Orientação a Objetos na Linguagem Python

1. Função hasattr

hasattr() é uma função em Python que é usada para verificar se um objeto tem um determinado atributo. Veja o exemplo

```
class Cachorro:
  def __init__(self, nome, raca, idade):
    self.nome = nome
    self.raca = raca
    self.idade = idade

dog = Cachorro("Rex", "bulldog", 3)
print(hasattr(dog, "nome")) #True
print(hasattr(dog, "dono")) #False
```

2. Atributos de Classe x Atributos de Instância

Os atributos de classe são atributos compartilhados por **todos** os objetos da classe. Já os atributos de instância são aqueles que existem somente para um determinado objeto. Veja o exemplo:

```
class Cachorro:
  def __init__(self, nome, raca, idade):
    self.nome = nome
    self.raca = raca
    self.idade = idade

dog1 = Cachorro("Rex", "bulldog", 3)
dog2 = Cachorro("Lili", "poodle", 1)
dog2.cor = "branco"
print(hasattr(dog1, "cor")) #False
```

No exemplo acima, o atributo "cor" foi atributo apenas para o objeto dog2. Portanto, o atributo "cor" é um **atributo de instância**. Já os atributos "nome", "raca", "idade" são atributos de classes, pois todos os objetos da classe Cachorro terão esses atributos.

3. Comparando objetos

Para comparar dois objetos da mesma classe é necessário criar o método `__eq__` dentro da classe

```
class Cachorro:
      #método __init()___ omitido
      #método  falar() omitido
      #método getIdade() omitido
      def __eq__(self, outro_dog):
         if(self.nome == outro_dog.nome):
            if(self.raca == outro_dog.raca):
               return True
         return False
```

Quando o método `__eq__` estiver implementado dentro da classe, a comparação entre objetos poderá ser feita usando o operador ==. Exemplo:

```
dog1 = Cachorro("Rex", "bulldog", 3)
dog2 = Cachorro("Rex", "bulldog", 2)
print(dog1 == dog2)
```

4. Iterator

Para executar uma iteração sobre o próprio objeto, devemos implementar os métodos `__iter__` e `__next__` dentro da classe. Veja o exemplo:

```
class Contador:
  def __init__(self, inicio, fim):
    self.inicio = inicio
    self.fim = fim

  def __iter__(self):
    return self

  def __next__(self):
    if(self.inicio < self.fim):
      x = self.inicio
      self.inicio += 1
      return x
    else:
      raise StopIteration

c = Contador(2,7)

for i in c:
  print("valor = ", i)
```

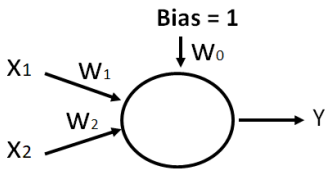
5. Exceções e erros

No exemplo anterior, se o usuário tivesse fornecido um numero de inicio maior que o número de fim, então o objeto da classe Contador não teria sido executado, pois haveria um erro. Para tratar erros devemos usar o `try` e `except`. Veja o exemplo:

```
class Contador:
  def __init__(self, inicio, fim):
    try:
      if(inicio > fim):
          raise Exception("Erro: inicio maior que
fim")
      self.inicio = inicio
      self.fim = fim
    except Exception as e:
      print(e)

  def __iter__(self):
    return self

  def __next__(self):
    if(self.inicio < self.fim):
      x = self.inicio
      self.inicio += 1
      return x
    else:
      raise StopIteration

c = Contador(21,7)#irá exibir mensagem de erro
```

Exercício

1. Implementando um perceptron



Entrada		Saida
X1	X2	
0	0	0
0	1	1
1	0	1
1	1	1

```
import numpy as np

class Neuronio:

    def __init__(self, qtdeEntradas, taxaAprendizado=0.2):
        self.pesos = np.random.rand(1, qtdeEntradas)
        self.pesoBias = np.random.random()
        self.taxaAprendizado = taxaAprendizado

    def ativacao(self, x):
        return 1 if (x >= 0) else 0 #ReLu
        #return 1.0 / (1.0 + np.exp(-x)) #sigmoid

    def derivada(self, x):
        return x #derivada ReLu
        #return x * (1.0 - x) #derivada sigmoid

    def saida(self, entrada):
        soma = np.sum(entrada*self.pesos)+self.pesoBias
        return self.ativacao(float(soma))

    def atualizarPesos(self, entrada, saidaEsperada):
        erro = saidaEsperada-self.saida(entrada)
        self.pesos +=  (self.taxaAprendizado * erro * self.derivada(entrada))
        self.pesoBias +=  (self.taxaAprendizado * erro)

    def imprimirPesos(self):
        print ("pesos: ", self.pesos)
        print ("bias: ", self.pesoBias)


X = np.array([[0,0],[0,1],[1,0],[1,1]], dtype=float) #entradas
Y = np.array([0,1,1,1], dtype=int) #saida
qtdeAmostras = Y.shape[0]

c = Neuronio(2)
c.imprimirPesos()

for epoca in range(100):
    print("epoca: ", epoca)
    qtdeAcertos = 0
    for i in range(qtdeAmostras):
        saidaObtida = c.saida(X[i])
        if(saidaObtida != Y[i]): #neuronio errou a classificação
            c.atualizarPesos(X[i], Y[i]) #vamos então atualizar os pesos
        else:
            qtdeAcertos += 1
    if(qtdeAcertos == qtdeAmostras):
        break

print(c.saida([0,0]))
print(c.saida([0,1]))
print(c.saida([1,0]))
print(c.saida([1,1]))
c.imprimirPesos()
```