Fundamentos de Programação

1º Semestre

2023/2024

Aulas 15 e 16

Joana Martinho Costa, António Raimundo, Paulo Vieira, João Monge, Thiago Pontes

Plano

- Métodos
- Variáveis Globais e Locais
- Ciclos
- Classes e objetos

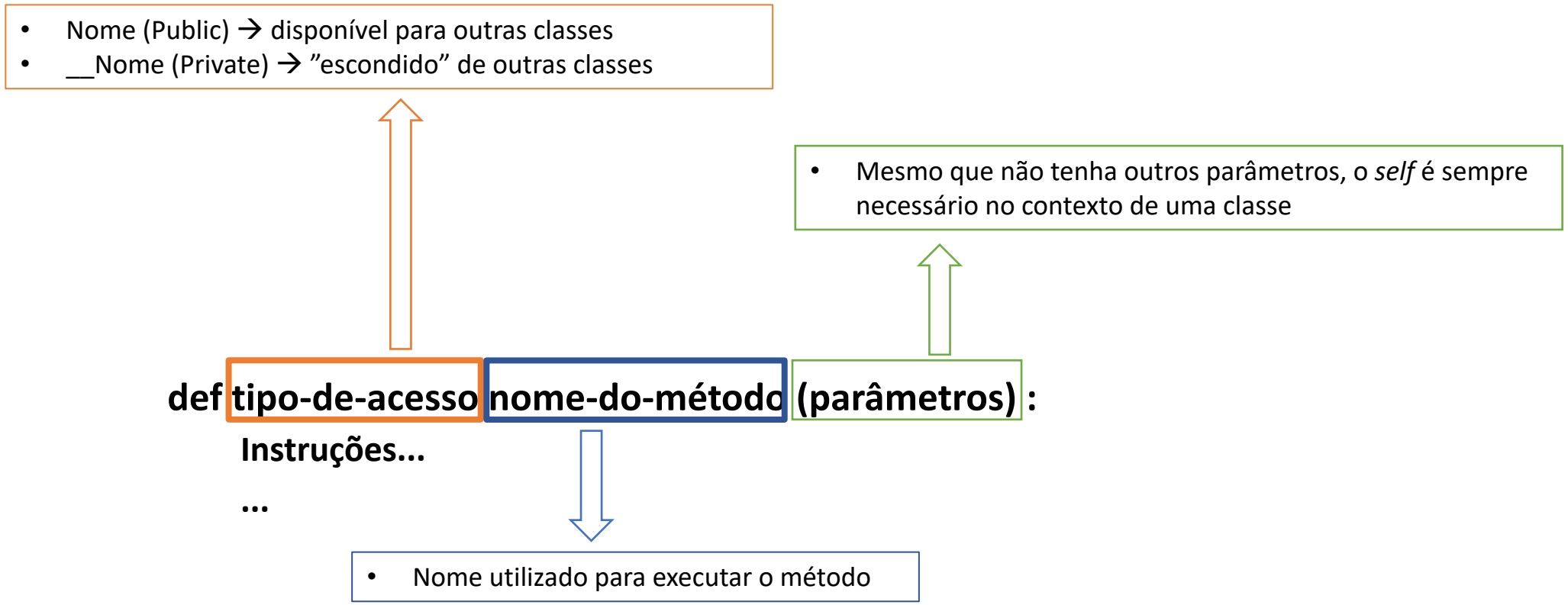
Métodos

- Um método é um conjunto de instruções agrupadas, como se fossem um bloco de código "autónomo"
  - Em Alice, os métodos correspondem às funções e aos procedimentos (incluindo o myfirst**Method**)

```
def my_first_method(self):
    print("Aula 15")
    print("*******")
    print("Aula 16")
```

Métodos

- Os métodos em Python têm a seguinte estrutura **base**:



