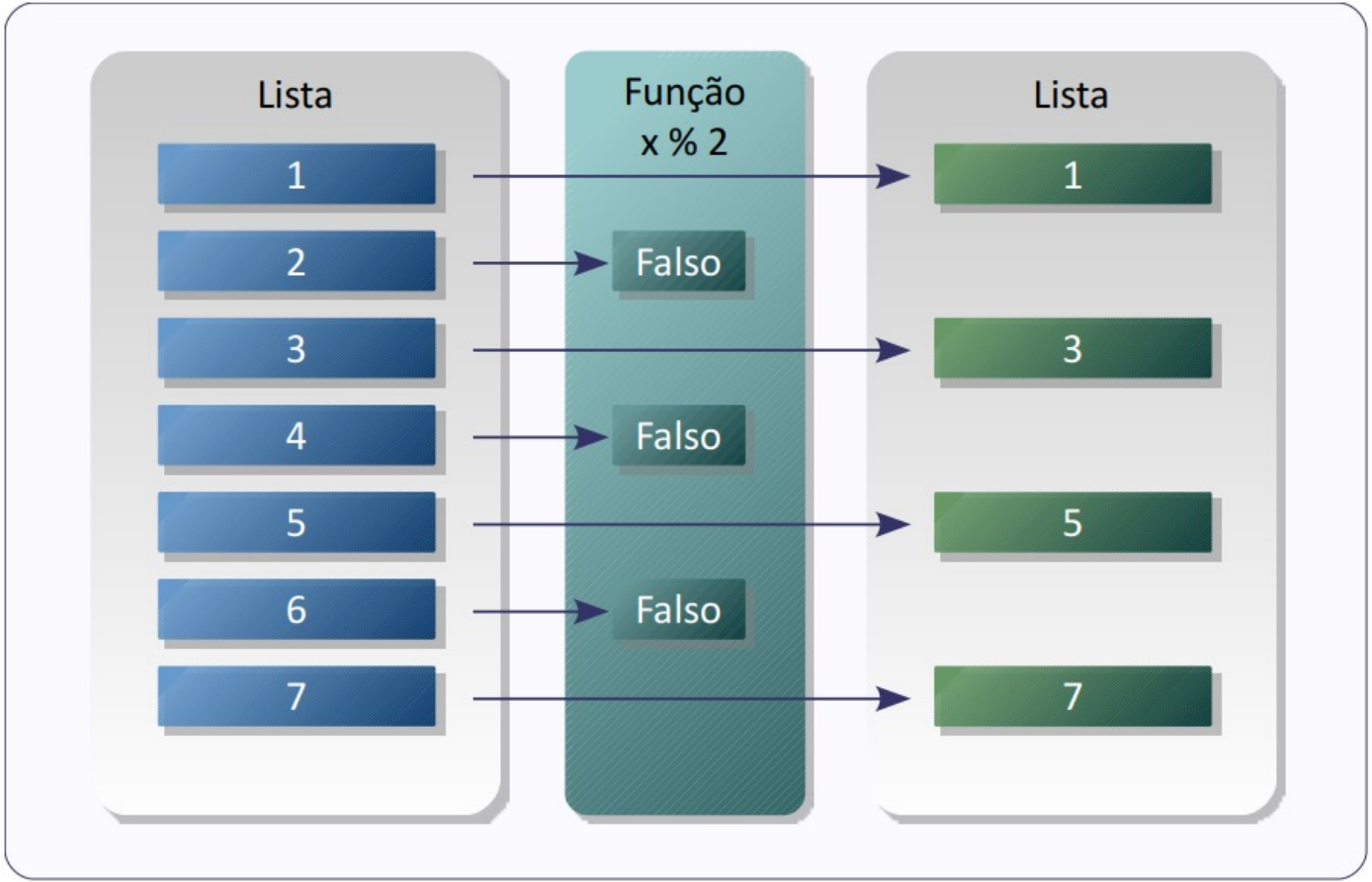
Na filtragem, uma função é aplicada em todos os itens de uma sequência. Se a função retornar um valor que seja avaliado como verdadeiro, o item original fará parte da sequência resultante.

Sintaxe:

filter(função, lista)

Retorna uma lista em forma de objeto. Use o comando *list()* para imprimir corretamente.

