Python – Orientação a Objetos – Parte 2

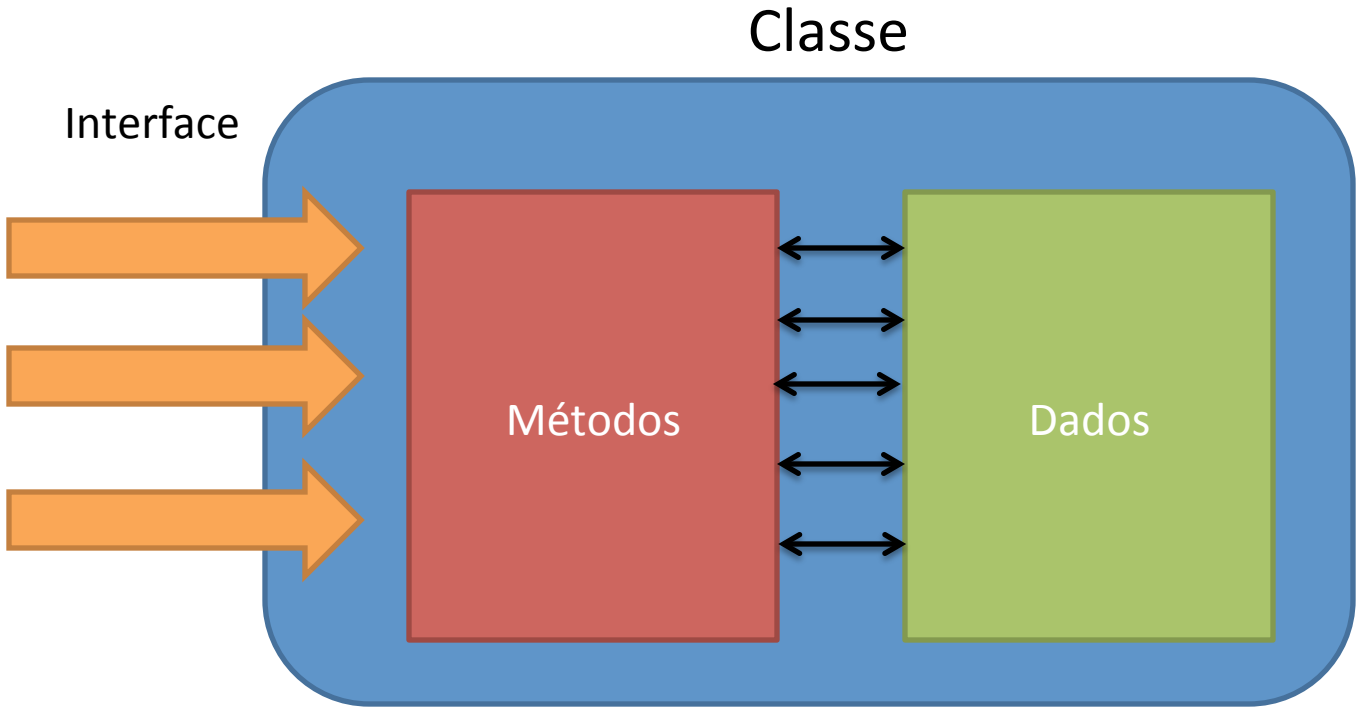
Introdução à Programação

SI1

Encapsulamento

- Na terminologia da orientação a objetos, diz-se que um objeto possui uma **interface**.
- A interface de um objeto é como ele **aparece** para os demais objetos:
  - Suas características, sem **detalhes internos**
- A interface de um objeto define os **serviços** que ele pode realizar e conseqüentemente as **mensagens que ele recebe**
  - **Um objeto é "visto" através de seus métodos**

Encapsulamento



Encapsulamento

- Encapsulamento é a **proteção** dos atributos ou métodos de uma classe.
- Em Python existem somente o public e o private e eles são definidos no próprio nome do atributo ou método.
- Atributos ou métodos iniciado por **dois sublinhados** (underline) são **privados** e todas as outras formas são **públicas**