Métodos

- Exemplos

Os novos métodos são chamados como se fossem instruções (igual ao software Alice)

```
idade = 10
def actualizar_idade(nova_idade):
    idade = nova_idade
```

```
def main():
    actualizar_idade(20)
```

```
def menor_numero(n1, n2):
    if n1 < n2:
        return n1
    else:
        return n2
```

```
def main():
    n = menor_numero(50, 45)
```

Métodos

- Exemplo completo

```
class Calculadora:
    def soma(self):
        a = int(input("Introduza um número"))
        b = int(input("Introduza um número"))
        return a + b


calculadora = Calculadora()
resultado = calculadora.soma()
print(resultado)
```

Variáveis Globais e Locais

- Variável Local
  - Variável criada dentro de um método. Só existe dentro do método e não é reconhecida fora desse âmbito

```
def actualizar_idade(nova_idade):
    idade = nova_idade
```

- Variável Global
  - Variável criada fora dos métodos. Fica acessível para todos e pode ser utilizada e alterada em qualquer método.

```
idade = 10
def actualizar_idade(nova_idade):
    idade = nova_idade
```

Variáveis Globais e Locais

```
class CalcularIdade:

    ANOACTUAL = 2022
    idade = 0

    def saber_idade(self):
        ano_nascimento = int(input("Em que ano nasceu?"))
        self.idade = self.ANOACTUAL - ano_nascimento

calcular_idade = CalcularIdade()
calcular_idade.saber_idade()
print(calcular_idade.idade)
```

Utiliza a constante *ANOATUAL* para guardar o ano em que estamos (se quisermos reutilizar o programa em 2023, só alteramos as constantes e não o código)

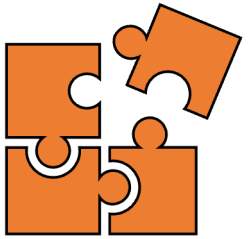
Calcula a idade e guarda o resultado na variável global *idade*

Depois de chamar o método *saber_idade()*, a variável *idade* ficará atualizada

Tarefa 12

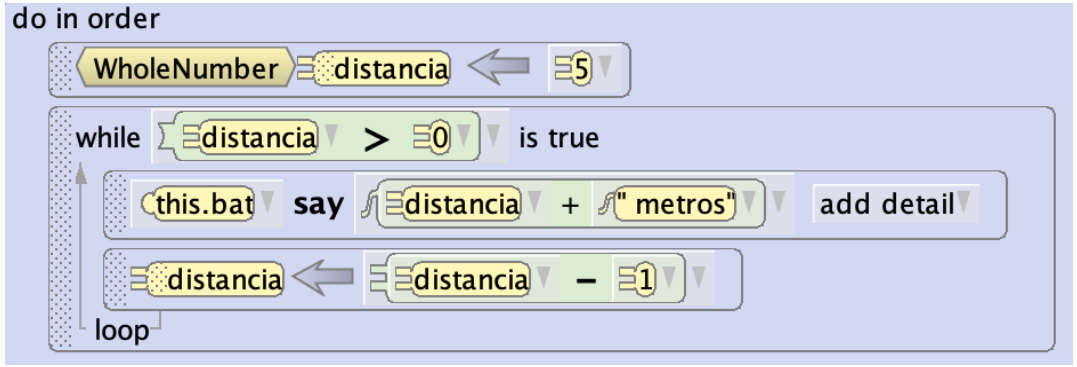
Trabalhar sobre o seguinte documento:

- **Python Tarefa 12 – Métodos.pdf**



Ciclos

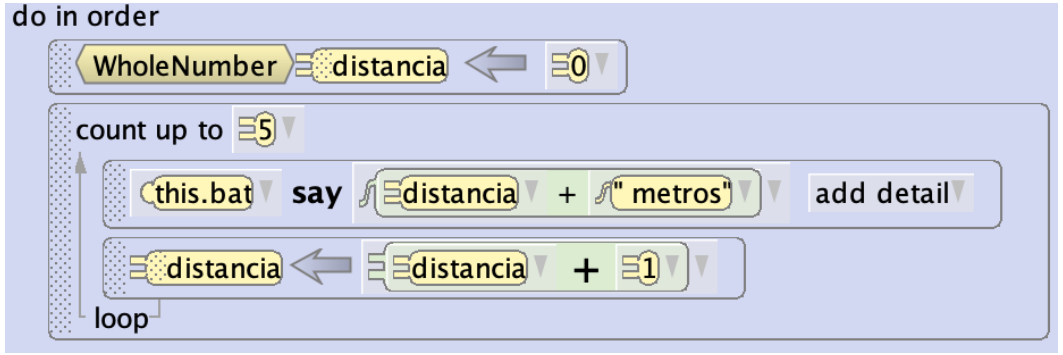
- **while**




```
distancia = 5
while distancia > 0:
    print(distancia, "metros")
    distancia = distancia - 1
```

Ciclos

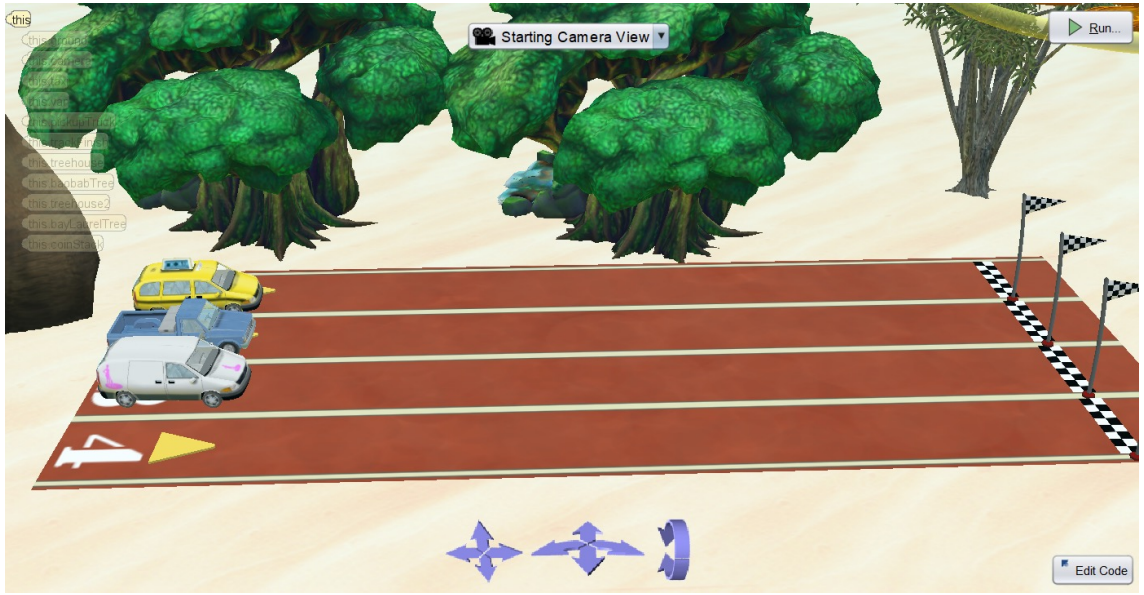
- **for**
  - O ciclo for tem 3 partes:
    - i = 0 → nome da variável contador (neste exemplo, é o i) e o valor de partida (neste exemplo, é o 0)
    - i < 5 → condição para sair do ciclo (neste exemplo, quando o i chegar a 5)
    - i++ → significa i = i+1, é a instrução que vai executar sempre que dá uma volta ao ciclo (neste exemplo, o i aumenta 1 unidade)