Programação Funcional - Filter



Exemplos:

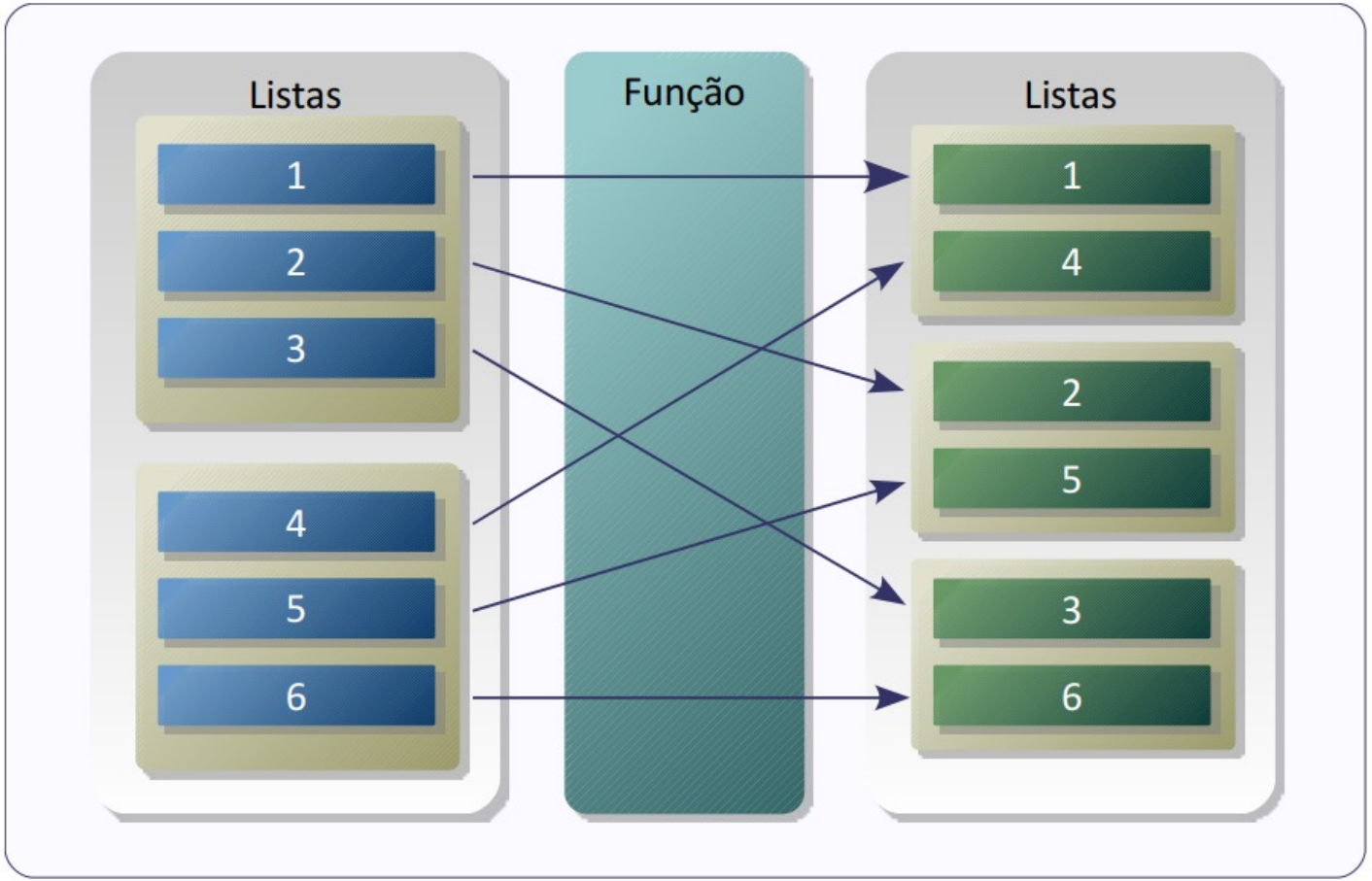
```
x = [1,2,3,4,5,6,7]
print (list(filter(lambda x: x % 2, x)))

number_list = range(-5, 5)
less_than_zero = list(filter(lambda x: x < 0, number_list))
print(less_than_zero)
```

Transposição é construir uma série de sequências a partir de outra série de sequências, onde a primeira nova sequência contém o primeiro elemento de cada sequência original, a segunda nova sequência contém o segundo elemento de cada sequência original, até que alguma das sequências originais acabe.