Exemplo

```
class Teste1:
    a = 1 #atributo publico
    __b = 2 #atributo privado
```

```
>>> t1 = Teste1()
>>> print t1.a
1
>>> print t1.__b
Erro
```

Get e Set

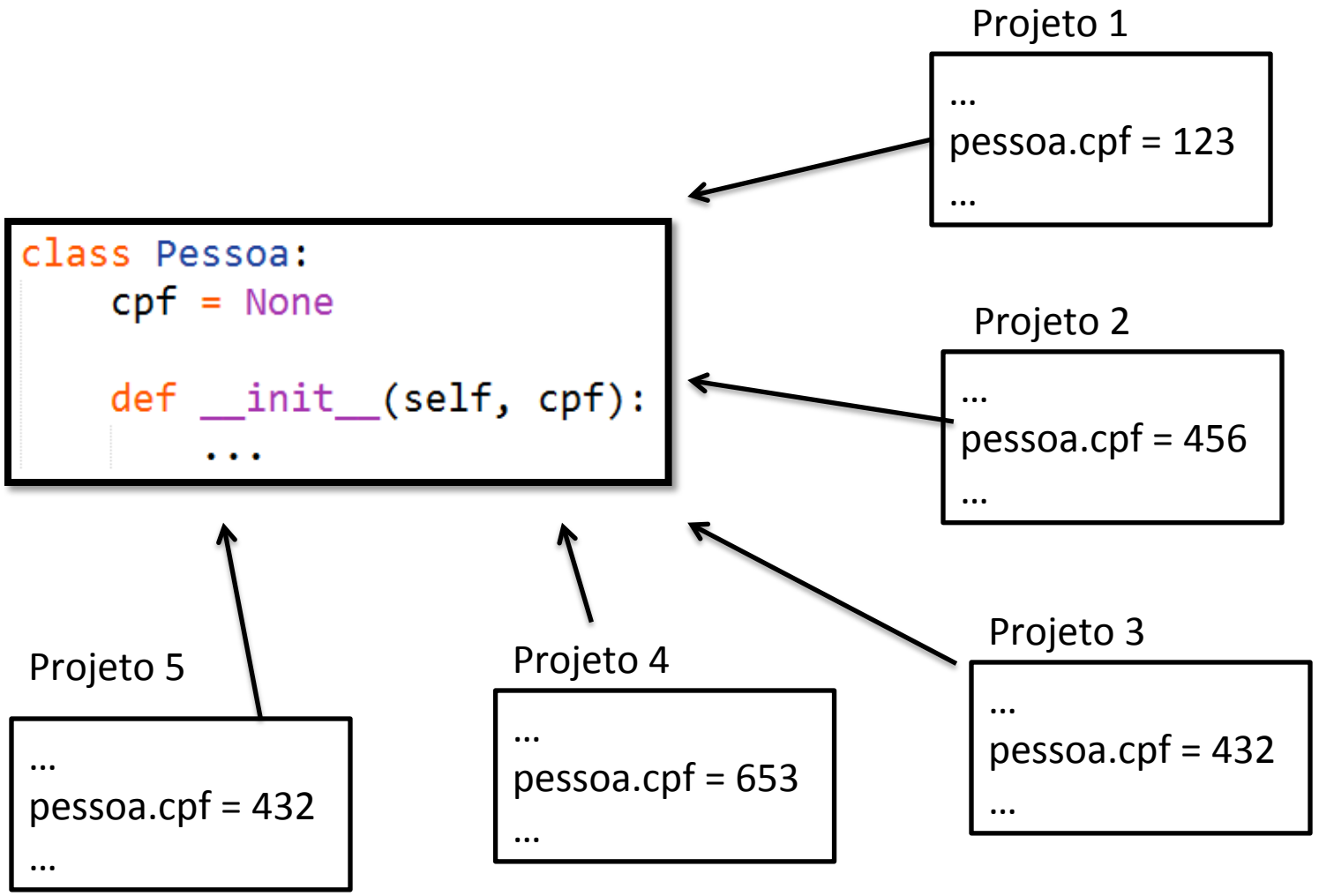
- O que são?
- Pra que servem?

