Laboratório de Programação 1

Aula 13

Mário Hozano
professor@hozano.com

Ciência da Computação
UFAL - Arapiraca

Relembrando a aula anterior...

- O que são arquivos?
- Como criar um arquivo?
- É possível alterar um arquivo existente*?*
- Como ler a 3a linha de um arquivo?
- Para que serve o módulo *pickle*?
- O que são Exceções?

Roteiro da aula

- Classes e Objetos
- Definindo Classes
- Criando Atributos
- Métodos
- Métodos Especiais

Classes e Objetos

- Durante o curso utilizamos diversos tipos de objetos oferecidos no Python
- Utilizamos as propriedades e métodos de Strings, Listas, Dicionários, Arquivos etc.
- Em Python é possível criar o seu próprio tipo de objeto (**classes**)
- As classes definem o tipo (fôrma) com suas características (**atributos**) e ações/comportamentos (**métodos**)

Definindo Classes

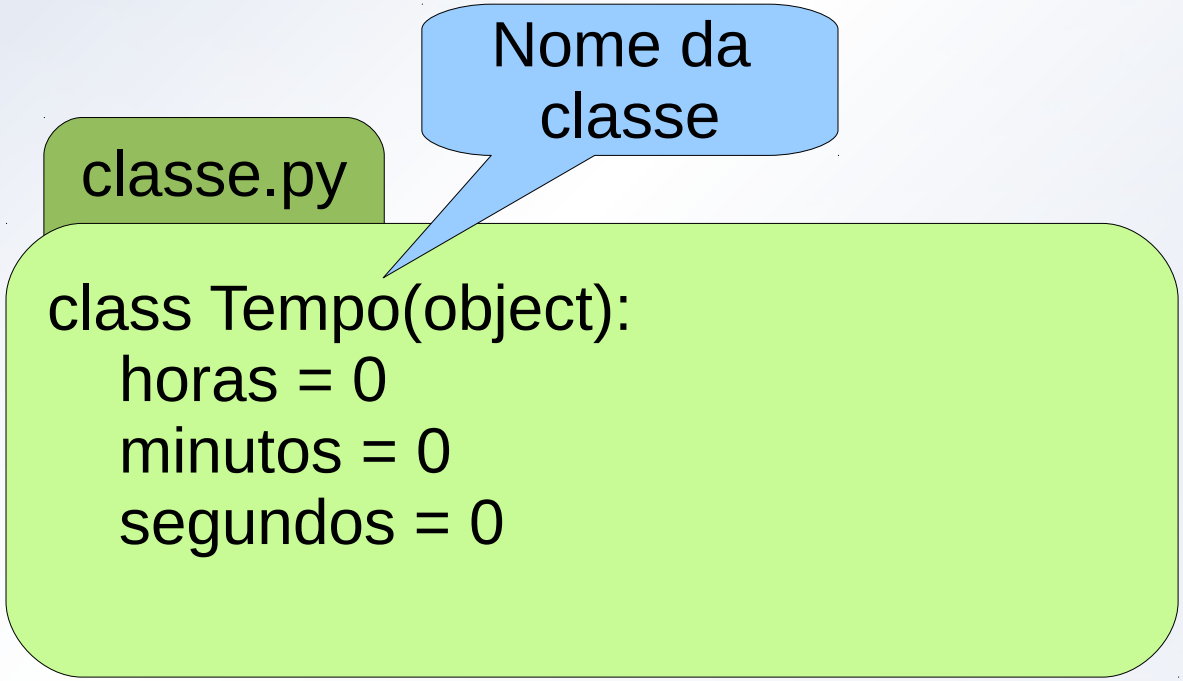
- Definição de uma Classe Simples (Tempo)

classe.py

```
class Tempo(object):
    horas = 0
    minutos = 0
    segundos = 0
```

