Python e Duck-Typing

Diferente das linguagens estáticas, o Python não fornece um meio explícito de implementar interfaces. Normalmente, em linguagens voltadas essencialmente ao paradigma OO, temos uma palavra reservada que determina que uma classe é uma interface. No Java, por exemplo, quando queremos determinar que uma classe é uma interface, utilizamos a palavra reservada ***Interface*** e nela definimos todos os métodos que queremos que uma classe implemente, caso essa implemente a interface definida.

No Python, assim como as outras linguagens de tipagem dinâmicas, utiliza um conceito chamado duck-typing. Basicamente, o duck-typing define que uma classe não precisa implementar uma determinada interface para que ela seja considerada de um tipo específico. Há uma citação bem famosa para explicar esse modelo: "Se um objeto anda como um pato e faz "quack" como um pato então ele é um pato".

Sendo assim, se temos um objeto do tipo **conta** e um objeto do tipo **venda** e ambos implementam o método **cancela(),** pouco importa se eles implementam uma interface que exige a implementação do método cancela. O que importa é que quando eu chamar o método **cancela()** através dos seus objetos, esse método será chamado.

Nas próximas aulas veremos como utilizar este conceito para implementar o conceito de interfaces de forma dinâmica no Python.