07

Mãos à obra!

Vamos começar criando o arquivo principal para começar a realizar a leitura do arquivo `contatos.csv`.

Neste arquivo, pedimos para o Python abrir o arquivo que está localizado em `dados/contatos.csv`. Logo, utilizamos a função `open()` e atribuímos seu retorno para uma variável:

```
arquivo_contatos = open('dados/contatos.csv')
```

Podemos ler todas as linhas do arquivo utilizando o método `readlines()` e atribuindo seu retorno a uma variável, que no caso, vou chamá-la de conteúdo:

```
arquivo_contatos = open('dados/contatos.csv')

conteudo = arquivo_contatos.readlines()
```

Esse método devolve uma lista na qual cada item da lista é uma linha do arquivo. Podemos falar para o Python iterar sobre essa lista e mostrar o resultado na tela:

```
arquivo_contatos = open('dados/contatos.csv')

conteudo = arquivo_contatos.readlines()

for linha in conteudo:
    print(linha)
```

Quando pedimos para o Python rodar esse arquivo, dependendo do sistema operacional, pode ser lançada uma exceção, ou, então, os caracteres podem ser impressos com uma codificação estranha (não representando o caractere que está no arquivo).

Isso tudo acontece por causa do **encoding** em que o arquivo está salvo. A fim de resolver esse problema, nós podemos falar para o Python abrir o arquivo com o *enconding* correto, logo, ele vai conseguir decodificar o arquivo corretamente. Para isso, basta adicionar o parâmetro `encoding` na função `open()`:

```
arquivo_contatos = open('dados/contatos.csv', encoding='latin_1')

conteudo = arquivo_contatos.readlines()

for linha in conteudo:
    print(linha)
```

Rodando o arquivo pelo terminal, vemos que cada linha no arquivo é impressa na tela e que existe uma quebra de linha adicional em cada linha impressa. Isso ocorre porque a quebra de linha é interpretada pelo caractere `\n`, e a função `print()`, por padrão, já insere uma quebra de linha adicional ao final de cada impressão. Por isso, podemos passar o parâmetro `end` falando que não queremos a quebra de linha da função `print()`:

```
arquivo_contatos = open('dados/contatos.csv', encoding='latin_1')

conteudo = arquivo_contatos.readlines()

for linha in conteudo:
    print(linha, end='')
```

Bacana! Essa forma de ler um arquivo funciona. Porém, quando utilizamos o método `readlines()` todo o conteúdo do arquivo é carregado na memória. Isso pode trazer alguns problemas de performance, então, para realizar a leitura de uma forma mais performática para com o gerenciamento de memória, podemos iterar diretamente a variável `arquivo_contatos` sem invocar o método `readlines()` antes:

```
arquivo_contatos = open('dados/contatos.csv', encoding='latin_1')

for linha in arquivo_contatos:
    print(linha, end='')
```

Dessa forma, cada linha do arquivo será carregada na memória à medida que avançarmos em sua leitura.