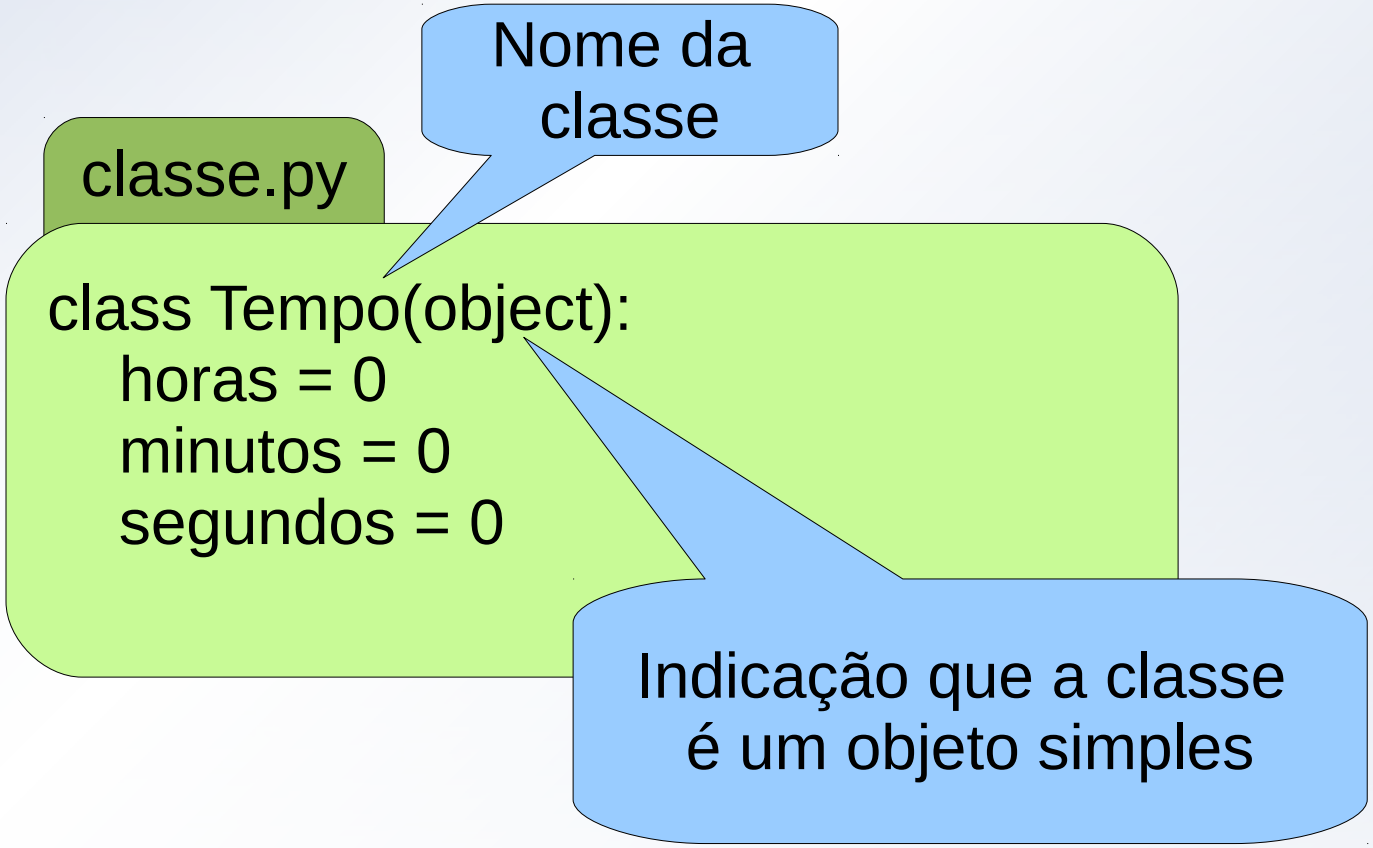
Definindo Classes

- Definição de uma Classe Simples (Tempo)



