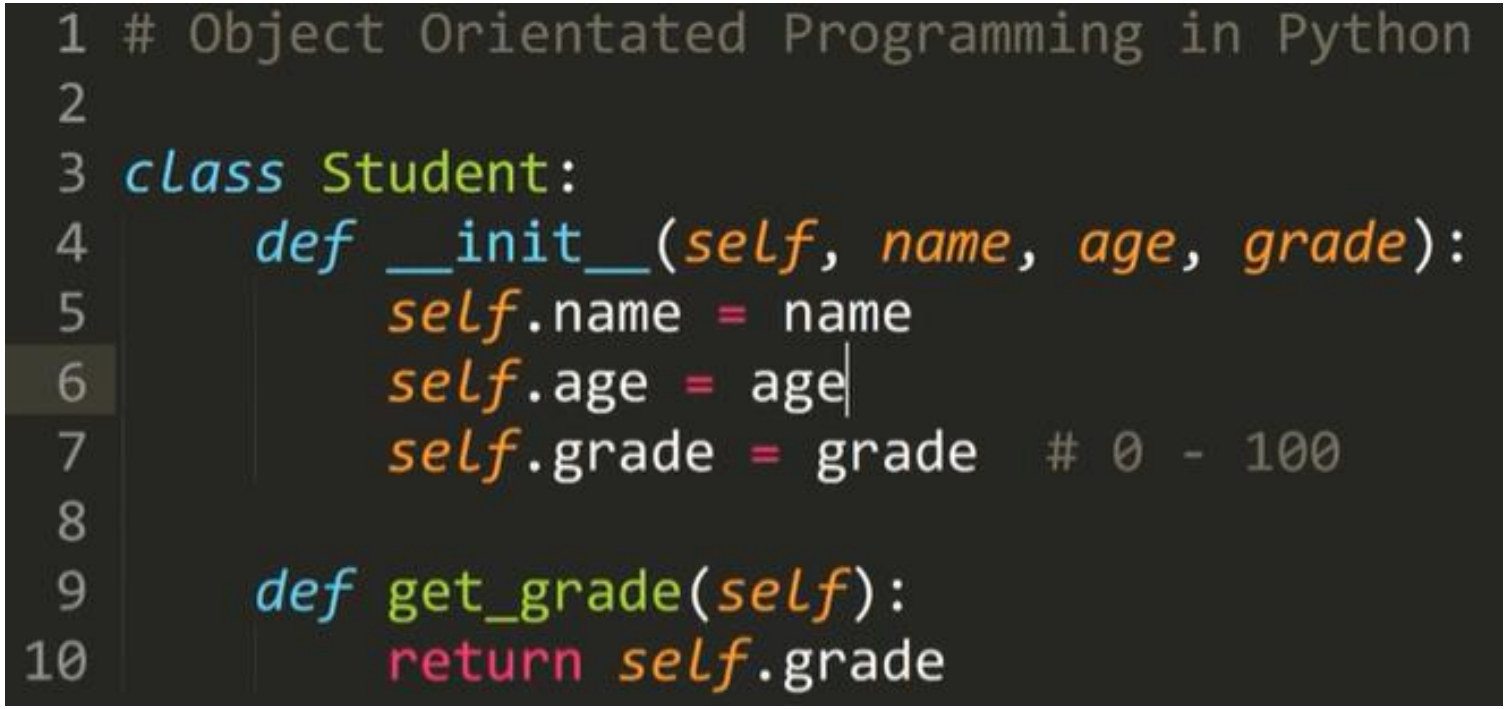
F<BR/C> DE SOFTWARE

UC: PROGRAMAÇÃO ESTRUTURADA E ORIENTADA A OBJETOS

PROFESSOR: THIAGO ALMEIDA

CLASSES EM PYTHON



DEFININDO UMA CLASSE COM PYTHON

```
class Objeto:
    def __init__(self,nome,cor,altura,largura):
        self.nome = nome
        self.cor = cor
        self.dimensão = altura * largura
```

INSTÂNCIA DE UM OBJETO

- Um objeto cujo comportamento e estado são definidos pela classe é denominado de Instância.
- Na Programação Orientada à Objetos uma instância de uma classe é chamada de objeto.
- Para que estes objetos existam na memória são criadas instancias destas classes;
- Uma nova instância da classe é criada usando nomeClasse( )