Exemplo – Cenário 1

```
class Pessoa:
    cpf = None

    def __init__(self, cpf):
        ...
```

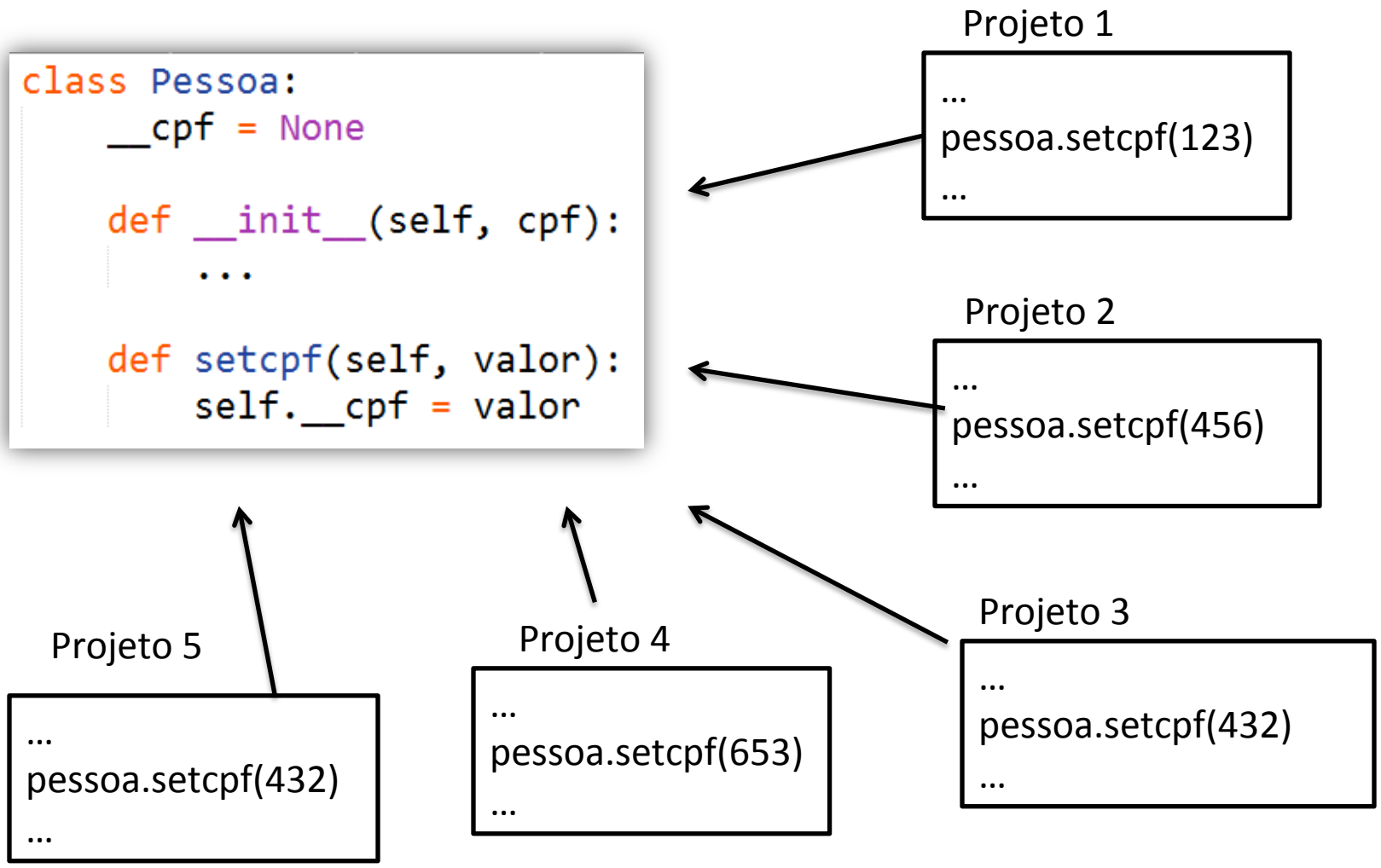
Exemplo – Cenário 1



Cenário 1

- Mudou a forma de atualizar o cpf!
- E agora?
  - Atualizar todos os projetos envolvidos

Cenário 2


class Pessoa:
__cpf = None
def __init__(self, cpf):
...
def setcpf(self, valor):
self.__cpf = valor
Projeto 1
...
pessoa.setcpf(123)
...
Projeto 2
...
pessoa.setcpf(456)
...
Projeto 3
...
pessoa.setcpf(432)
...
Projeto 4
...
pessoa.setcpf(653)
...
Projeto 5
...
pessoa.setcpf(432)
...