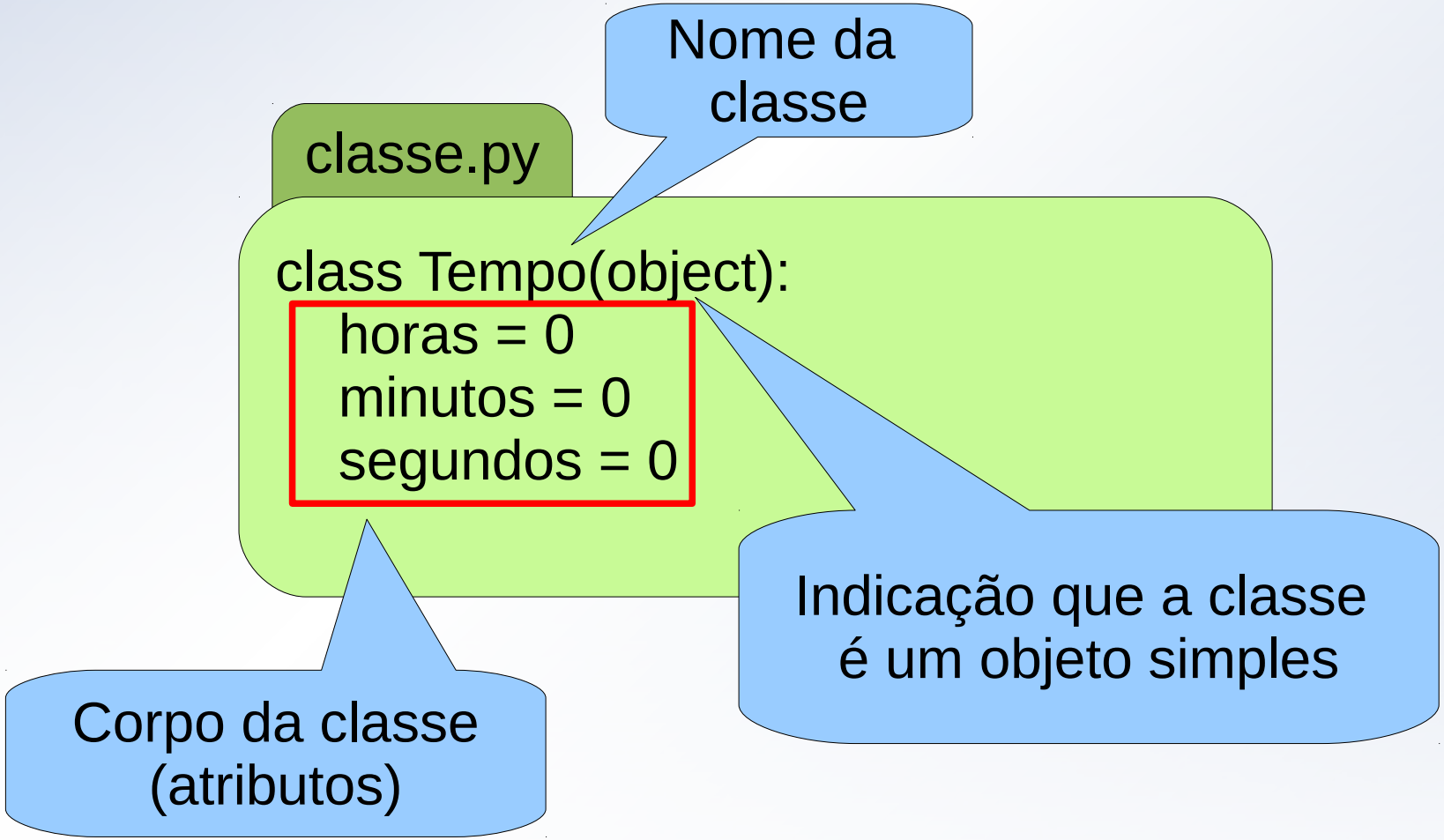
Definindo Classes

- Definição de uma Classe Simples (Tempo)



Definindo Classes

- Definição de uma Classe Simples (Tempo)



Atributos de Classes

- Atributos são valores associados aos objetos

- Normalmente eles definem características inerentes aos objetos

```
class Tempo(object):
    horas = 0
    minutos = 0
    segundos = 0
```

```
class Aluno(object):
    nome = ''
    matricula = ''
    masculino = True
```

```
class Carro(object):
    num_portas = 4
    motor = 1.4
    ano = 2000
```

```
class Contato(object):
    nome = ''
    telefone = ''
    endereco = ''
```

Instância de Classes

- Uma vez definidas as classes (tipos), podemos criar **objetos** com as mesmas

tempo.py

```
class Tempo(object):
    horas = 0
    minutos = 0
    segundos = 0

t = Tempo()
print(type(t))
t.horas = 15
t.minutos = 50
print(t.horas)
```

Instância de Classes

- Uma vez definidas as classes (tipos), podemos criar **objetos** com as mesmas

tempo.py

```
class Tempo(object):
    horas = 0
    minutos = 0
    segundos = 0

t = Tempo()
print(type(t))
t.horas = 15
t.minutos = 50
print(t.horas)
```

O que acontece aqui ?

Instância de Classes

- Uma vez definidas as classes (tipos), podemos criar **objetos** com as mesmas

tempo.py

```
class Tempo(object):
    horas = 0
    minutos = 0
    segundos = 0

t = Tempo()
print(type(t))
t.horas = 15
t.minutos = 50
print(t.horas)
```

O que acontece aqui ?

E aqui ?

Funções com Classes

Como criar uma função para somar dois objetos do tipo **Tempo**?

Funções com Classes

tempo.py

```
def soma_tempo(t1, t2):
    soma = Tempo()
    soma.horas = t1.horas + t2.horas
    soma.minutos = t1.minutos + t2.minutos
    soma.segundos = t1.segundos + t2.segundos

   if soma.segundos > 60:
        soma.minutos += 1
        soma.segundos -= 60

    if soma.minutos > 60:
        soma.horas += 1
        soma.minutos -= 60

    return soma
```

Funções com Classes

tempo.py

```
def soma_tempo(t1, t2):
    soma = Tempo()
    soma.horas = t1.horas + t2.horas
    soma.minutos = t1.minutos + t2.minutos
    soma.segundos = t1.segundos + t2.segundos

   if soma.segundos > 60:
        soma.minutos += 1
        soma.segundos -= 60

    if soma.minutos > 60:
        soma.horas += 1
        soma.minutos -= 60

    return soma
```

Esta função serve para somar outros tipos de objetos?