Exemplo:

```
controle_remoto = Objeto( )
```

VARIÁVEIS DA CLASSE

- As variáveis e os métodos são escritos precedidos pelo nome da classe e por um ponto (.)
- A variável nome definida na classe Objeto é escrita Objeto.nome

Exemplo:

```
print(Objeto.nome)
```

CONSTRUTORES

- Toda classe é uma fábrica de objetos, para isso a classe sempre inicia o método construtor, responsável por definir os atributos.
- O Python suporta construtores que podem ser chamados automaticamente na criação de instâncias
  - Basta definir na classe um método chamado __init__
  - Este método é chamado automaticamente durante a criação de um nova instância da classe, sendo que os argumentos são passados entre parênteses após o nome da classe

ATRIBUTOS

- Um atributo nome associado a uma instância Pessoa tem nome Pessoa.nome
- Se queremos nos referir a um atributo nome de um objeto dentro da própria classe, usamos o nome self.nome

Exemplo:

```
class Pessoa:
    def __init__(self,nome,idade):
        self.nome = nome
        self.idade = idade
    def mostrar_nome(self):
        return self.nome
```

SELF

- Os métodos sempre têm self como primeiro argumento
- self se refere a uma instância da classe
- Uma nova instância da classe é criada usando nomeClasse( )

```
class Pessoa:
    def __init__(self,nome,idade):
        self.nome = nome
        self.idade = idade
    def mostrar_nome(self):
        return self.nome
```

MÉTODOS

- Os métodos especificam a maneira pela qual os dados de um objeto são manipulados.
- Uma especificação dos passos pelos quais uma operação deve ser executada.
- Ele é um função para implementação das operações/ações de um objeto.
- Os métodos de um tipo de objeto referenciam somente as estruturas de dados desse tipo de objeto.
- Exemplo:

```
class Pessoa:
    def __init__(self,nome,idade):
        self.nome = nome
        self.idade = idade
    def mostrar_nome(self):
        return self.nome
```

EXEMPLO DE MÉTODOS

```python
class Quadrado:

    def __init__(self, lado_a, lado_b):
        self.__lado_a = lado_a
        self.__lado_b = lado_b
        print("Criada uma nova instância de um Quadrado")

    def get_lado_a(self):
        return self.__lado_a

    def calcula_area(self):
        return self.__lado_a * self.__lado_b

q1 = Quadrado(4,4)
print(q1.calcula_area())
```

EXEMPLO DE MÉTODOS

```
obj = Quadrado(8,8)
#Criada uma nova instancia Retangulo

#Chamada de função para calcular a área
obj.calcula_area()

#chamara de função para calcular o perímetro
obj.calcula_perimetro()
```