Sintaxe:

zip(objeto)

Programação Funcional – Zip (Transposição)



Exemplos:

```
x = [1, 2, 3]
y = [4, 5, 6]
print (list(zip(x, y)))

matriz = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
print (list(zip(*matriz)))
```

Fornecem uma maneira concisa de criar listas. Aplicações comuns são criar novas listas em que cada elemento é o resultado de alguma operação aplicada em cada membro de outra sequência, ou criar uma subsequência desses elementos que satisfaça uma condição.

Sintaxe:

Lista = [<expressão> for <referência> in <sequência> if <condição>]

Considere a seguinte notação matemática:

- $S = \{x * 2 | x \in \mathbb{N}, x < 10\}$
- S é um conjunto cujos membros são os dobros de todos os números naturais menores que 10

```
S = [x*2 for x in range(0, 10)]
print(S)
```

Exemplos:

```
nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
print ([ x**2 for x in nums if x % 2 ])
```

```
[[s.capitalize(), s.upper(), len(s)] for s in ['um', 'dois', 'tres']]
```

Podemos ampliar o conceito para Sets também, que são uma extensão natural das List Comprehension.

Considere a lista de strings abaixo. Suponha que queremos um Set que contenha apenas os tamanhos das strings. Podemos fazer isso da seguinte maneira:

```
strings = ['a', 'um', 'sim', 'nao', 'dois', 'exemplo']
```

```
unique_lengths = {len(x) for x in strings}
unique_lengths
```

De maneira similar, podemos criar um mapa de pesquisa dessas strings para suas localizações na lista usando Dict Comprehension:

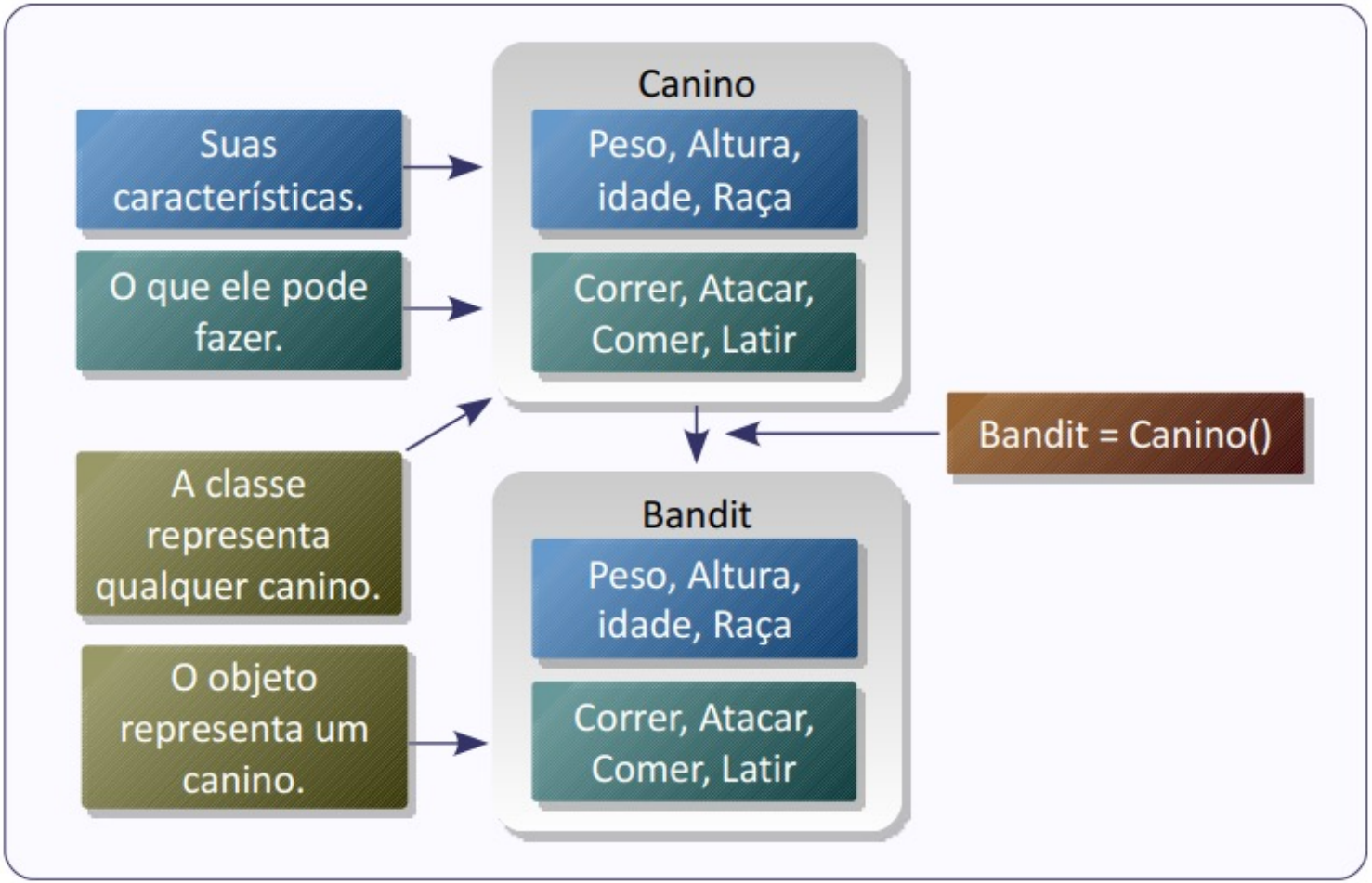
```
loc_mapping = {value:index for index, value in enumerate(strings)}
loc_mapping
```