Métodos de Classes

- Atributos são inerentes estruturas que representam características inerentes aos objetos
- Métodos representam ações/comportamentos inerentes aos objetos definidos por uma classe
- Em outras aulas já vimos métodos inerentes às classes de Python:
  - Strings: s.lower(), s.upper(), s.capitalize()
  - Listas: l.append(), l.remove(), l.sort()
  - Dicionários: d.items(), d.pop(), d.keys()

Métodos de Classes

- A criação de métodos em classes é similar com a criação de funções (parâmetros, valores de entrada e retorno...)
- Todo método deve indicar como primeiro parâmetro de entrada a variável *self*
- Através do *self* é possível acessar todos os atributos e métodos do objeto da classe
- Isso permite trabalhar com as variáveis de escopo local em cada método

Métodos de Classes

tempo.py

```
class Tempo(object):
    horas = 0
    minutos = 0
    segundos = 0

    def string_tempo(self):
        return “%s:%s:%s” %(self.horas, self.minutos, self.segundos)

t = Tempo()
t.horas = 11
t.minutos = 35
t.segundos = 40

print(t.string_tempo())
```

Métodos de Classes

tempo.py

```
class Tempo(object):
    horas = 0
    minutos = 0
    segundos = 0

    def string_tempo(self):
        return “%s:%s:%s” %(self.horas, self.minutos, self.segundos)

t = Tempo()
t.horas = 11
t.minutos = 35
t.segundos = 40

print(t.string_tempo())
```

O que acontece Aqui?

Métodos de Classes

Como utilizar a função soma_tempo() como um método?

t1.soma_tempo(t2)

Métodos de Classes

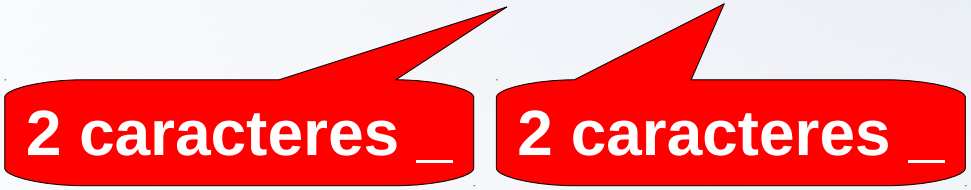
tempo.py

```
class Tempo(object):
    horas = 0
    minutos = 0
    segundos = 0

    def soma_tempo(self, t2):
        self.horas += t2.horas
        self.minutos += t2.minutos
        self.segundos += t2.segundos
        if self.segundos > 60:
            self.minutos += 1
            self.segundos -= 60
        if self.minutos > 60:
            self.horas += 1
            self.minutos -= 60
```

Métodos Especiais - Construtor

- Existem situações em que a criação de um objeto depende da indicação de alguns valores para seus atributos
- As classes podem definir métodos construtores que processam tais valores durante a criação do objeto
- Em python os construtores das classes devem ser definidos como métodos com o nome _ _*init*_ _



Métodos Especiais - Construtor

tempo.py

```
class Tempo(object):

    def __init__(horas, minutos, segundos):
        self.horas = horas
        self.minutos = minutos
        self.segundos = segundos

t = Tempo(11, 20, 34)
```

Agora é obrigatória a indicação de horas, minutos e segundos na construção (instanciação) de um objeto Tempo

Métodos Especiais – Representação String

- Todo objeto (o) deve apresentar uma representação string para ser impresso quando chamado com a instrução *print*

```
>>> t = Tempo(11,20,34)
>>> print t
<__main__.Tempo instance at 0x24065a8>
```

- A indicação desta representação pode ser alterada utilizando o método especial _ _*str*_ _

Métodos Especiais – Representação String

```
class Tempo(object):

    def __init__(horas, minutos, segundos):
        self.horas = horas
        self.minutos = minutos
        self.segundos = segundos

    def __str__(self):
        return “%s:%s:%s” % (self.horas, self.minutos, self.segundos)
```

```
>>> t = Tempo(11,20,34)
>>> print(t)
11:20:34
```

Exercícios

1. Crie uma classe para representar uma conta bancária (atributos: número, saldo; operações: saldo, saque e deposito)

2. Crie uma classe cliente (atributos: nome, CPF e um conjunto de contas; operações: adicionar conta, remover conta e listar contas)

3. Crie uma classe Agência Bancaria que contenha vários clientes, ela deve ser capaz de listar todos os clientes, dado um identificador de um cliente mostrar todas as contas do mesmo.