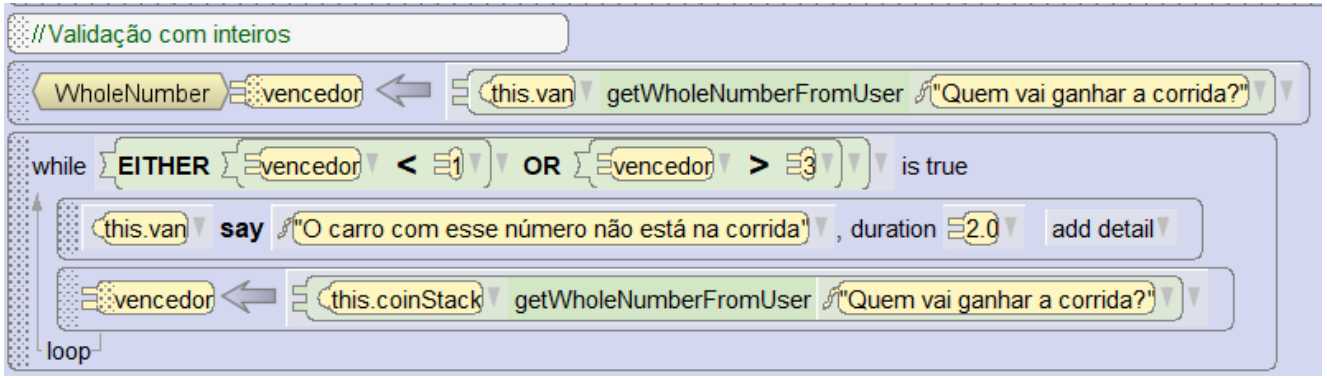

```
distancia = 5
for i in range(5):
    print(distancia, "metros")
    distancia = distancia - 1
```

Validação de *inputs* (1)

- Problema 1:
  - Vencedor deve ser o carro 1, 2 ou 3
    - Qualquer outro valor é inválido neste contexto
- Problema 2:
  - E se o *input* fosse textual?



Validação de *inputs* (2)




```
Quem vai ganhar a corrida?5
O Carro com esse número não está na corrida
Quem vai ganhar a corrida?4
O Carro com esse número não está na corrida
Quem vai ganhar a corrida?3
```

```
vencedor = int(input("Quem vai ganhar a corrida?"))
while vencedor < 1 or vencedor > 3:
    print("O Carro com esse número não está na corrida")
    vencedor = int(input("Quem vai ganhar a corrida?"))
```

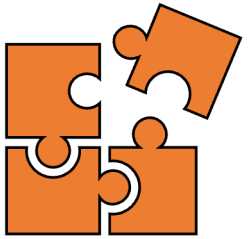
Validação de *inputs* (3)

```
str_input = input("Introduz a tua idade: ")
while not str_input.isdigit():
    print("Introduziste um valor inválido.")
    str_input = input("Introduz a tua idade: ")
number_input = int(str_input)
print("A tua idade é: ", number_input)
```

Tarefa 13

Trabalhar sobre o seguinte documento:

- **Python Tarefa 13 – Ciclos.pdf**



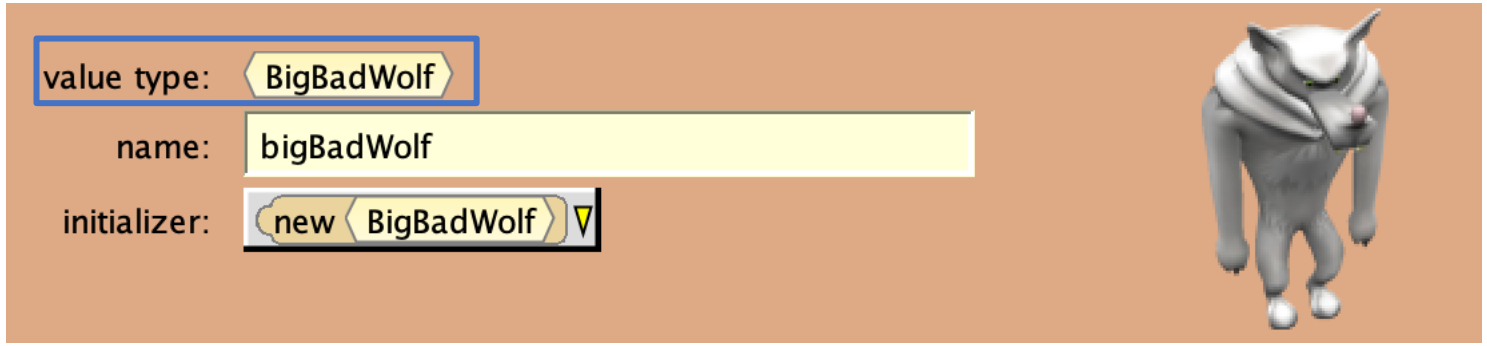
Classes

- Tal como em Alice, utilizávamos várias personagens, em Python utilizamos várias classes.

- Uma classe é um tipo de dado, tal como os tipos primitivos *int* ou *double*.
  - A diferença principal é que os tipos de dados primitivos não possuem métodos

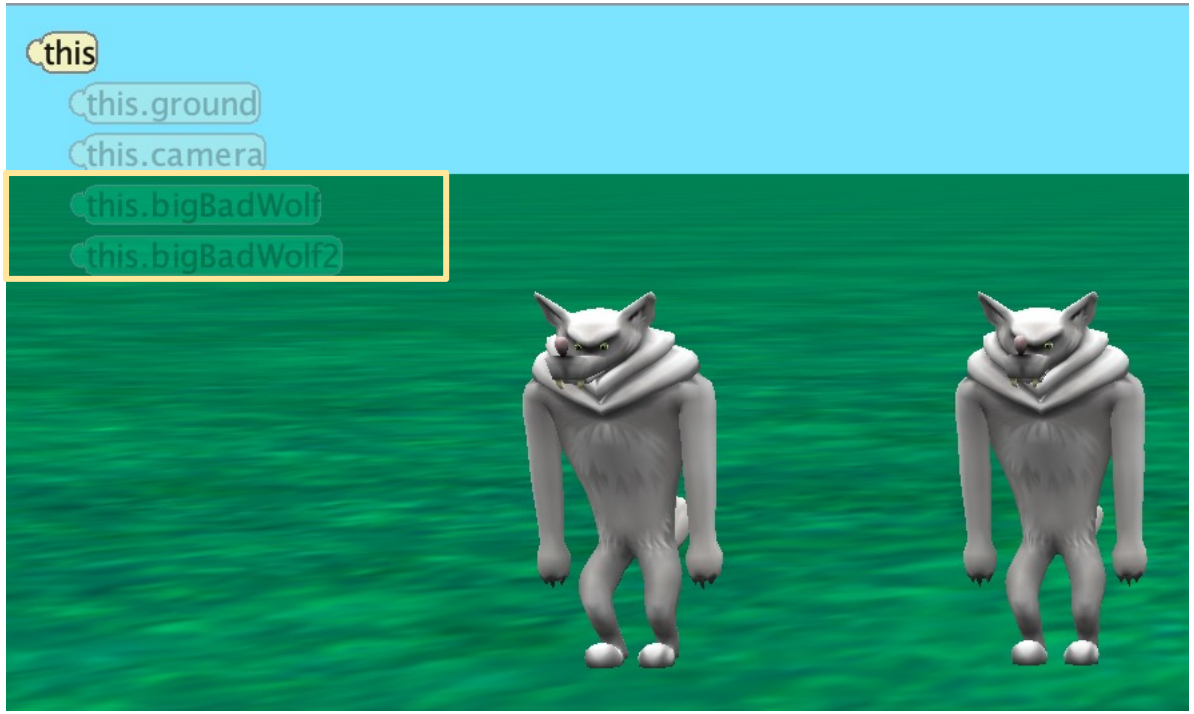
Classes

- Regressemos ao software Alice...
  - Ao criar um Lobo, o que o software faz internamente é ir criar uma instância da Classe BigBadWolf