EXEMPLO CLASS PESSOA

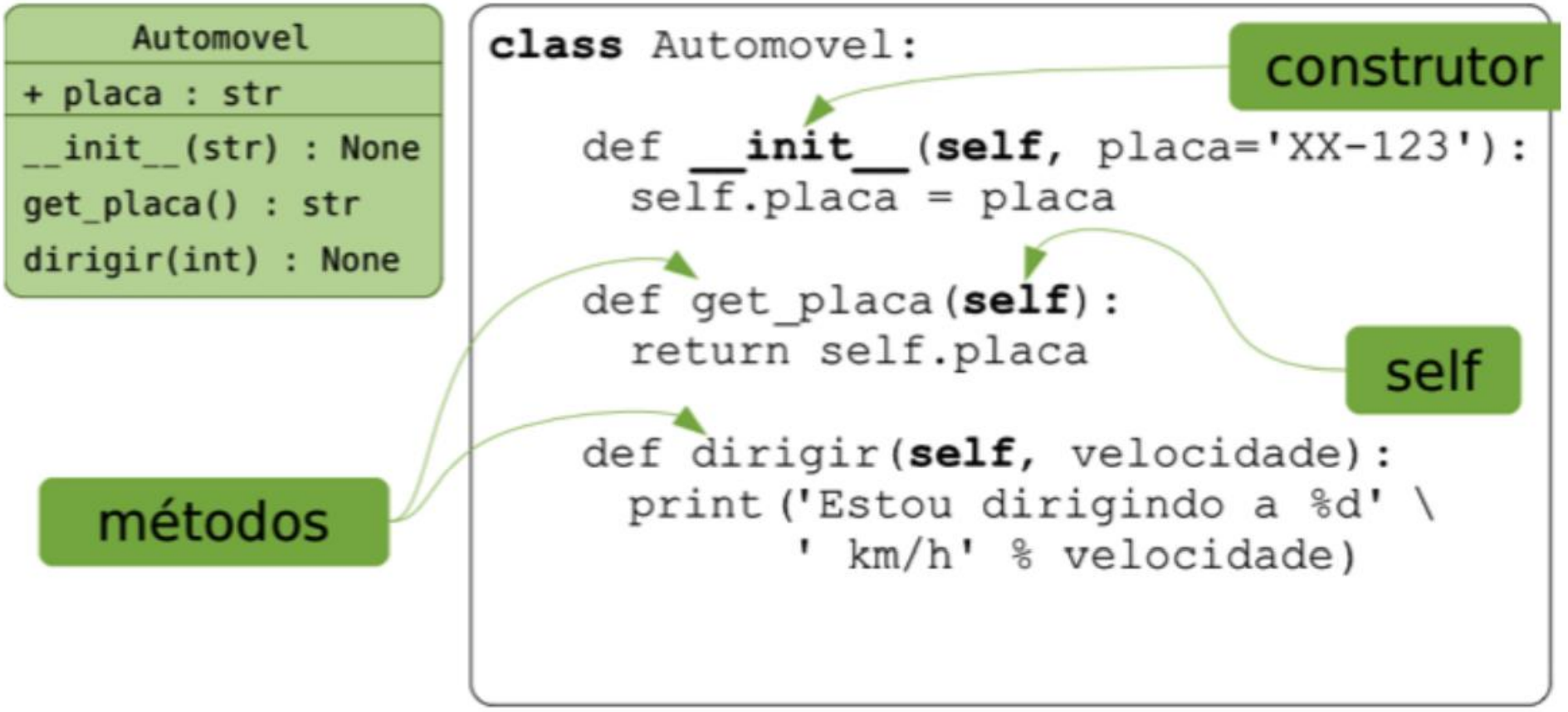
```
class Pessoa:
    nome = None
    idade = None

    def __init__(self, nome, idade):
        self.nome = nome
        self.idade = idade

    def getAnoNascimento(self, anoAtual):
        return anoAtual - idade

pessoa = Pessoa("Pedro", 21)
print(pessoa.getAnoNascimento(2013))
```

EXEMPLO CLASS AUTOMÓVEL



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 iniciados por no máximo dois sublinhados (underline) são privados e todas as outras formas são públicas.

ENCAPSULAMENTO

- O código de impressão abaixo irá retornar um erro:

```
class Retangulo:

    def __init__(self, lado_a, lado_b):
        self.__lado_a = lado_a
        self.__lado_b = lado_b
        print("Criada uma nova instância Retangulo")

    def get_lado_a(self):
        return self.__lado_a

    def calcula_area(self):
        return self.__lado_a * self.__lado_b

ret1 = Retangulo(8,12)
print(ret1.__lado_a)
```

EXERCÍCIOS

- 1 - **Classe Pessoa**: Crie uma classe que modele uma pessoa. Esta classe deve possuir os seguintes atributos:
  - Nome
  - Idade
  - Endereço
- **A classe deve ter os seguintes métodos:**
  - Mostrar nome;
  - Alterar a idade;
  - Imprimir endereço.

EXERCÍCIOS

2 - **Classe Livro**: Crie uma classe que modele um Livro. Esta classe deve possuir os seguintes atributos:

- Nome
- Autor
- Editora
- Páginas

- **A classe deve ter o seguintes métodos:**
  - Alterar_editora()
  - Listar_qtde_paginas()

EXERCÍCIOS

3 - **Classe Aluno**: Crie uma classe que modele um aluno. Esta classe deve possuir os seguintes atributos:

- Nome;
- RA;
- Nota 1, nota 2, nota 3, nota 4;

- **A classe deve ter o seguintes método:**
  - **Mostrar_situacao**: (APROVADO / EXAME / REPROVADO). Considere que, nesse caso, a média final é calculada pela média aritmética simples de todas as notas e que o aluno é aprovado somente se obtiver média maior ou igual a sete. Exame entre 5 e 6.9. Reprovado abaixo de 5
  - A situação será retornada a partir das notas obtidas pelo aluno.