Resolver o exercício no notebook

Orientação a Objetos

Objetos são abstrações computacionais que representam entidades, com suas qualidades (atributos) e ações (métodos) que estas podem realizar

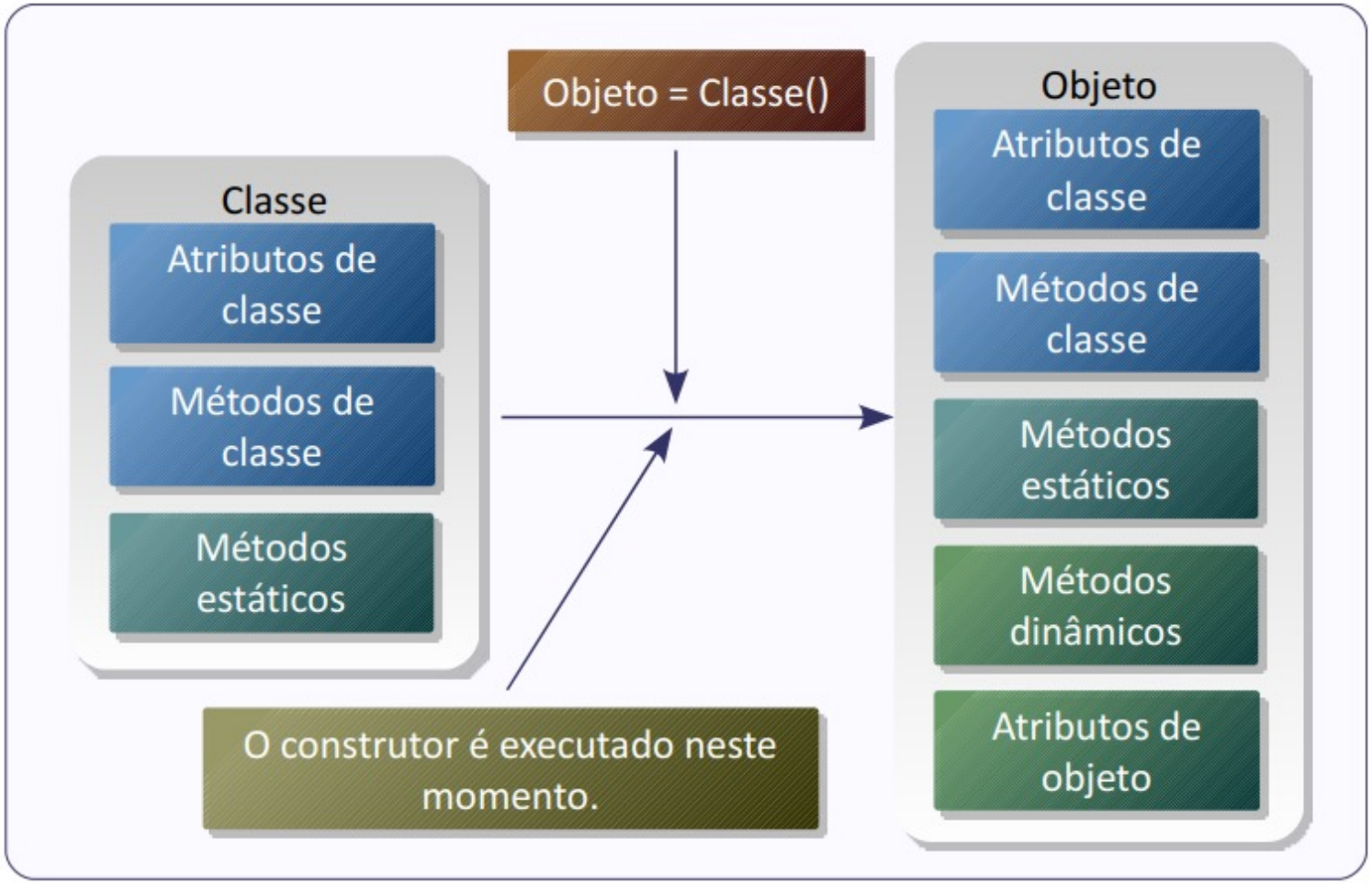
A classe é a estrutura básica do paradigma de orientação a objetos, que representa o tipo do objeto, um modelo a partir do qual os objetos serão criados.