Objetos

- Uma instância de uma classe é chamada de Objeto
  - Ao criarmos 2 lobos iguais, repara que são criados com nomes diferentes! Isto significa que, existem dois objetos: instâncias da classe BigBadWolf (o bigBadWolf e o bigBadWolf2) com as mesmas características



Classes em Python

- Uma classe em Python contém 3 tipos de informação:
  - Atributos → tipo de dados específicos que armazenam (as variáveis globais)
  - Métodos → as operações (funções e procedimentos) que podem ser executadas
  - Construtores → como é que os objetos vão ser criados/inicializados

- Exemplo
  1. Criar uma classe Triangulo
  2. A classe tem como atributos a base e a altura do triângulo
  3. Quando é criada uma instância da classe, é necessário dar a base e a altura
  4. A classe tem como métodos
     a) Alterar a base
     b) Alterar a altura
     c) Calcular a área

Classes em Java

- Exemplo
  1. Criar uma classe Triangulo

```
class Triangulo:
```

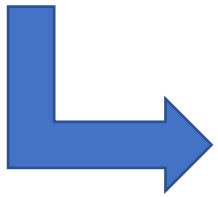
  2. A classe tem como atributos a base e a altura do triângulo

```
class Triangulo:
    base = 10
    altura = 2
```

Classes em Python

- Exemplo (continuação)
  3. Quando é criada uma instância da classe, é necessário dar a base e a altura

```
class Triangulo:
    base = 10
    altura = 2

    def __init__(self, b, a):
        self.base = b
        self.altura = a
```



O construtor da classe vai permitir criar vários triângulos. Tem de ter o nome __init__ e pode ou não ter parâmetros além do *self* (depende se no início é necessário ter informação – neste caso, o enunciado indica que o triângulo quando é criado, tem a base e a altura)

Classes em Python

4. A classe tem como métodos
   a) Alterar a base
   b) Alterar a altura
   c) Calcular a área

5. `help(Triangulo)`

devolve o texto definido na *docstring*

```
class Triangulo(builtins.object)
 |  Triangulo(b, a)
 |
 |  Methods defined here:
 |
 |  __init__(self, b, a)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  alterar_altura(self, nova_altura)
 |
 |  alterar_base(self, nova_base)
 |
 |  area(self)
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors defined here:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)
 |
 |  ----------------------------------------------------------------------
 |  Data and other attributes defined here:
 |
 |  altura = 2
 |
 |  base = 10
```

```python
'''Classe Triangulo. Permite alterar a base, a altura e calcular a área do triângulo'''
class Triangulo:
    base = 10
    altura = 2

    def __init__(self, b, a):
        self.base = b
        self.altura = a

    def alterar_base(self, nova_base):
        self.base = nova_base

    def alterar_altura(self, nova_altura):
        self.altura = nova_altura

    def area(self):
        return self.base * self.altura / 2
```

Classes em Python

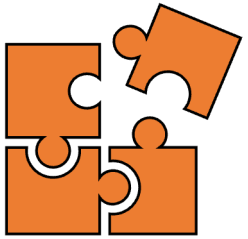
- Agora podemos criar muitos triângulos

```
# Um triângulo chamado t1 com base 2 e altura 3
t1 = Triangulo(2, 3)

# Um triângulo chamado t1 com base 5 e altura 6
t2 = Triangulo(5, 6)

# Um triângulo chamado t1 com base 5 e altura 10
t3 = Triangulo(5, 10)

# Altera a base do triângulo t2 para o valor 7
t2.alterar_base(7)

# Imprime a área do triângulo t3
print(t3.area())
```

Tarefa 14

Trabalhar sobre o seguinte documento:

- **Python Tarefa 14 – Classes e Objetos.pdf**



Comentários?