Cenário 2

- Mudou a forma de atualizar o cpf!
- E agora?
  - Atualiza **apenas** o método setcpf.

Cenário 2

```
class Pessoa:
    cpf = None

    def __init__(self, cpf):
        ...

    def __validaCPF(self, valor):
        ...

    def setCPF(self, valor):
        if self.__validaCPF(valor):
            self.cpf = valor
        else:
            print("CPF invalido")
```

Encapsulamento/Properties

```
>>> a = C()
>>> a.x = 10
>>> print(a.x)
10
>>> a.x = -10
>>> print(a.x)
0
```

Violação de Encapsulamento?

Que bruxaria é essa? Uso de properties

Properties

```
class C(object):
    def __init__(self, x):
        self.__x = x

    def getx(self):
        return self.__x

    def setx(self, valor):
        if valor >=0:
            self.__x = valor
        else:
            self.__x = 0

    x = property(getx, setx)
```

Atributos Estáticos

- Atributos que compartilham o mesmo valor para todos os objetos da classe.

Atributos da Classe

```
class Funcionario:
    contadorFunc = 0

    def __init__(self, nome, salario):
        self.nome = nome
        self.salario = salario
        Funcionario.contadorFunc += 1

    def mostraContador(self):
        print("Total Funcionarios: %d" % Funcionario.contadorFunc)

emp1 = Funcionario("Katarina", 2000)
emp2 = Funcionario("Garen", 5000)
emp1.mostraContador()
```

Saída>> Total Funcionarios: 2

ALGUNS MÉTODOS E ATRIBUTOS ESPECIAIS NATIVOS

Membros Nativos

- As classes contêm métodos e atributos especiais que são incluídos por Python mesmo se você não os defina explicitamente.
  - Todos os membros nativos tem 2 underscores ao redor dos nomes: __init__ , __doc__

Membros Nativos

- Ex:
  - __repr__ existe para todas as classes e você pode sempre redefiní-lo.
- A definição deste método especifica como tornar a instância de uma classe em uma string.
  - **print f** chama **f.__repr__()** para chamar a representação em string do objeto f

Exercícios

1. Construa uma classe Produto, que deve ter os atributos codigo e preco (privados). Adicione também um atributo estático qtdProd, que deverá ser acrescentado toda vez que um novo objeto é criado.
   – Crie os métodos get e set e teste a classe.

2. Defina a classe GerenciadorProdutos que possui uma lista de produtos (Produto é a classe definida no exercício anterior).

- Essa classe deverá possuir os métodos:
  - AdicionarProduto(Produto)
    - Adiciona o produto na lista de produtos
  - RemoverProduto(codigo)
    - Remove o produto que possui este código da lista de produtos

3. Adicione um novo método na classe GerenciadorProdutos:

- PrecoTotal(listaCodigos)
  - Esse método deve retornar a soma dos preços dos produtos cujos códigos estão na lista recebida como parâmetro.

Exercícios

4. Coloque cada objeto Ponto numa lista.

- Imprima cada elemento da lista

pontos.txt

```
A
100 200
B
130 150
C
500 239
OutroPonto
199 54
```

Exercícios

5. Crie as classes Biblioteca e Livro.
   - A Biblioteca deverá conter uma lista de livros disponíveis e lista de livros alugados
   - A biblioteca deverá possuir um método para alugar um livro. Caso o livro já esteja alugado a pessoa não poderá alugar.
   - A biblioteca deverá possuir um método para devolver o livro.
   - Adapte o código para poder informar o nome do livro mais alugado.