Canino
Suas características.
Peso, Altura, idade, Raça
O que ele pode fazer.
Correr, Atacar, Comer, Latir
Bandit = Canino()
A classe representa qualquer canino.
Bandit
Peso, Altura, idade, Raça
O objeto representa um canino.
Correr, Atacar, Comer, Latir

Os atributos são estruturas de dados que armazenam informações sobre o objeto e os métodos são funções associadas ao objeto, que descrevem como o objeto se comporta.

No Python, novos objetos são criados a partir das classes através de atribuição. O objeto é uma instância da classe, que possui características próprias.

Quando um novo objeto é criado, o construtor da classe é executado.


Objeto = Classe()
Classe
Atributos de classe
Métodos de classe
Métodos estáticos
Objeto
Atributos de classe
Métodos de classe
Métodos estáticos
Métodos dinâmicos
Atributos de objeto
O construtor é executado neste momento.

```python
class SomeClass:
    def create_arr(self):
        self.arr = []

    def insert_to_arr(self, value):
        self.arr.append(value)
```

```python
obj = SomeClass()
obj.create_arr()
obj.insert_to_arr(5)
obj.arr
```

Foi passado apenas 1 argumento na função *insert_to_arr*, por que não deu erro?

Self deixa explícito no Python qual objeto está chamando o método e quais atributos devem ser atualizados

```python
SomeClass.insert_to_arr(obj, 5)
```

Para entender de modo correto seu funcionamento, vamos contruir uma classe. Para tanto, vamos imaginar que a função built-in *set* não exista no python.

Set é uma coleção não ordenada de elementos únicos. Seus usos básicos são testar membresia e eliminar entradas duplicadas

Orientação a Objetos – Construindo uma classe

FIAP MBA+

```
#definição da classe
class Set:
    # Funções membras
    # self se refere ao objeto Set que está sendo usado
    def __init__(self, values=None):
        """
        Este é o construtor
        Ele é chamado quando você cria um novo Set;
        Deve ser usado da seguinte maneira:
        s1 = Set() # set vazio
        s2 = Set([1,2,2,3]) #inicializa com valores
        """
        # cada instância de um Set possui seu próprio dicionário
        # que será usado para rastrear membresia
        self.dict = {}  # cada instância de um Set possui seu próprio dicionário
        # que será usado para rastrear membresia
        if values is not None:
            for value in values:
                self.add(value)
    def __repr__(self):
        """
        Esta é a representação de um objeto Set por meio de uma string
        """
        return "Set: " + str(self.dict.keys())

    # vamos representar membresia como sendo uma chave no self.dict com valor True
    def add(self, value):
        self.dict[value] = True

    # valor está no Set se ele é uma chave no dicionário
    def contains(self, value):
        return value in self.dict

    def remove(self, value):
        del self.dict[value]
```

Agora, podemos instanciar a classe e chamar as devidas funções:

```
s = Set([1,2,3,2,3,1])
s
```

```
Set: dict_keys([1, 2, 3])
```

