Método atualizar_preco:

O método **def atualizar_preco**, faz parte da **Classe Produto**. O mesmo tem a função de inserir atualizações ou edições nos valores do banco de dados. Necessitando assim de 3 parâmetros:

```
def atualizar_preco(self, preco, codigo):
```

SELF: Faz referência a Classe Produto, informando que pertence a Classe.

PRECO e CODIGO: Variáveis usadas para passar os valores de atualização para a função vindos do código principal.

- Primeiro, usando a variável **conexao** que recebe o método **conexao( )**, ele retorna a conexão com o banco de dados por meio das informações passadas, fazendo o login no banco:

```
 conexao= self.conexao()
```

```
def conexao(self):
        con= pymysql.connect(
            host="localhost",
            user="root",
            password="",
            database="loja_db"
        )
        return con
```

- Em seguida os parâmetros recebem as variáveis locais **preco** e **codigo**.

```
        self.preco= preco
        self.codigo= codigo
```

- Cria-se uma variável de nome **comando** que recebe uma string com uma instrução SQL que atualiza os valores de **preco** usando como parâmetro (where) de localização o valor contido em **codigo**.

```
      comando= "update produto set preco = %s where codigo = %s"
```

- Uma variável chamada **valores** é criada para receber uma tupla com os valores **self.preco e self.codigo,** eles serão passados para o SQL com a informação contida na variável comando, valores estes que substituirão o **%s** (deve-se estar na mesma ordem que os parâmetros declarados no método).

```
      valores= (self.preco, self.codigo)
```

- A variável **consulta** é criada como um **cursor( )**, que é um objeto que permite a execução de comandos SQL no banco de dados.

```
    consulta= conexao.cursor()
```

- A função **execute( )** é chamada com **consulta** usando os argumentos **comando** e **valores**. Executando assim o comando SQL que irá atualizar o preço com base no código informado.

```
    consulta.execute(comando, valores)
```

- Logo depois precisamos confirmar e aplicar as alterações feitas no banco, para isso usamos o método **commit( )**. Ele insere por definitivo as alterações feitas no banco.

```
    conexao.commit()
```

- Após o comando acima é necessário saber se as informações foram inseridas com sucesso. Usando um print chamamos o comando **rowcount( )** pois ele retorna o número de linhas afetadas pela alteração.

```
    print(consulta.rowcount, "linha foi atualizada com sucesso!")
```

- Assim que é feita as alterações desejadas, chama-se o método **close( )** para encerrar a conexão com o banco de dados.

```
conexao.close()
```