Sistema de Empréstimo de Bicicletas

Vocês farão em duplas ou trios um sistema de empréstimo de bibliotecas, que envolverá duas classes principais (Cliente, Loja) e uma para testes unitários (TestaEmprestimoBikes). O projeto deve ser apresentado na última aula do módulo (30/08/2020).

Cliente pode:

- Ver as bicicletas disponíveis na Loja;
- Alugar bicicletas por hora (R$5/hora);
- Alugar bicicletas por dia (R$25/dia);
- Alugar bicicletas por semana (R$100/semana)
- Aluguel para família, uma promoção que pode incluir de 3 a 5 empréstimos (de qualquer tipo) com 30% de desconto no valor total.

Loja pode:

- Calcular a conta quando o cliente decidir devolver a bicicleta;
- Mostrar o estoque de bicicletas;
- Receber pedidos de aluguéis por hora, diários ou semanais validando a possibilidade com o estoque.

Por questão de simplicidade vamos assumir que:

1. Cada empréstimo segue apenas um modelo de cobrança (hora, dia ou semana);
2. O cliente pode decidir livremente quantas bicicletas quer alugar;
3. Os pedidos de aluguéis só podem ser efetuados se houver bicicletas suficientes disponíveis.

Ao projetar seus objetos você deve se atentar ao que cada classe será responsável por fazer, entenda o que cada elemento pode fazer, e em seguida abstraia o problema para desenhar as classes e seus métodos. Note que nem tudo que um objeto pode fazer é necessariamente um método desse objeto.

A classe de testes deve ser feita em outro arquivo, e você deve importar as classes Loja e Cliente do arquivo principal. Utilize a classe *datetime* para trabalhar com tempo no seu programa, e na classe teste utilize a biblioteca *unittest*. Cuja a documentação, e sites auxiliares, pode ser encontrada em:

https://docs.python.org/pt-br/3/library/unittest.html

http://devfuria.com.br/python/tdd-primeiros-passos-com-testes-unitarios/

https://realpython.com/python-testing/

Note na criação de testes as seguintes dicas:

- Os nomes dos métodos de testes devem ser descritivos quanto a funcionalidade que eles testam;
- Faça testes para inputs extremos como valores nulos (None), tipos inválidos, datas inválidas e etc.
- É comum que as classes de testes sejam maiores que as próprias classes que elas testam, mas isso é uma coisa boa!
- Busquem a maior cobertura de código possível, idealmente a cobertura de 100% do programa, para isso vão construindo os testes juntamente com o desenvolvimento do programa;
- Se o teste feito está correto mas apresentou um erro, isso significa que foi encontrado um erro no seu código principal, ***conserte o código principal e não teste.***

Bons estudos a todos!