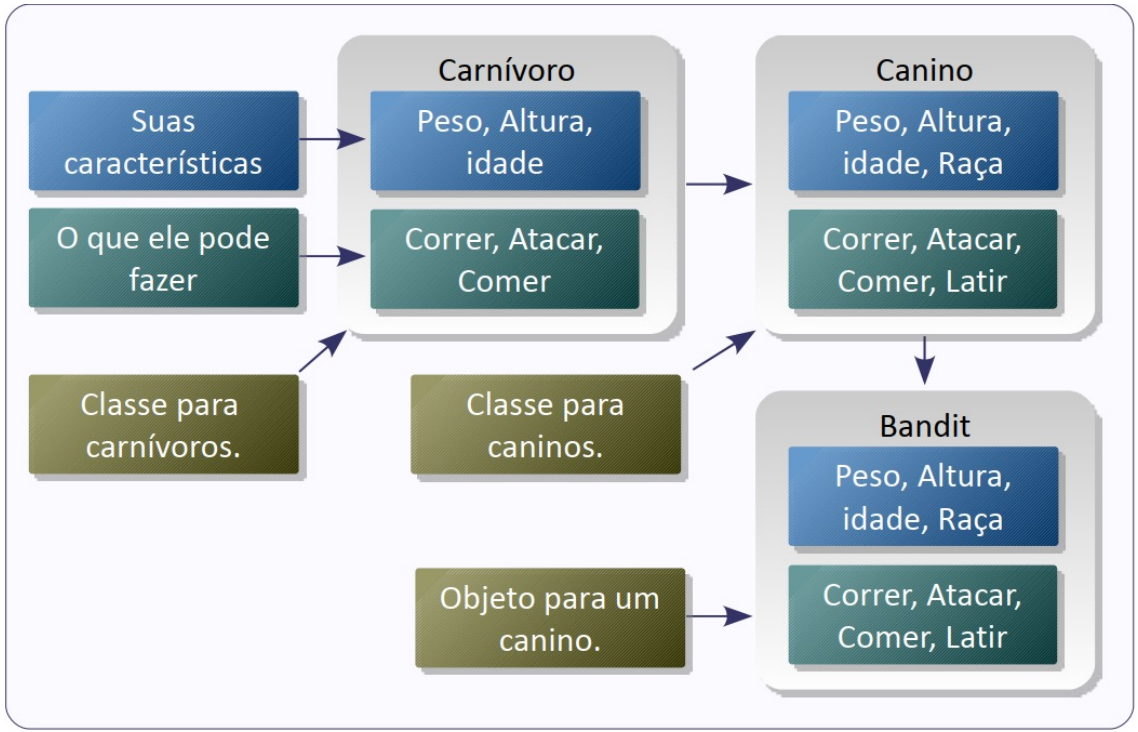
```
s.add(4)
print (s.contains(4)) # True
s.remove(3)
print (s.contains(3)) # False
```

```
True
False
```

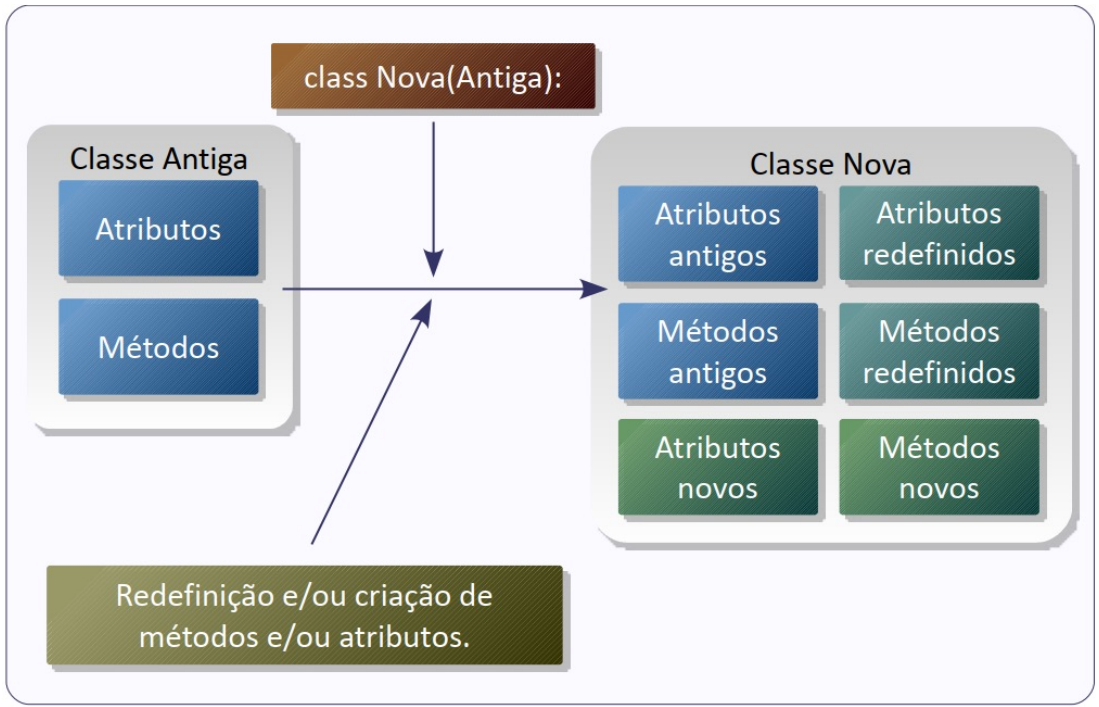
A título de curiosidade, estas são as operações que o método set pode realizar:

