**Python 3 – Orientações a Objetos**

**Curso 1 – Introdução a Orientação a Objetos:**

1. **Aula 1 – Aprenda o Paradigma OO com Python:**
   1. Ao criar uma função, para contas de um banco, por exemplo, podemos colocar como parâmetros o que desejamos que essa conta tenha, como titular, número, saldo, dentre outros.
      1. Dentro da função, criamos uma variável conta que recebe um dicionário: conta = {“titular”: titular, “numero”: numero, por exemplo, e retorna a conta no final.
   2. Para criar uma nova conta, criamos uma variável “conta” que recebe a função “cria\_conta(titular, numero, saldo...)”, com os parâmetros já definidos, desse modo, quando chamarmos essa “conta[“titular”]”, por exemplo, receberemos o nome que foi dado como parâmetro “titular” logo aqui acima.
   3. A ideia central do paradigma orientação a objeto é: Dado e funcionalidade (comportamentos) andam juntos.
   4. O que aprendemos:
      1. Dicionário;
      2. Funções;
      3. Encapsulamento de código.
2. **Aula 2 – Classes e Objetos:**
   1. Classes são como receitas de bolo, ou formas, que servem para que o projeto final seja um só.
   2. Class: Determina uma classe.
   3. Nomenclatura em classes: NomeDeUmaClasseSeEscreveAssim.
   4. Pass: Faz com que uma classe ou função passe mesmo que não tenha nada escrito ainda. Python entende que anda será escrito algo lá e não dá erro durante a execução do código.
   5. Para chamar uma classe no console, é só colocar o nome dela (com a inicial Maiúscula) seguida de (), como uma função.
   6. Quando guardamos uma classe dentro de uma variável (ex.: conta = Conta()), chamamos essa variável de referência. Ele guarda o endereço da memória onde essa classe foi alocada para saber onde encontrar esse objeto.
   7. O parâmetro *self* é o que indica a referência, a localização daquela classe.
   8. Def \_\_init\_\_(self):: É uma função construtora, responsável por criar um objeto.
      1. Funciona basicamente como a função *cria\_conta* que falamos lá acima.
      2. Logo depois do *self* colocamos outros parâmetros, como: numero, titular, saldo, dentre outros.
      3. Utilizamos o self para nos referir a locação da referência, ao local onde esse objeto está armazenado.
      4. O código final fica assim:



* + 1. Colocando os parâmetros ao invés de os números e nomes direto, torna esse molde interativo, fazendo com que possamos criar várias contas utilizando o mesmo construtor, apenas com parâmetros diferentes.
  1. Podemos criar atributos pré definidos. Caso queiramos que o limite de todas as contas seja o mesmo, menos para contas especiais, basta colocar, junto com os atributos, limite = 1000.0, por exemplo. Nesse caso, durante a criação das contas só colocamos o atributo limite naquelas que terão limites especiais, caso contrário, só precisamos atribuir até o saldo, pois o limite será adicionado automaticamente. Ex.:



* 1. Através da referência podemos acessar diretamente os atributos do objeto criado. Ex.:



* + 1. Primeira linha: variável conta recebe a classe Conta com os atributos.
    2. Segunda e Terceira linha: chama a referência e diz que a variável conta é um objeto no endereço 0x00...
    3. Quarta e Quinta linha: acesso direto ao atributo saldo da conta através da referência/variável conta e exibição do mesmo.