Especificação do Projeto 2

Módulo 1

LET'S CODE

Parte 1

A empresa PyCoders Ltda armazena os **contatos dos seus fornecedores em um arquivo .csv.** Nesse arquivo, existem três campos delimitados por ";", na seguinte ordem:

1. **Nome do fornecedor;**
2. **Telefone do fornecedor;**
3. **Email do fornecedor.**

Como sabemos, a ordem e os ";" no arquivo são necessários para identificar e delimitar os campos. Porém, alguém da PyCoders Ltda não sabia disso e sem querer acabou **mexendo na ordem.** Por exemplo, em algumas linhas colocou primeiro o telefone do fornecedor e depois o nome. Em outras, o email veio na primeira posição. Enfim, uma **bagunça**!

Você terá que criar um script para identificar se o que está no arquivo é o nome, telefone ou email do fornecedor. Depois, você deverá armazenar essas informações de uma **lista de listas ou dicionário** (com atributos nome, telefone e email), a qual será criada pela squad. E em seguida, **criar uma ferramenta de interface** para permitir a leitura de contatos específicos (**busca por nome, telefone ou email**), permitir a **remoção** e **adição** de **fornecedores** nessa lista de contatos, além de **armazenar** essas **informações em um .csv.**

Parte 1: Milestones

Milestone #1

Criar “constantes” para facilitar o mapeamento das posições de cada lista em relação aos atributos nome, email, e telefone. Testar e validar a criação de fornecedores quando necessário.

Milestone #2

Ler o arquivo .csv e armazenar os dados da forma correta em listas (às quais deverão estar em uma lista) ou dicionário.

Sugestão: quebrar ao máximo por funções e testar cada parte de forma separada. Por exemplo, criar uma função que dada uma string, retorna se ela é nome, telefone ou email.

Milestone #3

Criar interface com o usuário para adição e remoção de contatos, busca por contatos e armazenamento em .csv.

Parte 2

A empresa PyCoders Ltda precisa que o **armazenamento dos contatos** dos clientes seja feito de uma maneira mais bem estruturada. Para isso, definiu que deverá existir uma **agenda de contatos**, a qual deverá ter um **limite de 75 itens**. Os dados **armazenados referentes a um contato** deverão ser, **pelo menos**: **ID único**, **nome**, **sobrenome**, **telefone** e **email**.

Cada **contato** deverá ter **obrigatoriamente** um campo para **nome**, enquanto que o campo para **sobrenome não será obrigatório**, mas será apenas **um** (por um campo de nome ou sobrenome, entenda-se um campo para inserção de string, a qual pode conter espaços). O **ID deverá ser único** e **impossível de ser modificado** pelo usuário. **Emails e telefones poderão ser múltiplos**, sem limite pré-definido, mas deverão passar por um **processo de validação** a ser definido por você.

A agenda deverá ter **opções de navegação**, mostrando o **ID único, o nome e o sobrenome de todos os contatos**. Caso o usuário deseje verificar todas as informações de um contato específico, deverá selecioná-lo, sendo a **forma** como isso ocorrerá ficando **a cargo dos desenvolvedores**. Opções como **adicionar contato** (respeitando o **limite**), **remover** e **alterar** devem ser possíveis.

Os **contatos poderão ser agrupados em grupos definidos pelo usuário**, o qual pode criar **grupos** (“empresa A”, “público-alvo”, etc.) **dinamicamente** durante navegação na agenda. **Na navegação** deve existir também uma opção para mostrar **apenas contatos de um determinado grupo.**

Parte 2: Milestones

Milestone #1

Definir estrutura das classes a serem utilizadas.

Milestone #2

Popular as classes com os dados da Parte 1.

Milestone #3

Criar ambiente de navegação com as opções definidas na especificação.

Sobre a entrega

1) **Deadline:** 8/novembro, domingo, 23:59
2) Enviar **apenas um projeto por squad** para o email **flavio.nakasato@gmail.com** com o assunto: "**Projeto 2 - <Squad>**"
3) O programa deve ser entregue em **um** ou **mais** arquivos **.py** - estão livres para usar o Jupyter para **prototipar e testar**, mas a versão final deve rodar **a partir do terminal/prompt de comando**
4) Incluam um arquivo **README.md** com todas as **explicações necessárias para rodar o projeto**. Não precisam se preocupar tanto com detalhes de formatação, mas, se quiserem usar mais recursos de *markdown*, podem checar aqui: https://guides.github.com/features/mastering-markdown/
5) A menos que se peça explicitamente para se seguir certo caminho, há **diversas formas válidas** de se resolver os problemas deste projeto. Vocês estão **livres para escolher** a maneira que julgarem melhor, **desde que resolvam o problema levantado pelo cliente**
6) A **interpretação** dos textos e questões **faz parte do projeto**
7) **Legibilidade importa!** Tentem se **claros** no código e usem, quando pertinente, **comentários** e **docstrings**