EXERCÍCIOS

4 - **Classe Conta**: Um banco mantém contas de clientes armazenando o número da conta, o nome do cliente e o saldo atual da conta. Crie uma classe que modele um Conta-Corrente.

- Nome;
- CPF;
- Numero;
- Saldo;

- **A classe deve ter o seguintes métodos:**
  - Depositar()
  - Sacar() - *OBS: somente enquanto a conta possuir saldo positivo*
  - *Imprimir saldo()*

EXERCÍCIOS

4 - **Classe Funcionário**: Uma empresa quer criar um controle de pagamento para os funcionários. Crie uma classe que modele um Funcionário com os seguintes atributos e métodos:

- Nome; Sobrenome; Horas_trabalhadas; Valor_hora;

- **A classe deve ter o seguintes métodos:**
  - O método nomeCompleto deve escrever na tela o atributo nome concatenado ao atributo sobrenome.
  - O método calcularSalario faz o cálculo de quanto o funcionário irá receber no mês, multiplicando o atributo horasTrabalhadas pelo atributo valorPorHora. Em seguida, escreve o valor na tela.
  - O método incrementarHoras adiciona um valor passado por parâmetro ao valor já existente no atributo valorPorHora.

EXERCÍCIOS

5 - **Classe Círculo**: crie uma classe que represente um círculo. Esta classe deve possuir o seguinte atributo:

- raio

- A classe deve ter os seguintes métodos:
  - imprimir o valor do raio;
  - calcular a área do círculo;
  - calcular a circunferência do círculo.

EXERCÍCIOS

6 - **Classe Agenda**: crie uma classe que represente uma agenda de tarefas. Esta classe deve possuir os seguintes atributos:

- Dia;
- Mês;
- Ano;
- Anotação;

- A classe deve ter os seguintes métodos:
  - validar_data();
  - anotar_tarefa();
  - Mostrar_anotação();

EXERCÍCIOS

7 - **Classe Triangulo**: Crie uma classe que modele um triangulo:

  - Atributos: LadoA, LadoB, LadoC
  - Métodos: calcular Perímetro, getMaiorLado;

- Crie instâncias desta classe. Ele deve pedir ao usuário que informe as medidas de um triangulo. Depois, deve criar um objeto com as medidas e imprimir sua área e maior lado.

EXERCÍCIOS

8 - **Classe Aluno_Academia**: Uma academia mantem registro de seus alunos e precisa armazenar os seguintes atributos:

- Nome, Idade, Peso, Altura, Mensalidade (valor default: R$ 120,00)

- A academia faz um desconto especial para menores de idade, portanto, é necessário saber distinguir entre um aluno maior e menor. Além disso, a academia também tem interesse em acompanhar o desempenho de seus alunos, por isso, ela também necessita conhecer o índice de massa corporal (IMC) deles, para isso a classe deve ter os seguintes métodos:
  - Calcular_IMC()
  - Obter_valor_mensalidade()

EXERCÍCIOS

9 - **Classe Carro**: Crie uma classe que modele um Carro. Esta classe deve possuir os seguintes atributos:

- Modelo, Marca, Cor, Ano, Valor, Nível (default 5) , Consumo

- **A classe deve ter o seguintes métodos:**
  - ligar(); - para andar o carro deve estar ligado, use variável booleana.
  - abastecer(); - deve incrementar no nível do tanque de combustível
  - andar(); - recebe a distancia em km que o veículo andou
  - verificar_nível(); - o deve criar uma forma de calcular quantos litros de comb. foram gasto por km
  - calcular_imposto() - deve considerar o IPVA do carro em 2,5% do valor.

EXERCÍCIOS

10 - **Classe NF**: Crie uma classe que modele uma Nota_Fiscal. Esta classe deve possuir os seguintes atributos:

- **Numero; Tipo (Entrada/Saída); Série (1,2 ou 3); CNPJ; Razão Social; Data; Valor Produtos; ICMS; Frete; IPI; Valor total;**

- **A classe deve ter o seguintes métodos:**
  - obterNumero();
  - obterDataEmissão();
  - alterarRazaoSocial();
  - calcularValorTotal() – somar valor do produto + frete e impostos e armazenar na variável valor_total.