Function	Alternative syntax	Description
`a.add(x)`	N/A	Add element `x` to set `a`
`a.clear()`	N/A	Reset set `a` to an empty state, discarding all of its elements
`a.remove(x)`	N/A	Remove element `x` from set `a`
`a.pop()`	N/A	Remove an arbitrary element from set `a`, raising `KeyError` if the set is empty
`a.union(b)`	`a | b`	All of the unique elements in `a` and `b`
`a.update(b)`	`a |= b`	Set the contents of `a` to be the union of the elements in `a` and `b`
`a.intersection(b)`	`a & b`	All of the elements in *both* `a` and `b`
`a.intersection_update(b)`	`a &= b`	Set the contents of `a` to be the intersection of the elements in `a` and `b`
`a.difference(b)`	`a - b`	The elements in `a` that are not in `b`
`a.difference_update(b)`	`a -= b`	Set `a` to the elements in `a` that are not in `b`
`a.symmetric_difference(b)`	`a ^ b`	All of the elements in either `a` or `b` but *not both*
`a.symmetric_difference_update(b)`	`a ^= b`	Set `a` to contain the elements in either `a` or `b` but *not both*
`a.issubset(b)`	`<=`	`True` if the elements of `a` are all contained in `b`
`a.issuperset(b)`	`>=`	`True` if the elements of `b` are all contained in `a`
`a.isdisjoint(b)`	N/A	`True` if `a` and `b` have no elements in common

Herança é um mecanismo que a orientação a objeto provê com objetivo de facilitar o reaproveitamento de código. A ideia é que as classes sejam construídas formando uma hierarquia.



A nova classe pode implementar novos métodos e atributos e herdar métodos e atributos da classe antiga (que também pode ter herdado de classes anteriores), porém estes métodos e atributos podem substituídos na nova classe.



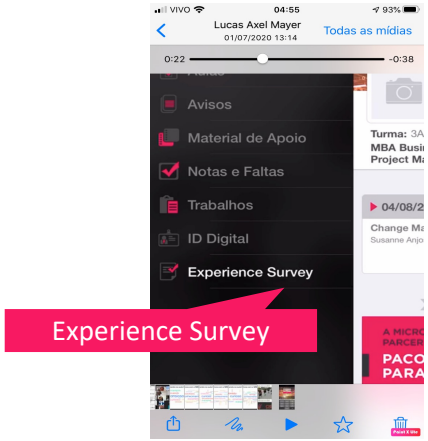
Orientação a Objetos – Herança - Exemplo


```python
class Person:
    def __init__(self, fname, lname):
        self.firstname = fname
        self.lastname = lname

    def printname(self):
        print(self.firstname, self.lastname)

class Student(Person):
    def __init__(self, fname, lname, year):
        super().__init__(fname, lname)
        self.graduationyear = year

    def welcome(self):
        print("Bem-vindo", self.firstname, self.lastname, "à turma de", self.graduationyear)

x = Student("João", "Silva", 2020)
x.welcome()
```

Resolver o exercício no notebook

O que você achou da aula de hoje?

Entrar no aplicativo FIAPP, e no menu clicar em Experience Survey



Ou pelo link: https://fiap.me/Pesquisa-MBA

Obrigado!

profdheny.fernandes@fiap.com.br

/dhenyfernandes


FIAP